RM0511
Reference Manual

STM32WL30xx/31xx/33xx Arm®-based wireless MCUs with sub-GHz radio solution

Introduction

This reference manual provides complete information on how to use the STM32WL30xx/31xx/33xx MCU memory and peripherals.

The STM32WL30xx/31xx/33xx are powerful and ultra-low-power microcontrollers with sub-GHz radio with an Arm® Cortex®-M0+ core which can operate with various clock sources.

The STM32WL30xx/STM32WL31xx/STM32WL33xx are microcontrollers with different memory sizes, packages and peripherals.The devices include ST state-of-the-art patented technology.

STM32WL33xx TX-only device variants (STM32WL33xx-X) are also available. These variants support only the transmitter capability of the Sub-GHz radio IP (MRSUBG).

In this document, any reference to STM32WL33xx devices is also valid for STM32WL33xx-X devices (the only exception being the Radio receiver capability).

Related documents

- STM32WL30xx datasheet (DS14855)
- STM32WL31xx datasheet (DS14836)
- STM32WL33xx datasheet (DS14221)

For information on device errata with respect to the datasheets and reference manuals:

- STM32WL30xx errata sheet (ES0653)
- STM32WL31xx errata sheet (ES0652)
- STM32WL33xx errata sheet (ES0612)

Contents

List of tables

List of figures

1 General information

For information on the Arm®[a] Cortex®-M0+ core, refer to the corresponding Arm® Technical Reference Manual, available from the http://www.arm.com website.

1.1 List of abbreviations for registers

The following abbreviations are used in register descriptions:

read/write (rw or R/W)	Software can read and write to these bits.
read-only (r or R)	Software can only read these bits.
write-only (w or W)	Software can only write to this bit. Reading the bit returns the reset value.
read/write-once (RWOnce)	Software can read these bits but write is only allowed once.
read/clear (rc_w1 or RWC1)	Software can read as well as clear this bit by writing 1. Writing ‘0’ has no effect on the bit value.
read/clear (rc_w0 or RWC0)	Software can read as well as clear this bit by writing 0. Writing ‘1’ has no effect on the bit value.
read/clear by read (rc_r or RC)	Software can read this bit. Reading this bit automatically clears it to ‘0’. Writing ‘0’ has no effect on the bit value.
read/set (rs or RWS1)	Software can read as well as set this bit. Writing ‘0’ has no effect on the bit value.
read-only write trigger (rt_w or RWH)	Software can read this bit. Writing ‘0’ or ‘1’ triggers an event but has no effect on the bit value.
toggle (t or RWT1)	Software can only toggle this bit by writing ‘1’. Writing ‘0’ has no effect.
Reserved (Res.)	Reserved bit, must be kept at reset value.

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

1.2 Glossary

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

- The SoC integrates
  - SWD debug port (SWD-DP) provides a 2-pin (clock and data) interface based on the Serial Wire Debug (SWD) protocol.
- Word: data/instruction of 32-bit length.
- Half word: data/instruction of 16-bit length.
- Byte: data of 8-bit length.
- Double word: data of 64-bit length.
- AHB: advanced high-performance bus.
- APB: advanced peripheral bus.
- CPU: refers to the Cortex®-M0+ core.

1.3 Acronyms

Table 1. List of acronyms

Acronym	Description
Absolute time	The absolute time is based only on the slow clock and expressed in slow clock time unit. The absolute time represents only the 28 MSB information (without interpolated part) of the ABSOLUTE_TIME[31:0] Misc register
Interpolated absolute time	The **interpolated** absolute time is based on the interpolated clock and expressed in 16 x slow clock time unit. The **interpolated** absolute time is represented by the full ABSOLUTE_TIME[31:0] Misc register.
ADC	Analog to digital converter
AES	Advanced encryption standard hardware accelerator
AFC	Automatic frequency compensation
AGC	Automatic gain control
AHB	Advanced high-performance bus
APB	Advanced peripheral bus
BOR	Brown out reset
BPU	Breakpoint unit (Arm debug component)
CRC	Cyclic redundancy check
DBG	Debug
DBM	Data buffer manager
DC	Direct current
DLL	Delay-locked loop
DMA	Direct memory access
DMAMUX	Direct memory access multiplexer

Table 1. List of acronyms

Acronym	Description
FSM	Finite state machine
f_{ref} or f_{XO}	In this document, f_{ref} or f_{XO} means the frequency of the high frequency XO (may be 48 or 50 MHz, typically 48 MHz)
f_{SYS}	In this document, f_{SYS} indicates the digital fast clock provided by the SoC, equivalent to XO / 3 (typically 16 MHz)
DWT	Data watchpoint and trace (Arm debug component)
GPIO	General purpose input output
HSE	High speed external clock oscillator
HSI	High speed internal clock oscillator
HW	Hardware
I2C	Inter Integrated circuit (communication standard)
I2S	Inter Integrated (communication standard)
IIR	Infinite impulse response
IRQ	Interrupt request
ITM	Instrumentation trace macrocell (Arm debug component)
LCSC	LC sensor controller
IWDG	Independent watchdog
LDO	Low drop output
LDO	Low drop output
LP	Low power
LSB/b	Least significant byte/bit
LPAWUR	Low power autonomous wakeup radio: acronym given to the signals/registers sub-blocks of the Wakeup Radio IP
LSE	Low speed external clock oscillator
LSI	Low speed internal clock oscillator
MCU	Micro controller unit
MPU	Memory protection unit
MR_SubG	Digital radio IP for sub-GHz RF
MSB/b	Most significant byte/bit
NVIC	Nested vector interrupt controller
OBL	Option byte loading
OSC	Oscillator
OTP	One time programmable
PA	Power amplifier
PDR	Power down reset
PLL	Phase locked loop

Table 1. List of acronyms

Acronym	Description
POR	Power on reset
PVD	Programmable voltage detector
PVM	Peripheral voltage monitoring
RC	Resistor capacitor oscillator
RCC	Reset and clock controller
RF	Radio frequency
RFSUBG	Analog radio block used with the MR_SubG IP.
ROM	Read only memory
RRM	Radio register manager
RSSI	Received signal strength indicator
RTC	Real time clock
Rx	Reception
SMPS	Switch mode power supply
SoC	System on chip.
SPI	Serial peripheral interface (communication standard)
SRAM	Static random access memory
SW	Software
SWD	Single wire debug
SWJ-DP	Single wire joint test access group - debug port (Arm debug component)
SYSCFG	System configuration
TIM	Timer
Tx	Transmission
UDRA	Unified direct register access (part of the RRM block)
USART	Universal synchronous asynchronous receiver transmitter (communication standard)
UDRA	Unified direct register access (part of the RRM block)
VCO	Voltage controlled oscillator
VREF	Voltage reference
WFI	Wait for instruction (arm instruction entering low power mode)
WKUP	Wakeup
WRP	Write protection
WDG	Watch dog
XO	External crystal oscillator

1.4 Related documents

Table 2. Document references

Document reference	Name	Author	Revision
[1]	STM32WL30xx datasheet (DS14855)	STMicroelectronics	Latest revision
[2]	STM32WL31xx datasheet (DS14836)		
[3]	STM32WL33xx datasheet (DS14221)		
[4]	STM32WL30xx device errata (ES0653)		
[5]	STM32WL31xx device errata (ES0652)		
[6]	STM32WL33xx device errata (ES0612)		

2 Memory and bus architecture

2.1 System architecture

The main system consists of 32-bit multilayer AHB bus matrix that interconnects:

- Three masters:
  - CPU (Cortex®-M0+) core S-bus
  - DMA1
  - Sub-1 GHz radio subsystem
- Seven slaves:
  - Internal flash memory on CPU (Cortex®-M0+) S bus
  - Internal SRAM0 (up to 16 Kbytes)
  - Internal SRAM1 (16 Kbytes) only on STM32WL33xx device
  - APB0 peripherals (through an AHB to APB bridge)
  - APB1 peripherals (through an AHB to APB bridge)
  - AHB0 peripherals
  - AHBRF including AHB to APB bridge, and radio peripherals (connected to APB2)

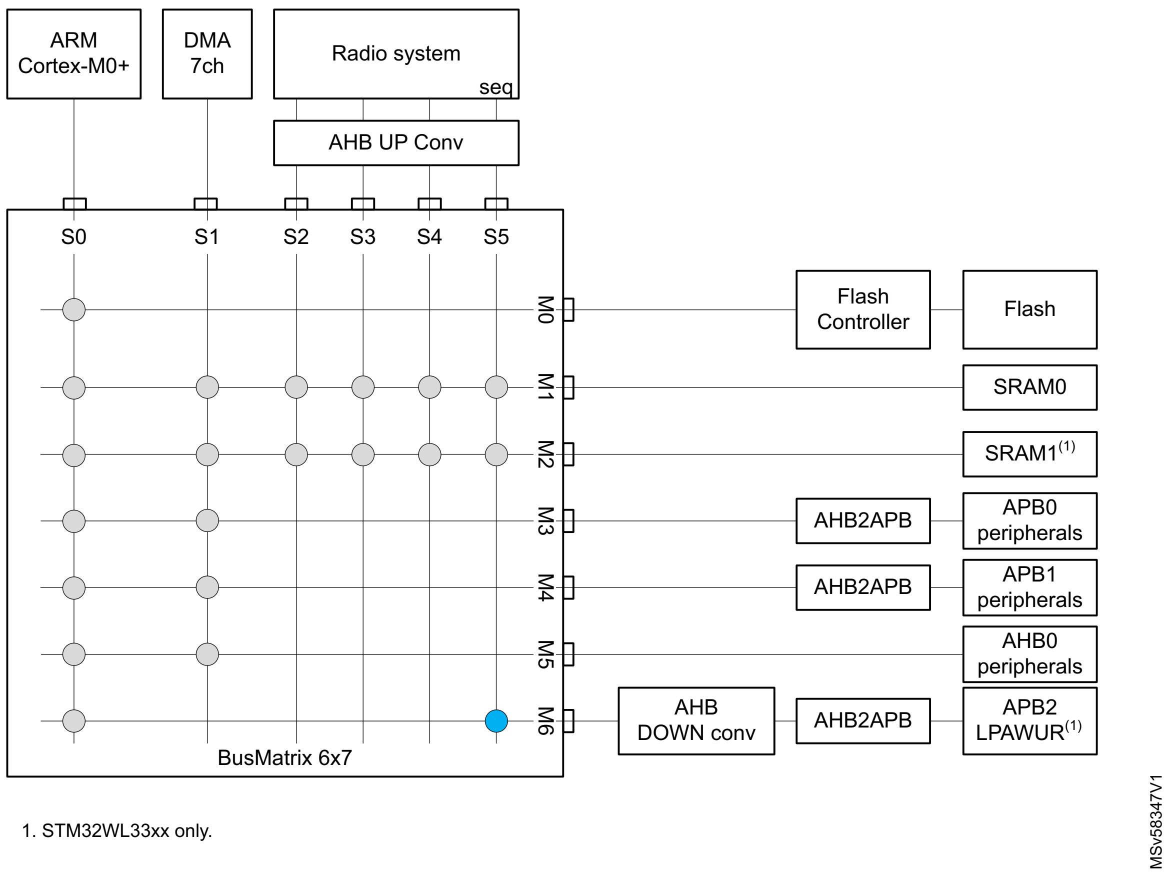
The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in *Figure 1: System architecture*:

The system consists of a Cortex®-M0+ “Radio protocol and application” processor with its radio sub-system.

There is a single flash memory to be used by the CPU for both Sub-1 GHz protocols and application management.

The peripherals are located on the different system buses (AHB, APB0, APB1, APB2 for the radio system). There are up to 2 SRAM banks, SRAM0, which is always power supplied and SRAM1 (only on STM32WL33xx) that can be programmed to be always on or switchable. For more information see *Control register 2 (PWRC_CR2)*.

Figure 1. System architecture



1. STM32WL33xx only.

2.1.1 S0: CPU (Cortex®-M0+) S-bus

This bus connects the system bus of the CPU core to the BusMatrix. This bus is used by the core to fetch instructions, for literal load and debug access, and access data located in a peripheral or SRAM area. The targets of this bus are all the possible peripherals (the internal flash and SRAM memories, the AHB0, APB0, APB1 and APB2 peripherals).

2.1.2 S1: DMA-bus

This bus connects the AHB master interface of the DMA to the BusMatrix.The targets of this bus are the twoSRAM0 and SRAM1 (only on STM32WL33xx) banks and the AHB0, APB0 and APB1 peripherals.

2.1.3 S2-S5: Sub-1 GHz radio system-bus

This bus connects the 4 AHB master interfaces of the Radio system to the BusMatrix through a dedicate 4x1 BusMatrix. The targets of this bus are the two SRAM0 and SRAM1 (only on STM32WL33xx) banks and, only for the Sequencer AHB master port of the MR_SUBG IP, the APB peripherals internal to the MR_SUBG IP.

2.1.4 Bus matrix

The bus matrix manages the access arbitration between masters. The arbitration uses a round-robin algorithm. The bus matrix is composed of three masters (CPU, DMA1-bus and radio-system bus) and seven slaves (FLASH, SRAM0, SRAM1 (STM32WL33xx only), APB0 and APB1, AHB0 and AHBRF).

Bus matrix 4x1

The bus matrix 4x1 manages the access arbitration between 4 AHB master ports of the sub-GHz radio system. The arbitration uses a round-robin algorithm. The BusMatrix 4x1 is composed of four masters (Radio system-bus) and one slave (the main BusMatrix).

AHB/APB bridges

The two bridges AHB to APB0 and AHB to ABP1 provide full synchronous connections between the AHB and the two APB buses.

All the AHB and APB ports of the MR_SUBG radio controller work at fixed 16 MHz frequency clock. Two blocks are added on the AHB/APB path to manage potential prescaled MR_SUBGHz frequency versus the system frequency:

- AHB up converter is used for the MR_SUBG AHB master transactions towards the SRAM.
- AHB down converter is used for the CPU AHB master transactions towards the MR_SUBG and LPAWUR (STM32WL33xx only) APB registers.

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the address mapping of the peripherals connected to this bridge.

After each device reset, all peripheral clocks are disabled (except for the SRAM and flash memory interface). Before using a peripheral the user has to enable its clock in the RCC_AHBxENR and the RCC_APBxENR registers.

Note: When a 16- or 8-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 16- or 8-bit data to feed the 32-bit vector.

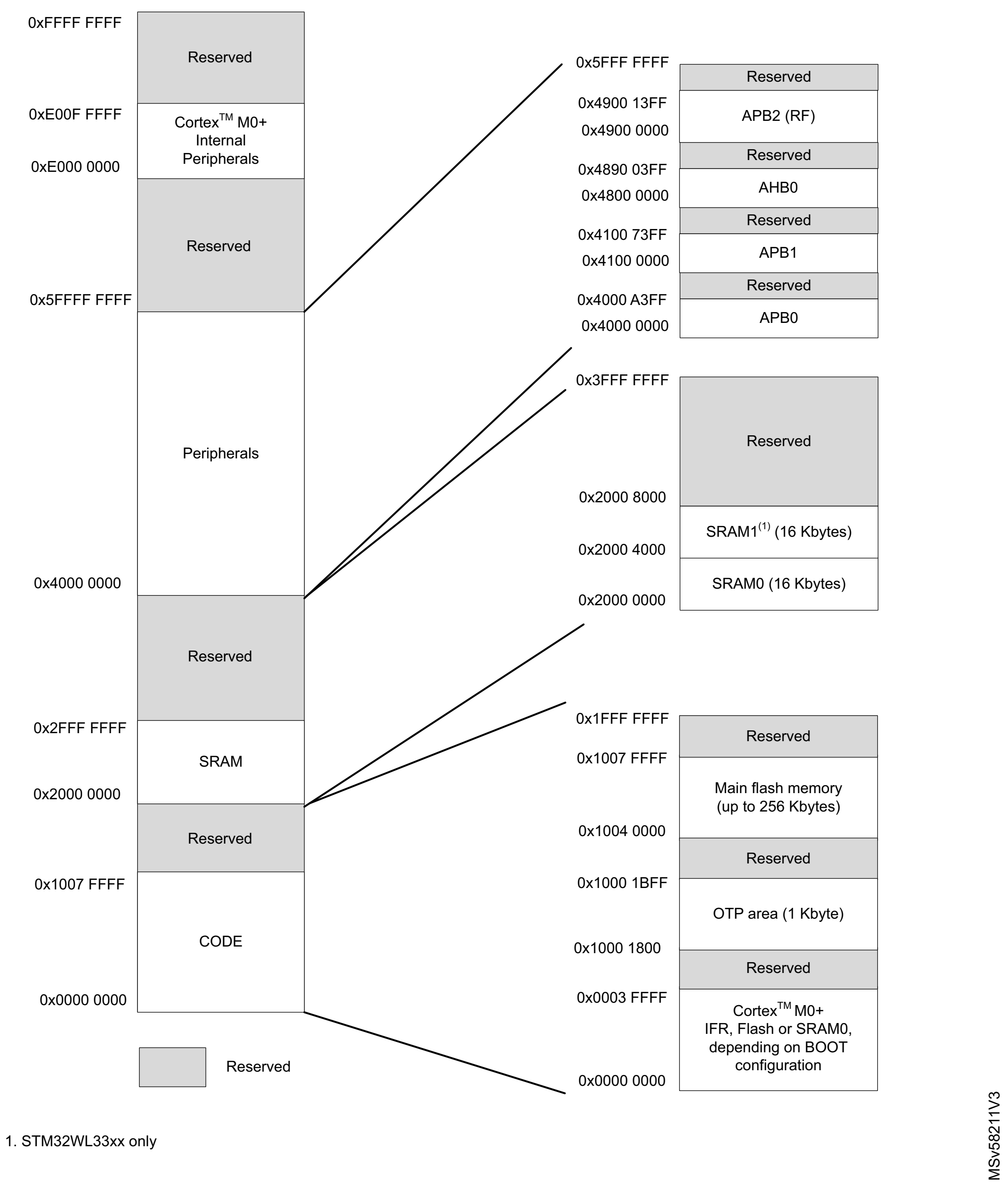
2.2 Memory organization

2.2.1 Introduction

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

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

Figure 2. Memory map



1. STM32WL33xx only

All the memory areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. For the detailed mapping of available memory and registers areas, refer to the chapter and peripheral chapters.

2.2.2 Memory map and register boundary addresses

The *Table 3* gives the boundary addresses of the peripherals available in the device.

Table 3. Memory map and peripheral register boundary addresses

Bus	Boundary address	Actual size (bytes)	Peripheral	Peripheral register map
Misc	0xE00F FFFF - 0xFFFF FFFF	255 M	Reserved	-
	0xE000 0000 - 0xE00F FFFF	1 M	Private peripheral bus	Cortex-M0+ registers (interrupt controller, SysTick, and so on.)
APB2	0x4900 1000 - 0x4900 13FF	1 K	LPAWUR	SubGHz wakeup radio registers
	0x4900 0000 - 0x4900 07FF	2 K	MR_SUBG	MR_SUBG registers
AHB0	0x4890 0000 - 0x4890 03FF	1 K	AES	See *Section 18.7: AES register map*
	0x4880 0000 - 0x4880 03FF	1 K	DMAMUX	See *Section 11.6: DMAMUX register map*
	0x4870 0000 - 0x4870 00FF	256	DMA slave	See *Section 10.5: DMA register map*
	0x4860 0000 - 0x4860 0FFF	4 K	RNG	See *Section 17.2: RNG registers map*
	0x4850 0000 - 0x4850 03FF	1 K	PWRC	See section *Section 5.7: PWRC register map*
	0x4840 0000 - 0x4840 03FF	1 K	RCC	See section *Section 6.7: RCC register map*
	0x4830 0400 - 0x4830 13FF	4 K	Reserved	-
	0x4830 0000 - 0x4830 03FF	1 K	Reserved	-
	0x4820 0000 - 0x4820 03FF	1 K	CRC	See *Section 19.5: CRC register map*
	0x4810 0000 - 0x4810 03FF	1 K	GPIOB	See section *Section 7.5: GPIO register map*
	0x4800 0000 - 0x4800 03FF	1 K	GPIOA	See section *Section 7.5: GPIO register map*
APB1	0x4100 7000 - 0x4100 73FF	1 K	SPI3	See *Section 27.10: SPI/I2S register map*
	0x4100 6000 - 0x4100 63FF	1 K	ADC	See *Section 12.7: ADC register map*
	0x4100 5000 - 0x4100 53FF	1 K	LPUART	See *Section 26.5.11: LPUART register map*
	0x4100 4000 - 0x4100 43FF	1 K	USART	See *Section 25.8: USART register map*
	0x4100 3000 - 0x4100 33FF	1 K	Reserved	-
	0x4100 2000 - 0x4100 23FF	1 K	SPI1	See *Section 27.10: SPI/I2S register map*
	0x4100 1000 - 0x4100 13FF	1 K	I2C2	See *Section 24.7: I2C register map*
	0x4100 0000 - 0x4100 03FF	1 K	I2C1	See *Section 24.7: I2C register map*

Table 3. Memory map and peripheral register boundary addresses

Bus	Boundary address	Actual size (bytes)	Peripheral	Peripheral register map
APB0	0x4000 A000 - 0x4000 A3FF	1 K	LCSC	See *Section 15.9: LCSC register map*
	0x4000 9000 - 0x4000 93FF	1 K	COMP	See *Section 13.7: COMP register map*
	0x4000 8000 - 0x4000 83FF	1 K	DBGMCU	See *Section 30.4: DBG register map*
	0x4000 7000 - 0x4000 73FF	1 K	LCDC	See *Section 16.7: LCD register map*
	0x4000 6000 - 0x4000 63FF	1 K	DAC	See *Section 14.6: DAC register map*
	0x4000 5000 - 0x4000 53FF	1 K	TIM16	See *Section 21.4: TIM16 registers*
	0x4000 4000 - 0x4000 43FF	1 K	RTC	See *Section 22.7: RTC register map*
	0x4000 3000 - 0x4000 33FF	1 K	IWDG	See *Section 23.5: IWDG register map*
	0x4000 2000 - 0x4000 23FF	1 K	TIM2	See *Section 20.5: TIM2 register map*
	0x4000 1000 - 0x4000 1FFF	4 K	FLASH_CTRL	See *Section 9.6: Flash controller register map*
	0x4000 0000 - 0x4000 03FF	1 K	SYSTEM_CTRL	See *Section 8.3: System controller register map*
SRAM	0x2000 8000 - 0x3FFF FFFF	512 K	Reserved	-
	0x2000 4000 - 0x2000 7FFF	16 K	SRAM1 (only on STM32WL33xx)	-
	0x2000 0030 - 0x2000 3FFF	16 340	SRAM0	Start of user RAM
	0x2000 0000 - 0x2000 002F	48		Reserved. See *Table 4: SRAM0 reserved locations*
Flash	0x1004 0000 - 0x1007 FFFF	Up to 256 K	Main flash	-
DESIG	0x1000 1EF0 - 0x1007 1EF7	-	-	*Section 31: Device electronic signature (DESIG)*
OTP	0x1000 1800 - 0x1000 1BFF	1 K	OTP area	*Section 31: Device electronic signature (DESIG)* applies only to the DESIG
Boot(1)	0x0000 0000 - 0x0003 FFFF	Up to 256 K	CPU Boot area	Mirroring of flash or SRAM0

1. This area is a mirroring area. The CPU accesses are redirected to other memory map depending in REMAP bit located in the flash controller CONFIG register. See *Table 5* for remapping detail.

Note: 1 Comparator, DAC, I2C2, LCSC, LCD, LPAWUR are available only on STM32WL33xx devices.

2 ADC, temperature sensor, I2C1, SPI1, I2S are available only on STM32WL31xx/33xx devices.

Table 4. SRAM0 reserved locations

Address	Actual size (bytes)	Identifier	Description
0x2000 0004	4	Reserved1	Reserved for future use.
0x2000 0008	4	SavedMSP	Used by the LPM module to save context information pointer.
0x2000 000C	4	WakeupFromSleepFlag	Used by the LPMmodule. Indicates whether the system has been woken up from Deepstop mode.
0x2000 0010	4	ResetReason	Copy of last reset reason from register RCC_CSR. The register is read, copied to this location by the bootloader code and finally cleared. As a consequence, software reading the register always reads 0. Users should read this location to know the last reset reason.
0x2000 0014	4	AppBase	Relocation of application base. The Bootloader jumps to the location pointed to by this value when a wakeup from DeepStop occurs.
0x2000 0018	4	bootloader_vr	Bootloader virtual register
0x2000 001C	4	bootloader_vr1	Bootloader virtual register1
0x2000 0020	16	Reserved2	Reserved for future use.

Table 5. Address remapping depending on the REMAP bit

REMAP	Memory mapped
0	Main flash memory
1	SRAM0

2.3 ARM® Cortex®-M0+

Table 6. Address remapping depending on REMAP bit

REMAP	Memory mapped
0	Main flash
1	SRAM0

The ARM® Cortex®-M0+ processor was developed to provide a low-cost platform that meets the needs of MCU implementation, with a reduced pin count and low-power consumption, while delivering outstanding computational performance and an advanced response to interrupts.

The embedded Cortex-M0+ embeds:

- four breakpoints
- two watchpoints
- a MPU (Memory Protection Unit) providing eight unified protection regions
- a SysTick running only with the system clock (external clock option not supported).

2.3.1 CPU memory remap

Following CPU boot the application software can modify the memory map at address 0x0000 0000. This modification is performed by programming the REMAP bit in the flash controller (see *Configuration register (CONFIG)*).

The following memory can be remapped:

- Main flash memory
- SRAM0 memory

Embedded boot loader

ST provides a boot loader executed after each CPU reboot. This boot loader has its own documentation.

Note: STM32WL30xx/31xx/33xx devices latch the PA8 / PA9 / PA10 / PA11 / PB12 / PB13 / PB14 / PB15 pads value at POR. The information is available in PWRC_SR2 register (see Status register 2 (PWRC_SR2)). One of those eight I/Os can be used by the boot loader as boot indication between a normal boot or a boot on serial interface.

2.3.2 Interrupts

Interrupts are handled by the Cortex-M0+ Nested Vector Interrupt Controller (NVIC). The NVIC controls specific Cortex-M0+ interrupts (address 0x00 to 0x3C) as well as 32 user interrupts (address 0x40 to 0xBC). In STM32WL30xx/31xx/33xx devices, the user interrupts have been connected to the interrupt signals of the different peripherals (GPIO, flash controller, timer, UART,...). These interrupts can be controlled using the ISER, ICER, ISPR and ICPR registers (see the *Cortex-M0+ Devices Generic User Guide*).

Vector table

On reset, the Cortex-M0+ vector table is fixed at address 0x0000_0000. The software may relocate the vector table address to a different memory location, in a range 0x0000_0000 to 0xFFFF_FF80 in multiples of 256 bytes through the Vector Table Offset Register (VTOR) located in the Cortex-M0+ registers area (see *Cortex-M0+ Devices Generic User Guide*).

The STM32WL30xx/31xx/33xx interrupt-vector mapping is detailed in *Table 7*.

Table 7. Interrupt vectors(1)(2)(3)

Position	Priority	Type of priority	Acronym	Description	Address offset
-	-	-	-	Initial main SP	0x0000_0000
-	-3	Fixed	Reset	Reset_Handler	0x0000_0004
-	-2	Fixed	NmiSR	NMI_Handler	0x0000_0008
-	-1	Fixed	FaultISR	HardFault_handler	0x0000_000C
-	-	Reserved	Reserved	Reserved	0x0000_000C - 0x0000_0038
-	6	Settable	SysTick	System tick timer	0x0000_003C
0	Init 0	Settable	NVM	Non-volatile memory (flash) controller	0x0000_0040
1	Init 0	Settable	RCC	Reset and clock controller	0x0000_0044
2	Init 0	Settable	BATTERY	PVD / BORH	0x0000_0048
3	Init 0	Settable	I2C1	I2C1 interrupt	0x0000_004C
4	Init 0	Settable	I2C2	I2C2 interrupt	0x0000_0050
5	Init 0	Settable	SPI1	SPI1 interrupt	0x0000_0054
6	Init 0	Reserved	Reserved	Reserved	0x0000_0058
7	Init 0	Settable	SPI3	SPI3 interrupt	0x0000_005C
8	Init 0	Settable	USART	USART interrupt	0x0000_0060
9	Init 0	Settable	LPUART	LPUART interrupt	0x0000_0064
10	Init 0	Settable	TIM2	TIM2 interrupt	0x0000_0068
11	Init 0	Settable	RTC	RTC interrupt	0x0000_006C
12	Init 0	Reserved	Reserved	Reserved	0x0000_0070
12	Init 0	Settable	ADC	ADC interrupt	0x0000_0070
13	Init 0	Settable	AES	AES interrupt	0x0000_0074
14	Init 0	Reserved	Reserved	Reserved	0x0000_0078

Table 7. Interrupt vectors(1)(2)(3)

Position	Priority	Type of priority	Acronym	Description	Address offset
15	Init 0	Settable	GPIOA	GPIOA interrupt	0x0000_007C
16	Init 0	Settable	GPIOB	GPIOB interrupt	0x0000_0080
17	Init 0	Settable	DMA	DMA interrupt	0x0000_0084
18	Init 0	Settable	LPAWUR	LPAWUR interrupt	0x0000_0088
19	Init 0	Settable	COMP	Comp interrupt through SYSCFG	0x0000_008C
20	Init 0	Settable	MRSUBG_BUSY	MRSUBG Busy interrupt	0x0000_0090
21	Init 0	Settable	MRSUBG	MRSUBG interrupt	0x0000_0094
22	Init 0	Settable	MRSUBG_TX_RX_SEQUENCE	MRSUBG TX/RX sequence interrupt	0x0000_0098
23	Init 0	Settable	MRSUBG_TIMER_CPU_WKUP	MRSUBG TIMER CPU Wakeup interrupt	0x0000_009C
24	Init 0	Settable	MRSUBG_WKUP	MRSUBG Wakeup interrupt	0x0000_00A0
25	Init 0	Settable	DAC	DAC interrupt	0x0000_00A4
26	Init 0	Settable	TIM16	TIM16 interrupt	0x0000_00A8
27	Init 0	Settable	LCD	LCD interrupt	0x0000_00AC
28	Init 0	Settable	LCSC	LCSC interrupt	0x0000_00B0
29	Init 0	Settable	LCSC	LCSC LC_ACTIVITY interrupt	0x0000_00B4
30- 31	Init 0	Reserved	Reserved	Reserved	0x0000_00B8- 0x0000_00BC

1. Priority level is set to 0 after reset, each interrupt can be programmed with 4 higher priorities.
2. Comparator, DAC, I2C2, LCSC, LCD, LPAWUR are available only in STM32WL33xx device.
3. ADC, temperature sensor, I2C1, SPI1, I2S are available only in STM32WL31xx/33xx devices.

Interrupt activation sequence

Safely activating a peripheral interrupt requires the following steps:

- Make sure the interrupt is disabled and cleared on the peripheral side (this prevents receiving an interrupt due to a previous event)
- Clear pending requests for this interrupt on the Cortex-M0+ side using the NVIC ICPR register.
- Set the priority using the NVIC IPR0-IPR7 registers.
- Activate on the Cortex-M0+ side using the NVIC ISER register.
- Activate on the peripheral side.

Note: Most peripherals require clearing interrupt requests on the peripheral side when handling interrupt service requests to prevent triggering continuous interrupts for the same event.

For more details on the Cortex-M0+ exception model, see *ARMv6-M Architecture Reference Manual*, §B1.5.

For more details on the Cortex-M0+ NVIC behavior and registers, see the *ARMv6-M Architecture Reference Manual*, §B3.4.

3 AHB up/down converter

STM32WL30xx/31xx/33xx devices can support several system clock frequencies (see *Figure 13: System clock details*).

The SUBG IP does not need more than 16 MHz to achieve the processing of the radio transfers while the system (CPU, DMA, memories) may require higher performance for application purpose.

To avoid useless overconsumption, AHB up/down converter block has been added to introduce an adjustable divider by one, two or four on AHB and APB bus of the MR_SUBG (linked to AHBRF / APB2 bridge) versus the system bus matrix frequency. This block allows dividing by one, two or four the system clock for the MR_SUBG IP of the device.

When the system and the MR_SUBG share the same frequency, the AHB up/down converter block only transfers the AHB signals from one clock domain to the other.

Note: The system clock must be always equal or faster than MR_SUBG clock when radio is used (no other frequencies).

3.1 AHB up/down converter description

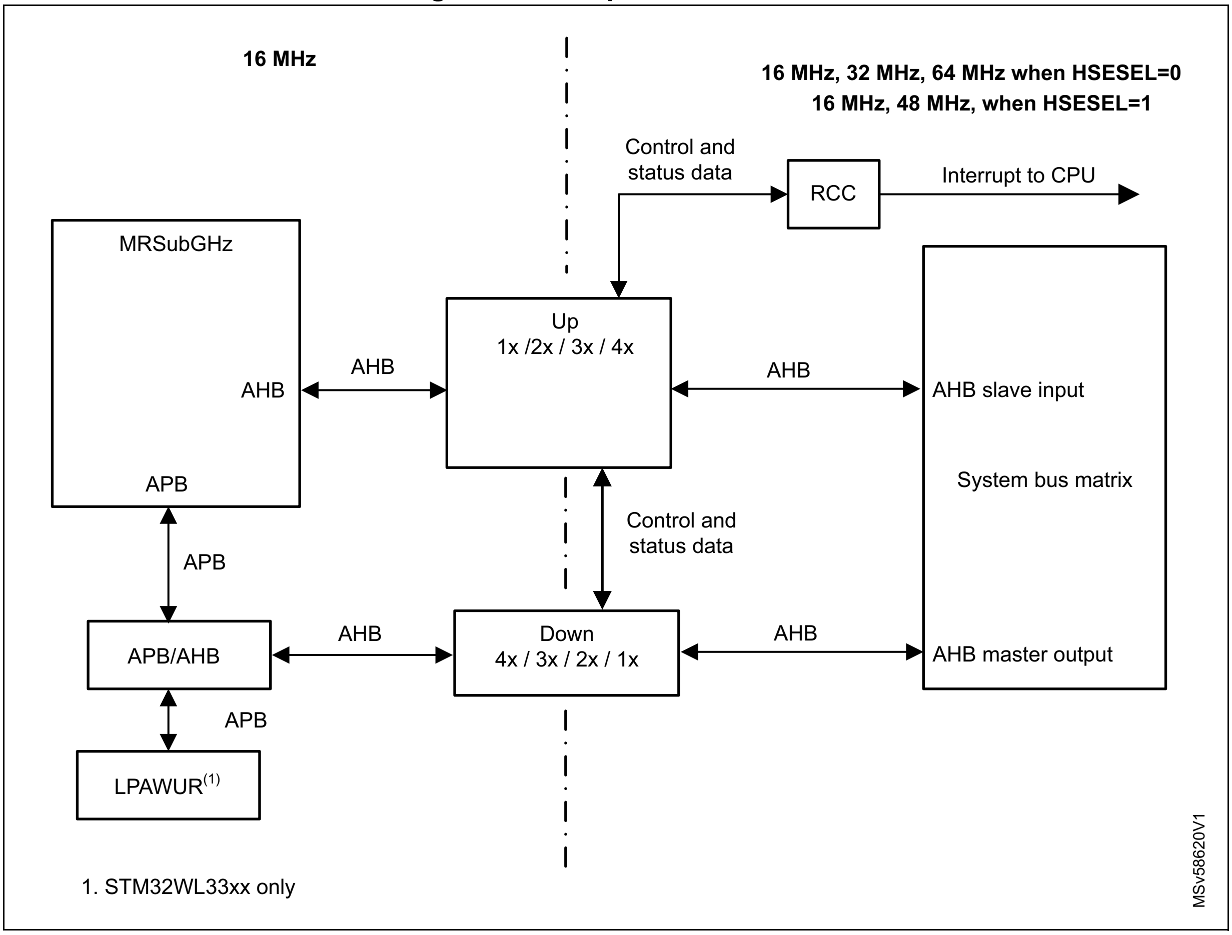
The AHB up/down converter role is to allow STM32WL30xx/31xx/33xx devices to support a fast system clock (up to PLL clock frequency output).

The AHBUPCONV block manages:

– proper on-the-fly (up-to-down and down-to-up) frequency switching by safely updating the ratio (one, two, four) between the system and the MR_SUBG IP frequencies.

– AHB and APB data transfer between MR_SUBG running at the same frequency or half or quarter of the frequency of the rest of the system (AHB and APB) that may run up to PLL clock frequency output.

Figure 3. AHB up/down converter



The management of data transfer versus clock domain and possible clock switch request is done using state machines:

- one for the AHB master up converter
- one for the AHB slave down converter.

When a CPU/system clock frequency switch is needed (activate or deactivate the divider by two between the system and the MR_SUBG), the user must request the new system clock targeted frequency in the RCC_CSCMDR.REQUEST bit (see *Clock switch command register (RCC_CSCMDR)* for details).

When receiving a new divider ratio to apply (from the RCC), the AHBUPCONV block:

- Informs the AHBDOWNCONV block (managing the APB transfer from the CPU to the MR_SUBG APB registers)
- Check the traffic on the AHB bus between MR_SUBG and CPU:
  - if no transfer is on going, it uses the HREADY signal on the bus to hold any potential transaction that could occur during the clock frequency switch.

Note: To respect the AHB lite protocol, the HREADY signal is fallen down only after the address phase of a new transfer, the new transfer phase data being stored internally in the converter.

    – if an AHB transfer is on-going, it wait until the current AHB transfer ends and then hold the AHB traffic as explain above.

- In parallel, the AHBDOWNCONV block does the same action to hold any new transfer on the slave path of the data path (CPU access to MR_SUBG APB registers)
- Once the AHBUPCONV block has hold its AHB line and has the confirmation from the AHBDOWNCONV the other AHB2APB data path is also safely held, it allows the RCC block to change the system clock frequency to the requested one.
- When the new system clock frequency is stable, the RCC informs the AHBUPCONV it is done. Then:
  - the AHBUPCONV release the AHB transfers
  - the AHBUPCONV informs the AHBDOWNCONV is can also releases the AHB/APB transfers.

4 I/O operating modes

STM32WL30xx/31xx/33xx devices propose up to 32 programmable I/Os (depending on the package).

Each I/O can be programmed in several modes:

- input mode
- output mode
- Alternate function
- Analog mode (when available)

STM32WL30xx/31xx/33xx devices support eight alternate modes called AFx (x = 0 to 7). The configuration of each I/O for those alternate modes is detailed in *Table 8*, *Table 9* and *Table 10*. Some additional functions are also available see *Table 11*.

Note: The I/Os features presented in Table 8, Table 9 are valid only when the associated I/O is programmed as alternate function.

Refer to Section 7: General-purpose I/Os (GPIO) for more details on all configurations.

The number of I/Os available in the STM32WL30xx/31xx/33xx device depends on the package:

- 32 I/Os in QFN48
- 17 I/Os in QFN32

Type acronyms correspond to:

- I: Input
- O: Output
- I/O: Input Output
- OD: Open Drain

Table 8. GPIO alternate options AF0 - AF3(1)

Pin name	AF0 mode		AF1 mode		AF2 mode		AF3 mode	
	Type	Signal	Type	Signal	Type	Signal	Type	Signal
Port A								
PA0(2)	I/OD	I2C1_SCL	I	USART_CTS	-	-	-	-
PA1(2)	I/OD	I2C1_SDA	I/O	USART_TX	I/O	TIM16_BRK	-	-
PA2	I/O	SWDIO	I/O	USART_CK	I/O	TIM16_CH1	O	I2S3_MCK
PA3	I	SWCLK	O	USART_RTS_DE	O	TIM16_CH1N	I/O	SPI3_SCK/ I2S3_CK
PA4	O	LCO	I/O	LPUART_TX	O	COMP1_OUT	-	-
PA5	O	MCO	I/O	LPUART_RX	-	-	-	-
PA6	I/OD	I2C2_SCL	I	USART_CTS	-	-	-	-
PA7	I/OD	I2C2_SDA	O	LPUART_RTS_DE	-	-	-	-

Table 8. GPIO alternate options AF0 - AF3(1) (continued)

Pin name	AF0 mode		AF1 mode		AF2 mode		AF3 mode	
	Type	Signal	Type	Signal	Type	Signal	Type	Signal
PA8	I/O	RTC_OUT /RTC_TAMP1 /RTC_TS	I/O	USART_RX	O	RX_SEQUENCE	I/O	SPI3_MISO
PA9	-	-	I/O	USART_TX	O	RTC_OUT	I/O	SPI3_NSS/ I2S3_WS
PA10	-	-	I	LPUART_CTS	O	TX_SEQUENCE	O	I2S3_MCK
PA11	O	MCO	-	-	O	RX_SEQUENCE	I/O	SPI3_MOSI/ I2S3_SD
PA12	I/O	I2C1_SMBA	I/O	SWDIO	-	-	I/O	SPI1_NSS
PA13	I/OD	I2C2_SCL	I	SWCLK	-	-	I/O	SPI1_SCK
PA14	I/OD	I2C2_SDA	-	-	-	-	I/O	SPI1_MISO
PA15	I/O	I2C2_SMBA	I/O	USART_RX	-	-	I/O	SPI1_MOSI
Port B								
PB0 (3)	I/O	USART_RX	O	LPUART_RTS_DE	I/O	TIM16_CH1	I/O	SPI1_NSS
PB1	I/O	USART_CK	I/O	SWDIO	O	TIM16_CH1N	I/O	SPI1_SCK
PB2	O	USART_RTS_DE	I	SWCLK	I/O	TIM16_BRK	I/O	SPI1_MISO
PB3	I	USART_CTS	I/O	LPUART_TX	-	-	I/O	SPI1_MOSI
PB4	-	-	I/O	LPUART_TX	-	-	I/O	SPI3_MISO
PB5	-	-	I/O	LPUART_RX	-	-	I/O	SPI3_NSS/ I2S3_WS
PB6(2)	I/OD	I2C1_SCL	I/O	LPUART_TX	O	COMP1_OUT	I/O	SPI3_SCK/ I2S3_CK
PB7(2)	I/OD	I2C1_SDA	I/O	LPUART_RX	O	RF_ACTIVITY	I/O	SPI3_MOSI/ I2S3_SD
PB8	I/O	USART_CK	I/O	LPUART_RX	-	-	I/O	SPI1_MISO
PB9	I/O	USART_TX	I	LPUART_CTS	-	-	I/O	SPI1_MOSI
PB10	I/OD	I2C1_SDA	-	-	-	-	I/O	SPI1_NSS
PB11	I/OD	I2C1_SCL	O	USART_RTS_DE	O	LC_ACTIVITY	I/O	SPI1_SCK
PB12(3)	O	USART_RTS_DE	I	LPUART_CTS	O	LCO	-	-
PB13(4)	I/O	I2C2_SMBA	-	-	-	-	-	-
PB14	I/O	I2C1_SMBA	I/O	USART_RX	O	TX_SEQUENCE	O	MCO
PB15	-	-	I/O	USART_TX	O	COMP1_OUT	-	-

1. The green cells correspond to I/Os available on the VFQFN32 package
   The blue cells correspond to unbonded pads on the VFQFN32 package
   STM32WL30xx devices are only available in the VFQFPN32 package.
2. This I/O is able to be configured in open-drain.
3. This I/O is shared with OSC32_OUT (low speed clock oscillator pin).
4. This I/O is shared with OSC32_IN (low speed clock oscillator pin).

Note: 1 Comparator, DAC, I2C2, LCSC, LCD, LPAWUR are available only on STM32WL33xx devices.

2 ADC, temperature sensor, I2C1, SPI1, I2S are available only on STM32WL31xx/33xx devices.

Table 9. GPIO alternate options AF4 - AF6(1)

Pin name	AF4 mode		AF5 mode		AF6 mode	
	Type	Signal	Type	Signal	Type	Signal
Port A						
PA0(2)	I/O	TIM2_CH3	-	-	O	LCD_SEG12 / LCD_COM4
PA1(2)	I/O	TIM2_CH4	-	-	O	LCD_SEG0
PA2	I/O	TIM2_CH1	I/O	SWDIO	O	LCD_SEG1
PA3	I/O	TIM2_CH2	I	SWCLK	O	LCD_SEG2
PA4	I/O	TIM2_CH1	-	-	-	-
PA5	I/O	TIM2_CH2	-	-	O	LCD_SEG9
PA6	I/O	TIM2_CH1	-	-	-	-
PA7	I/O	TIM2_CH2	-	-	-	-
PA8	I/O	TIM2_CH3	-	-	O	LCD_COM0
PA9	I/O	TIM2_CH4	-	-	O	LCD_SEG3
PA10	I	SUBG_TX_DATA	-	-	O	LCD_SEG4
PA11	O	SUBG_TX_CLOCK	-	-	O	LCD_SEG5
PA12	I/O	TIM2_CH1	-	-	-	-
PA13	I	TIM2_ETR	-	-	O	LCD_SEG10
PA14	-	-	-	-	O	LCD_SEG11
PA15	-	-	-	-	-	-

Table 9. GPIO alternate options AF4 - AF6(1) (continued)

Pin name	AF4 mode		AF5 mode		AF6 mode	
	Type	Signal	Type	Signal	Type	Signal
Port B						
PB0	O	ANTENNA_SWITCH	-	-	O	LCD_COM1
PB1	O	SUBG_RX_DATA	-	-	O	LCD_SEG6
PB2	O	SUBG_RX_CLOCK	-	-	O	LCD_SEG7
PB3	I/O	TIM2_CH4	-	-	O	LCD_SEG8
PB4	-	-	-	-	O	LCD_SEG13 / LCD_COM5
PB5	-	-	-	-	O	LCD_SEG14 / LCD_COM6
PB6 (2)	I/O	TIM2_CH3	-	-	O	LCD_SEG15 / LCD_COM7
PB7 (2)	I	TIM2_ETR	-	-	O	LCD_COM2
PB8	I/O	TIM2_CH4	-	-	-	-
PB9	-	-	-	-	-	-
PB10	I/O	TIM2_CH2	-	-	-	-
PB11	I/O	TIM2_CH1	-	-	-	-
PB12(3)	I/O	TIM2_CH3	-	-	-	-
PB13(4)	I/O	TIM2_CH4	-	-	-	-
PB14	I	TIM2_ETR	-	-	O	LCD_COM3
PB15	-	-	-	-	I/O	LCD_VLCD

1. The green cells correspond to I/Os available on the VFQFN32 package
   The blue cells correspond to not bonded pads onthe VFQFN32 package
   STM32WL30xx devices are only available in the VFQFPN32 package.
2. This I/O is able to be configured in open-drain.
3. This I/O is shared with OSC32_OUT (low speed clock oscillator pin).
4. This I/O is shared with OSC32_IN (low speed clock oscillator pin).

Note: 1 Comparator, DAC, I2C2, LCSC, LCD, LPAWUR are available only on STM32WL33xx devices.

2 ADC, temperature sensor, I2C1, SPI1, I2S are available only on STM32WL31xx/33xx devices.

Concerning the ADC block present in STM32WL31xx/33xx devices.

- The 8 ADC channels are available on PA2, PA3, PB0, PB1, PB2, PB3, PB4, PB5.

Concerning the comparator present in the STM32WL33xx device:

- The Comp minus channel is available on PA13, PB0, PB3
- The Comp plus channel is available on PA14, PB1, PB2

Note: The ADC/Comp features on those pins are available if the associated pin is configured in analog mode (see Section 7: General-purpose I/Os (GPIO) for more details).

Table 10 shows the mapping associated to analog configuration (for pins able to support analog mode).

Table 10. I/O Analog feature mapping(1) (2)

Pin name	Analog feature	Parallel Analog feature	Pin name	Analog feature	Parallel Analog feature
Port A			Port B		
PA0	N/A	-	**PB0**	ADC_VINM1	COMP1_INN1
PA1	N/A	-	**PB1**	ADC_VINP1	COMP1_INP/LCSCOUT1
PA2	ADC_VINM2	-	**PB2**	ADC_VINM0	COMP1_INP/LCSCOUT3
PA3	ADC_VINP2	-	**PB3**	ADC_VINP0	COMP1_INN2
PA4	N/A	-	**PB4**	ADC_VINM3	VCMBUFF
PA5	N/A	-	**PB5**	ADC_VINP3	N/A
PA6	N/A	-	**PB6**	N/A	N/A
PA7	N/A	-	**PB7**	N/A	N/A
PA8	N/A	-	**PB8**	N/A	N/A
PA9	N/A	-	**PB9**	N/A	N/A
PA10	N/A	-	**PB10**	N/A	N/A
PA11	N/A	-	**PB11**	N/A	
PA12	N/A	-	**PB12**	-	SXTALO (3)
PA13	COMP1_INN0	DACOUT_GPIO	**PB13**	-	SXTALI (3)
PA14	COMP1_INP	LCSCOUT2	**PB14**	-	PVD VIN(4)
PA15	N/A	-	**PB15**	-	N/A

1. N/A means Not Applicable as the associated I/O does not support analog option.
2. The green cells correspond to I/Os available on VFQFN32 package
The blue cells correspond to not bonded pads on VFQFN32 package
STM32WL30xx devices are only available in the VFQFPN32 package.
3. The parallel analog feature is obtained by setting the RCC_CR.LSEON bit in the RCC registers. Then the PB12 and PB13 are forced by hardware to manage the LSE through SXTALO/SXTALI whatever the selected mode in the associated GPIO_MODERx register, but if APC is set is necessary to disable PUB12/PUB13/PDB12/PDB13 in PWRC registers.
4. The selection between ATB1 and PVD is done through a register in the SYSCFG block (see *8.2.11: I/O analog switch control register (GPIO_SWA_CTRL)* for details).

Table 11. I/O Additional function mapping(1)

Pin name	Function	Pin name	Function
Port A		Port B	
PA0	-	PB0	-
PA1	-	PB1	LCSCOUT1
PA2	-	PB2	LCSCOUT3
PA3	-	PB3	-
PA4	-	PB4	-
PA5	-	PB5	-
PA6	-	PB6	-
PA7	-	PB7	-
PA8	-	PB8	-
PA9	-	PB9	-
PA10	LCO	PB10	-
PA11	-	PB11	-
PA12	-	PB12	SXTALO(2)
PA13	-	PB13	SXTALI(2)
PA14	LCSCOUT2	PB14	-
PA15	-	PB15	-

1. The green cells correspond to I/Os available on VFQFN32 package
The blue cells correspond to not bonded pads on VFQFN32 package
STM32WL30xx devices are only available in the VFQFPN32 package.

2. The parallel analog feature is obtained by setting the RCC_CR.LSEON bit in the RCC registers. Then the PB12 and PB13 are forced by hardware to manage the LSE through SXTALO/SXTALI whatever the selected mode in the associated GPIO_MODERx register, but if APC is set is necessary to disable PUB12/PUB13/PDB12/PDB13 in PWRC registers.

5 Power controller (PWRC)

The power controller block controls the analog supplies block and manages the startup, active and low power phase of the device including the transition from one state to another.

5.1 Features

The power controller block supports the following features:

- low power mode choice and entry/exit sequences
- the flash memory power (ON/OFF) and the power down sequence
- RAM banks retention control
- power monitoring:
  - POR/PDR reset on rising/falling VDDIO voltage
  - Programmable voltage detector (PVD) monitoring of the VDDIO with programmable threshold or of an external analog input voltage (compared to the internal VBGP).
- I/Os pull-up/-down during low power mode
- Wakeup I/O configuration
- SMPS bypass on-the-fly (BOF)

5.2 Power supply domains

STM32WL30xx/31xx/33xx devices embed three power domains:

- VDD33 (also termed VDDIO or VDD):
  - the voltage range is between 1.7 V and 3.6 V,
  - it supplies a part of the I/O ring, the embedded regulators and the system analog IPs as power management block and embedded oscillators,
- VDD12o:
  - always-on digital power domain,
  - this domain is generally supplied at 1.2 V during active phase of the device
  - this domain is supplied at either 1.0 V or 1.2 V during low power mode (Deepstop). Refer to *Section 5.4.2: Deepstop mode* for details
- VDD12i:
  - interruptible digital power domain,
  - this domain is generally supplied at 1.2 V during active phase of the device
  - this domain is shut down during low power mode (Deepstop)

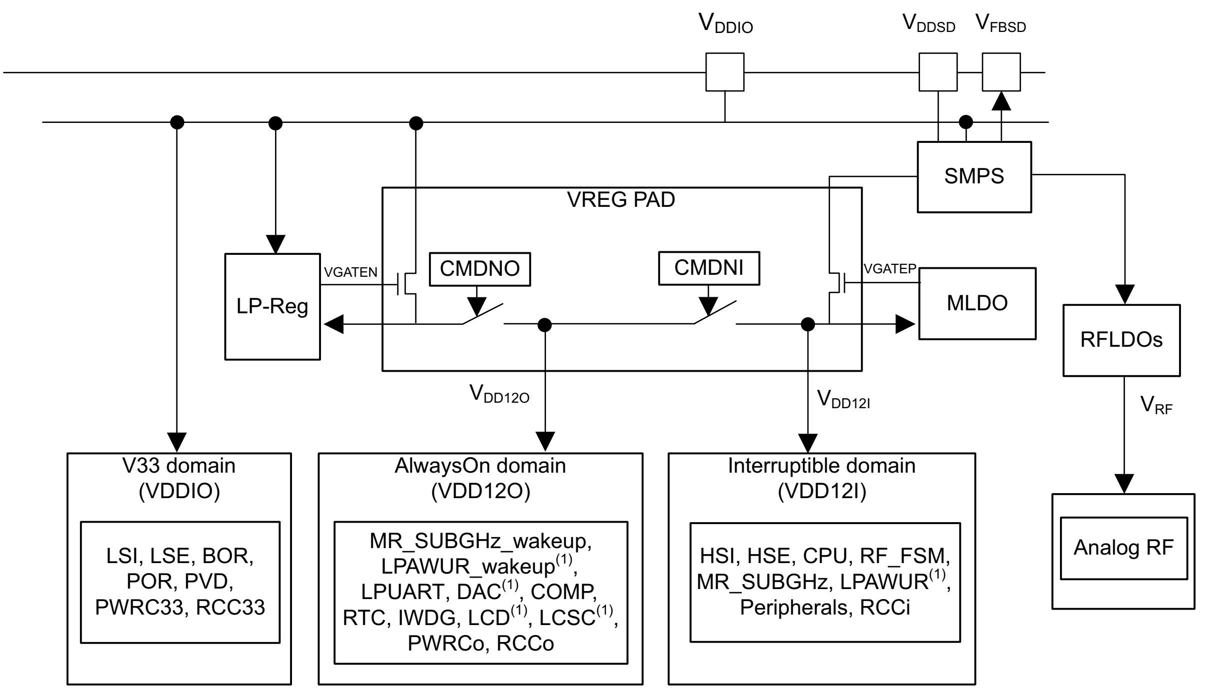
The digital power supplied are provided by different regulators:

- a main LDO (MLDO):
  - providing the 1.2 V from a 1.4-3.3 V input voltage,
  - supplies both VDD12i and VDD12o when the device is active,
  - is disabled during the low power mode (Deepstop).
- a low power LDO (LPREG):
  - stays enabled during both active and low power phases,
  - providing a 1.0 V or 1.2 V voltage,
  - is not connected to the digital domain when the device is active,
  - is connected to the VDD12o domain during low power mode (Deepstop).
- a dedicated LDO (RFLDO) to provide a 1.2 V to the analog RF block.

An embedded SMPS step down converter is available (inserted between the external power and the LDOs), but can be bypassed statically or dynamically (see *Control register 5 (PWRC_CR5)*).

Figure 4 shows an overview of the different regulators and connections between the power suppliy domains.

Figure 4. Power supply domain overview



1. STM32WL33xx only

MSv58344V1

5.3 Power voltage supervisor

STM32WL30xx/31xx/33xx devices embed several power voltage monitoring:

- Power On reset (POR) / Power Down reset (PDR) / Brown-Out reset (BOR)
- BORH monitoring
- Power voltage Detector (PVD)

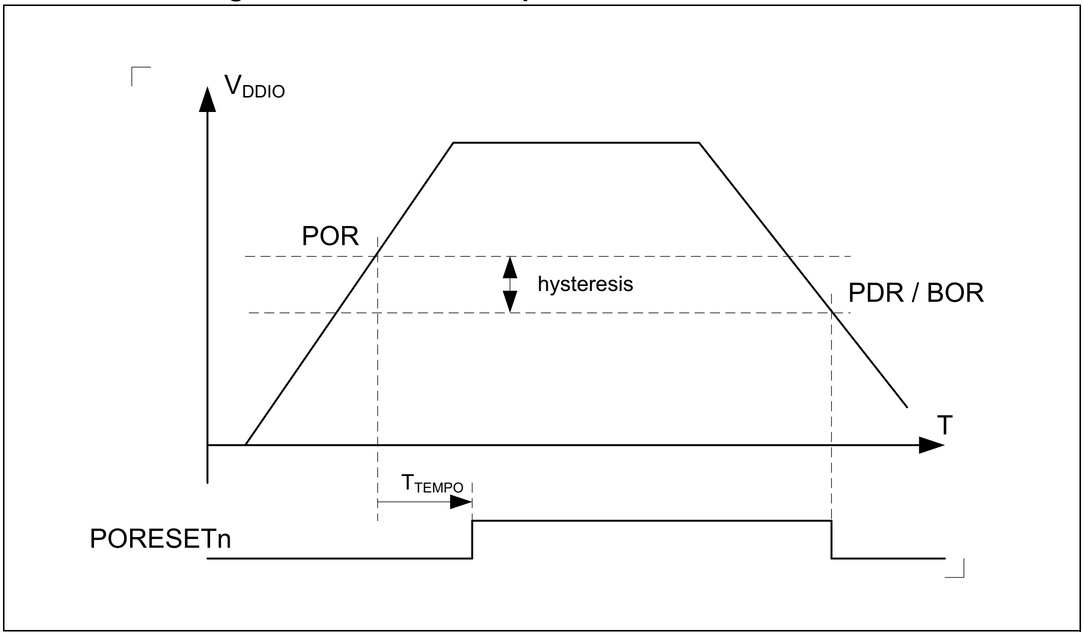
5.3.1 Power On reset POR / Power Down reset (PDR) / Brown-Out reset (BOR)

The device has an integrated power on reset / power down reset, coupled with a brown-out reset circuitry.

During power on, the device remains in reset mode as long as VDDIO is below a V_{POR} threshold (typically 1.65V).

During power down, the PDR puts the device under reset when the supply voltage (VDD) drops below the V_{PDR} threshold (around 20mV below V_{POR}). The PDR feature is always enabled.

Figure 5. Power on reset/power down reset waveform



With typical values as follow:

- V_{POR}: 1.65 V
- hysteresis: 20 mV (so V_{PDR}: 1.63 V)
- T_{TEMPO}: 250 us

The Brown-Out reset (BOR) generates a device reset when the power supply (VDD) drops under V_{PDR}.

This feature is always active except during the Shutdown mode where the software can decide to enable it or not (through PWRC_CR1.ENSDNBOR bit).

5.3.2 BORH

The BORH is a VDDIO monitoring that raised an associated signal when the monitored voltage drops under a specified threshold.

However, the goal of this device feature is to monitor the VDDIO versus 2.0 V. Indeed, when the VDDIO goes below 2.0 V, the SMPS should be stopped.

It is recommended to use the PVD feature at application level to monitor the VDDIO with freedom to program dynamic threshold without any impact nor relationship with the hardware.

This feature can be enabled/disabled through PWRC_CR1.ENBORH bit.

This feature is disabled during Shutdown mode.

When the feature is enabled and the BORH detect a VDDIO below the threshold , a status flag is raised in the SYSCFG block that can generate an interrupt to the CPU if unmasked (see *I/O interrupt status and clear register (IO_ISCR)*).

Caution: The BORH is associated to a flash write protection: if a programming is on-going, the flash controller stops it before the end but through a proper sequence to avoid any flash corruption.

5.3.3 Power voltage detection (PVD)

The PVD can be used to monitor:

- the VDDIO:
  - the VDDIO is compared to a programmed threshold (between 2.05V and 2.91 V),
  - the threshold programming is done through PWRC_CR2.PVDLS[2:0] bit field,

The PVD can be enabled or disabled through PWRC_CR2.PVDE bit.

When the feature is enabled and the PVD measure a voltage below the comparator, a status flag is raised in the SYSCFG block that can generate an interrupt to the CPU if unmasked (see *I/O interrupt status and clear register (IO_ISCR)*).

5.4 Operating modes

STM32WL30xx/31xx/33xx devices support 3 main operating modes:

- Run mode
- Deepstop mode

Shutdown modeThe transition from one mode to another is managed through a PMU state machine.

5.4.1 Run mode

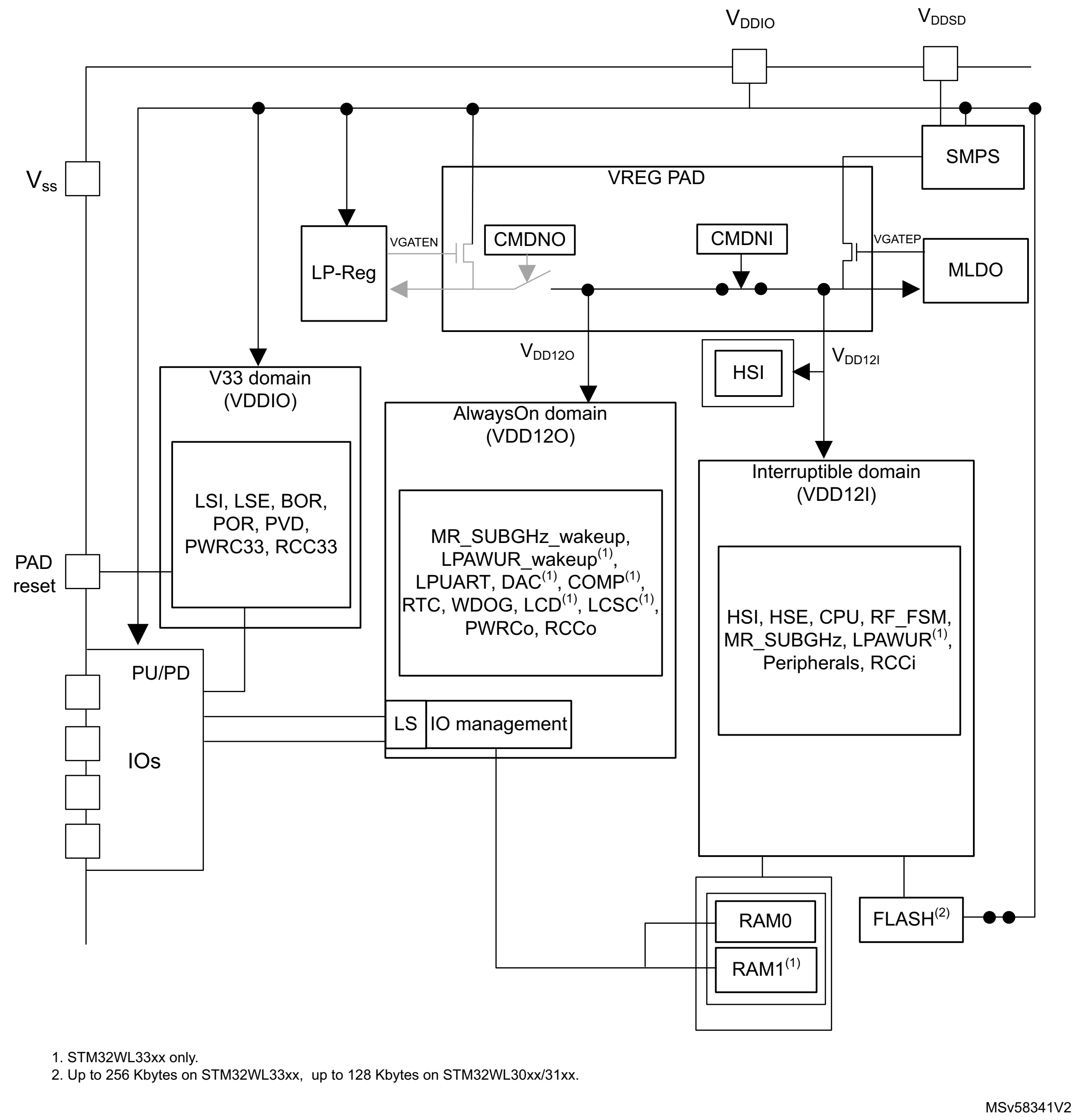
In Run mode:

- both regulators (MLDO and LPREG) are enabled,
- the MLDO provides the power supply for both VDD12i and VDD12o,
- the system clock and bus clock are running,
- the CPU and the radio can be used.

The power consumption may be reduced by gating the clock of the unused peripherals through the RCC clock enable registers (see *Section 6.7: RCC register map*).

The *Figure 6* shows the regulators and SMPS configuration in Run mode with a product with 32 Kbytes of RAM.

Figure 6. Power regulators and SMPS configuration in Run mode



1. STM32WL33xx only.
2. Up to 256 Kbytes on STM32WL33xx, up to 128 Kbytes on STM32WL30xx/31xx.

5.4.2 Deepstop mode

Deepstop is the only low-power mode of STM32WL30xx/31xx/33xx devices, allowing to restart from a saved context environment and go on running the application at wakeup.

The conditions to enter the Deepstop mode are:

- The radio (MR_SUBGHz) is sleeping
- The CPU is sleeping (WFI with SLEEPDEEP information active)
- No unmasked wakeup sources is active (including those from a previous wakeup sequence for which the software did not clear the associated flag after wakeup) the PWRC_CR1.LPMS bit is equal to 0.
- Set PWRC_CR2.GPIORET bit (see *Control register 2 (PWRC_CR2)*) to enable GPIOs configuration retention.
- If SMPS clock variable rate multiplier is enabled RCC_KRMR.KRMEN=1 (see *SMPS clock variable rate multiplier register (RCC_KRMR)*), in order to guarantee a good SMPS startup at next wakeup, its mandatory to put RCC_CFGR.SMPSDIV=0.

In Deepstop mode:

- The system and bus clocks are stopped as the RC64MPLL block is OFF
- The VDD12i power domain is switched off
- The VDDI2o power domain is ON and supplied by the LPREG which regulated voltage is:
  - 1.2 V if the LCD[a] is enabled (LCD_CR.LCDEN=1)
  - 1.2 V if the comparator[a] scaler is enable (COMP_CSR.SCALEEN=1 and COMP_CSR.EN=1)
  - 1.2 V if the bit PWRC_CR2.LPREG_FORCE_VH=1
  - 1.0 V in all the other cases

The current regulation status of the LPREG is reported by the PWRC_CR2.LPREG_VH_STATUS bit

- The RAM0 bank is kept in retention
- The other RAM banks are in retention or not, depending on software choice in PWRC_CR2 register
- The slow clock can be running or stopped, depending on the software configuration present before Deepstop entry:
  - ON or OFF
  - LSE or LSI source
- The comparator[a], DAC[a], LCSC[a], LCD[a], RTC, IWDG and LPUART stay active (if enabled and one slow clock source is ON)
- The MR_SUBG wakeup block including its timer stay active (if enabled and one slow clock source is ON)
- The LPAWUR[a] block including its timer stay active (if enabled and one slow clock source is ON)
- The configurations of all the I/Os are latched before entering Deepstop mode:
  - AF configuration is latched only for the I/Os on which is mapped at least one pin of a peripheral that can be active in Deepstop mode (comparator[a], DAC[a], LCSC[a], LCD[a], RTC, IWDG and LPUART)
  - I/O analog switch configuration is retained for the I/Os on which is mapped at least

one analog pin of a peripheral that can be active in Deepstop mode (comparator[a], DAC[a], LCSC[a])
  - All the I/Os able to drive either a static low or high output level, some of which also carry the slow clock information LCO or RTC_OUT.
- The DAC[a] is only internally connected to the COMP[a], so the output on PA13 cannot be used
- VCMBUF is connected to PB4 only in the LCSC[a] use case.
  A version of the Deepstop mode called DEEPSTOP2 has been implemented to emulate the Deepstop mode without losing the debugger connection and breakpoints nor watchpoints.
- This variant can be selected by setting the PWRC_DBGR.DEEPSTOP2 bit.
- In this case, the Deepstop mode sequence (entry and exit) is done without shutting down the VDD12i power domain.

Possible wakeup sources:

- the MR_SUBG block is able to generate two events to wakeup the system through its embedded wakeup timer running on slow clock:
  - SUBG RFIP wakeup time is reached,
- The LPAWUR[a] is able to generate a wakeup event
- The LCD[a] is able to generate a wakeup event
- The COMP[a] is able to generate a wakeup event (with polarity selection, like I/Os)
- The LCSC[a] is able to generate a wakeup event
- The RTC is able to generate a wakeup event
- The LPUART is able to generate a wakeup event
- The IWDG is able to generate a reset event
- All I/Os are able to wakeup the system

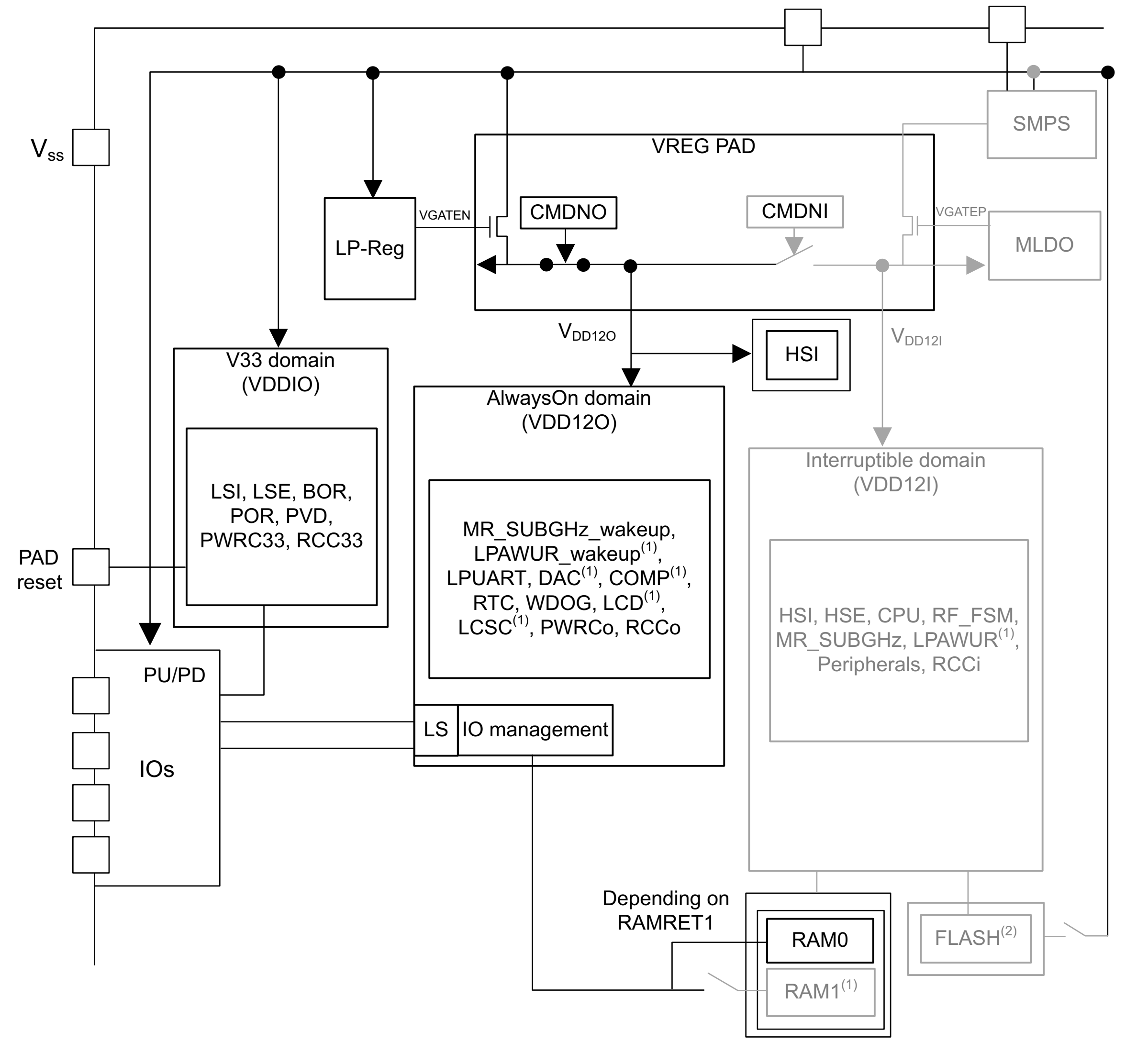
After wakeup from Deepstop, all the I/Os (except PA2 and PA3) are in retention mode; if DBGRET bit is set (in this case also PA2 and PA3 are in retention mode), retaining the configuration they had before entering Deepstop. To change their configuration, when exiting Deepstop, it is necessary to reset GPIORET bit after having re-configured GPIO registers through *Section 7.4: GPIO registers* (this GPIO configuration, can be the same before entering DEESPTOP, or a new one). If DBGRET bit was reset before entering Deepstop, PA2 and PA3 take their GPIO reset configuration after wakeup from Deepstop, until a new GPIO configuration is set.

At wakeup, the hardware resources located in the VDD12i power domain are reset, the CPU reboots. The wakeup reason is visible in a PWRC register (see *Internal source wakeup status register (PWRC_IWUF)*, *Port A wakeup status register (PWRC_WUFA)* and *Port B wakeup status register (PWRC_WUFB)* for details).

Figure 7 shows the regulators and SMPS configuration in Deepstop mode, with a configuration requesting retention on SRAM0 and SRAM1 (only in STM32WL33xx) banks.

a. STM32WL33xx only.

Figure 7. Power regulators and SMPS configuration in Deepstop mode



1. STM32WL33xx only.
2. Up to 256 Kbytes on STM32WL33xx, up to 128 Kbytes on STM32WL30xx/31xx.

MSv58345V2

5.4.3 Shutdown mode

The Shutdown mode is the least power consuming mode.

The conditions to enter the Shutdown mode are the same conditions needed to enter in Deepstop mode except that the PWRC_CR1.LPMS bit must be equal to 1 (PWRC_DBGR.DEEPSTOP2 bit must be maintained equal to 0).In Shutdown mode:

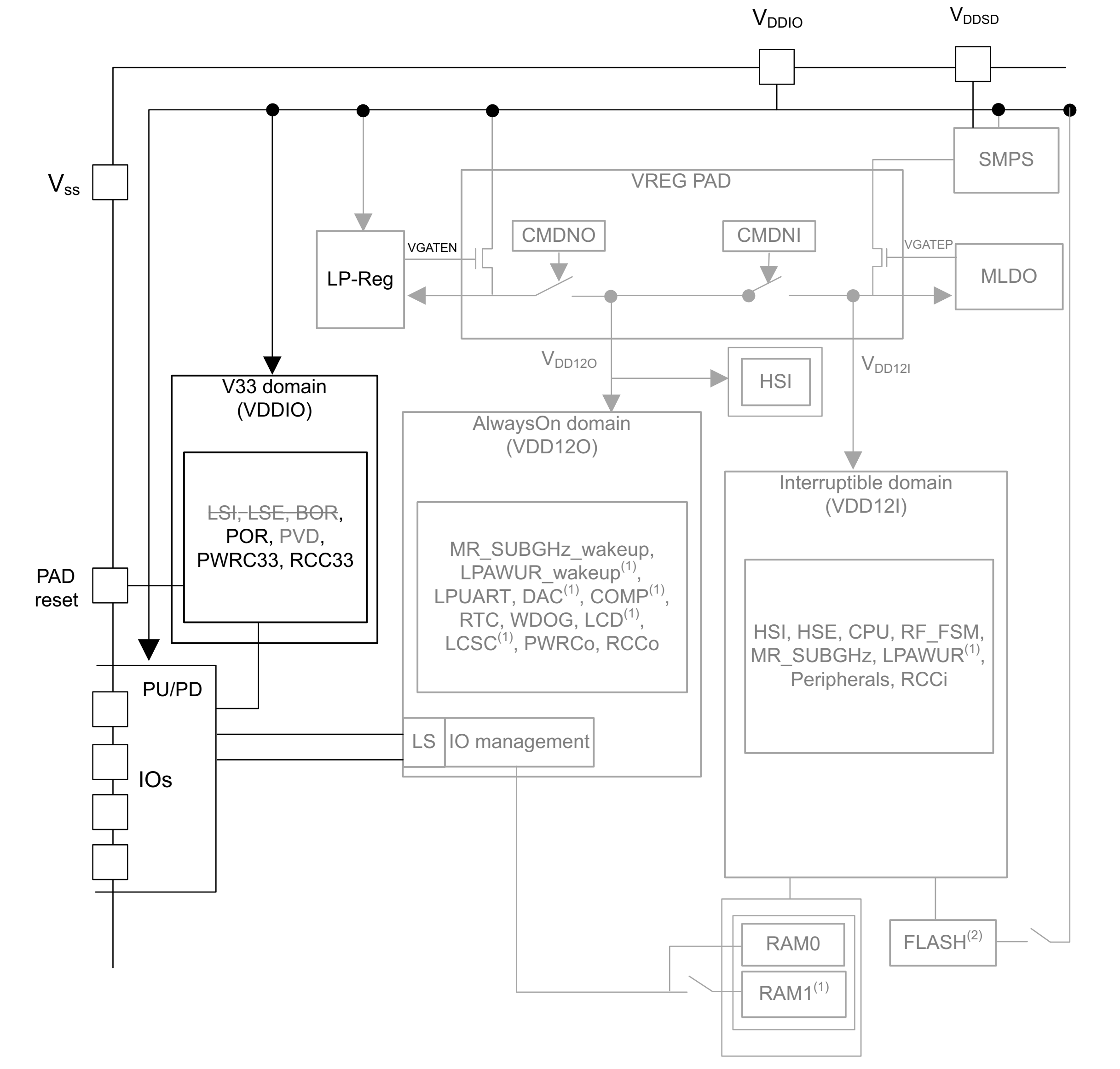
- the system is powered down as both regulators are OFF (so both VDD12i and VDD12o power domains are OFF),
- only the VDDIO power domain is ON,
- All clocks are OFF (system and slow clock tree) as RC64MPLL, LSI and LSE are OFF,
- if PWRC_CR1.APC = 1, the I/Os pull-up/-down are controlled by the PWRC_PUCRx/PWRC_PDCRx during the Shutdown mode.
- the only wakeup sources are a low pulse on the RSTN pin or configurable pulse on PB0 pin.
- the only wakeup sources are a low pulse on the RSTN pin or configurable pulse or level on PB0 pin through the *Shutdown I/O wakeup enable register (PWRC_SDWN_WUEN)* and *Shutdown I/O wakeup polarity register (PWRC_SDWN_WUPOL)*

A Shutdown exit is similar to a POR startup of the board. The associated reset reason is the PORRSTF flag or EWUF flag (see *V33 reset status register (RCC_CSR)* for reset reason flags detail or *Shutdown I/O Wakeup flag register (PWRC_SDWN_WUF)* for PB0 wakeup flag).

The BOR feature may be enabled or disabled during Shutdown through the PWRC_CR1.ENSDNBOR bit).

The *Figure 8* shows the regulators and SMPS configuration in Shutdown mode, configured with the BOR reset disabled.

Figure 8. Power regulators and SMPS configuration in Shutdown mode



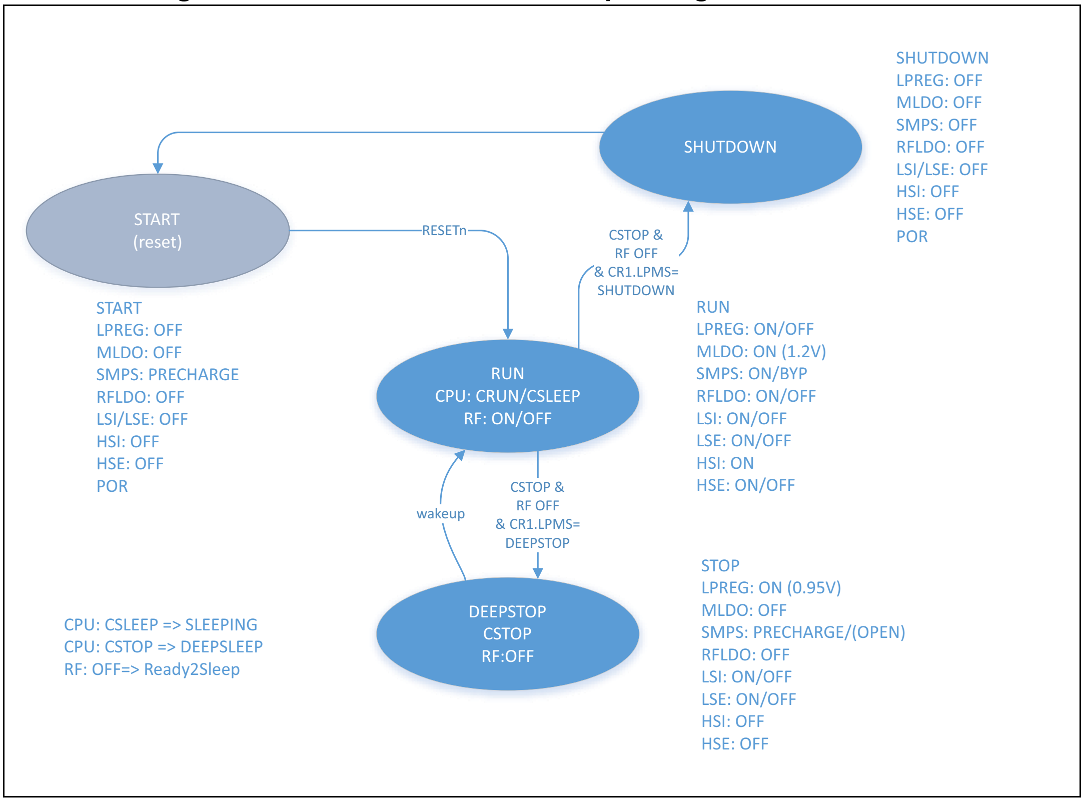
1. STM32WL33xx only.
2. Up to 256 Kbytes on STM32WL33xx, up to 128 Kbytes on STM32WL30xx/31xx.

MSv58346V2

5.4.4 Operating modes transition management

The PWRC block manages the switches from an operating mode to another through a state machine.

Figure 9. PWRC state machine for operating mode transitions



5.5 SMPS step down regulator

The STM32WL30xx/31xx/33xx SMPS is a 120 mA output step down SMPS (switch mode power supply)

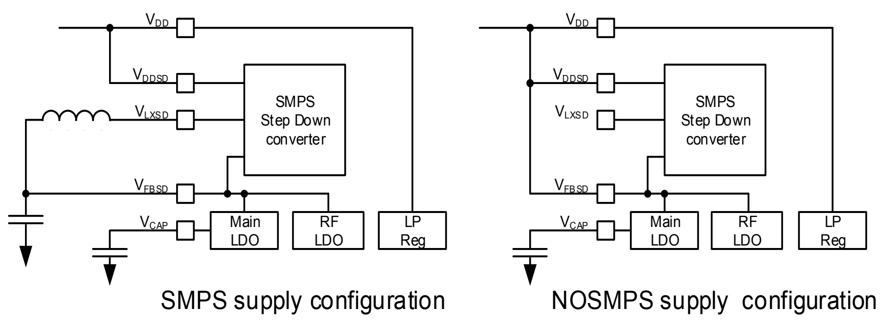
The SMPS output voltage can be programmed from 1.2 V to 2.4 V. Its internal clock frequency is 4 MHz or 8 MHz when RCC_KRMR.KRMEN=0 (see *SMPS clock variable rate multiplier register (RCC_KRMR)*) or a programmable frequency clock SMPS_CLK_KRM when RCC_KRMR.KRMEN=1. In this latter case the SMPS can be clocked at System Clock divided by 8 to 16 with unitary steps (CLK_SMPS_KRM). This feature can be useful in case the channel to be received is at a frequency that is an integer multiple of the SMPS clock.

The SMPS can be in different configurations:

- ON:
  - the V_{FBSD} pin of the SMPS outputs a regulated voltage (from 1.2 V to 2.4 V),
  - the SMPS needs a clock.
- OFF:
  - the V_{FBSD} pin has to be forced externally with VDDIO,
  - the SMPS does not need a clock.
- STATIC BYPASS ON THE FLY
  - the V_{FBSD} pin is internally connected to VDDSD via a switch, with a maximum current of 40mA.
  - the SMPS doesn't need a clock and is disabled.
- DYNAMIC BYPASS ON THE FLY
  - the V_{FBSD} pin outputs a programmable regulated voltage, with a maximum current of 40mA.
  - the SMPS doesn't need a clock and is disabled.
- OPEN:
  - the V_{FBSD} pin is floating,
  - the SMPS does not need a clock.

Except for the configuration SMPS OFF, a L/C BOM must be present on the board and connected to the V_{FBSD} pad (see *Figure 10*).

Figure 10. Power supply configuration



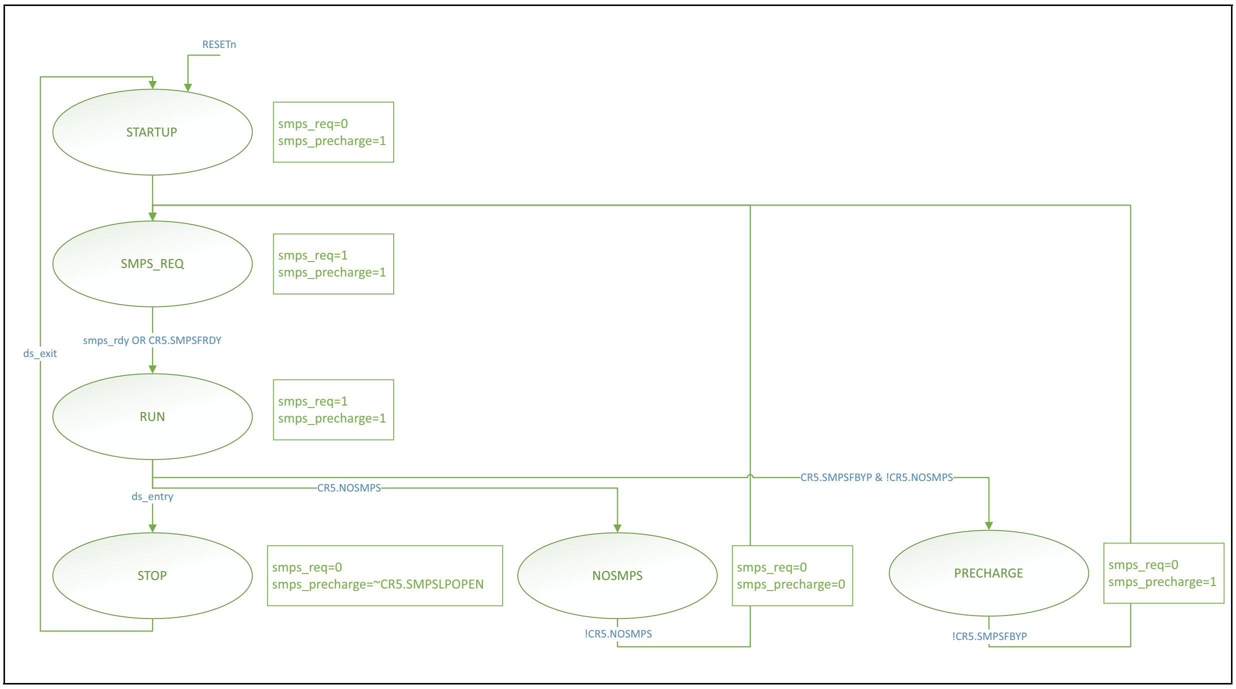
The user must configure the field PWRC_CR5.SMPSBOMSEL[1:0] according to the BOM implemented on his board. The value to program is indicated in *Table 12*.

Table 12. SMPS BOM information

BOM	Inductor (L)	Output capacitor (C)	SMPSBOMSEL[1:0]
BOM1	1.5 µH	2.2 µF	00
BOM2	2.2 µH	4.7 µF	01
BOM3	10 µH	4.7 µF	10

The SMPS is managed by the PWRC through a state machine shown in *Figure 11*.

Figure 11. PWRC SMPS state machine overview



After a Power On reset sequence, the SMPS FSM always goes up to RUN state.

From there, the SMPS FSM can stay in three states (others are transition states):

- Run:
  - the SMPS is ON in a Run mode,
  - the SMPS clock is running,
  - the V_{FBSD} is regulated and voltage amplitude is the one programmed in the PWRC_CR5.SMPSLVL bit field,
  - an L/C BOM is present on the board and is connected to the VFBSD pad of the STM32WL30xx/31xx/33xx (see *Figure 10*).
- NOSMPS (if the software configures PWRC_CR5.NOSMPS=1 or PWRC_CR5.NOSMPS_BOF=1):
  - the SMPS is OFF,
  - the SMPS clock is stopped,
  - the V_{FBSD} is directly connected to the VDDSD through the VFBSD pad of the STM32WL30xx/31xx/33xx (see *Figure 10*),
  - the PWRC does not control any specific sequencing on the SMPS during low power entry/exit phases.

When the device enters in Deepstop mode, the PWRC automatically switches the SMPS from Run to STOP mode, which can be:

- if PWRC_CR5.SMPSLPOPEN = 0: SMPS output is the VDDSD
- if PWRC_CR5.SMPSLPOPEN = 1: the SMPS output is floating.

5.5.1 SMPS bypass on-the-fly (BOF)

Bypass on-the-fly (BOF) is a feature that permits to bypass the SMPS. This can be done directly with a power switch (**static bypass mode**) or via an LDO (**dynamic bypass mode**).

In case radio extra sensitivity is needed, the user can switch on the fly on LDO (dynamic bypass mode) before entering radio receiver mode. In this way the SMPS is OFF.

This can be handled by software through an interrupt handler on RX request. The interrupt that may control is MR_SUBG TX/RX Sequence interrupt.

When BOF is done in static bypass mode, the SMPS is disabled and the SMPS output is connected to the battery via an internal switch. In this case both Deepstop and Run mode operations can be chosen.

The static mode connects the SMPS output to the battery after the first start-up of STM32WL30xx/31xx/33xx devices and the connection is done until a reset occurs.

When BOF is done in dynamic bypass mode, the SMPS is disabled and the LDO is enabled. The LDO is connected between the battery and the SMPS output. LDO voltage is programmable through register PWRC_BOF_TUNE.BOF_TUNE. It is recommended to use the same output voltage than the one chosen for the SMPS through PWRC_CR5.SMPSLVL register. If it’s not the case, only BOF LDO output voltage less than SMPS output voltage must be chosen. In this case only Run mode operation is possible.

A current limitation is implemented in both static and dynamic bypass modes. It is set via PWRC_DBGSMPS.BOF_CUR_SEL bits. Always use a bigger current limitation than the current expected to flow through the BOF. For example, for +14dBm transmission, a current consumption from the battery of 40mA is expected. In this case 60mA current limitation has to be set.

5.5.1.1 How to program a static bypass mode:

1. Program PWRC_DBGSMPS.BOF_CUR_SEL
2. Program PWRC_CR5.SMPS_BOF_STATIC=1
3. ProGRAM PWRC_CR5.NOSMPS_BOF=1

5.5.1.2 Program a dynamic bypass mode:

1. Program PWRC_DBGSMPS.BOF_CUR_SEL, PWRC_CR5.SMPSLVL if not already done in the sequence, PWRC_BOF_TUNE.BOF_TUNE
2. Program PWRC_CR5.SMPS_BOF_DYN=1
3. Program PWRC_CR5.NOSMPS_BOF=1

5.6 PWRC register descriptions

Note: 1 *All the PWRC APB registers are only 16-bit registers. The 16 MSBs are always stuck at 0.*

2 *Comparator, DAC, I2C2, LCSC, LCD, LPAWUR are available only on STM32WL33xx devices.*
ADC, temperature sensor, I2C1, SPI1, I2S are available only on STM32WL31xx/33xx devices.
Any reference to these IPs on PWRC registers is valid within this perimeter.

5.6.1 Control register 1 (PWRC_CR1)

This register control the BOR in Shutdown, the BORH feature, the low power mode selection and the IO control owner.

Address offset: 0x00

Reset value: 0x0000 0114

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

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

Bit 8 **ENBORL**: Enable BORL reset supervisioning during Run mode
- 0: No BORL is monitored during Run mode.
- 1: BORL is monitored during Run mode (a POR reset happens if VDDIO goes below 1.6 V during Run mode).

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

Bit 5 **ENBORH**: Enable the BORH software configuration.
- 0: BORH is OFF (VBOR0): threshold is 1.6 V (default).
- 1: BORH is enabled, the threshold is

Bit 4 **APC**: Apply pull-up/-down configuration from PWRC or GPIO register.
- 0: The GPIOx_PUPDR of the GPIO block are used to control the product pull-up/-down of the IOs.
- 1: The PWRC_PUCRx and PWRC_PDCRx of the PWRC block are used to control the product pull-up/-down of the IOs (default).

Bit 3 **IBIAS_RUN_STATE**: Enable/Disable IBIAS during Run mode when automatic mode is disabled
- 0: IBIAS control is disabled (default)
- 1: IBIAS control is enabled

Bit 2 **IBIAS_RUN_AUTO**: Enable automatic IBIAS control during Run/Deepstop mode.
- 0: IBIAS control is manual (and controlled by IBIAS_RUN_STATE register)
- 1: IBIAS control is automatic (default)

Bit 1 **ENSDNBOR**: Enable BOR reset supervising during Shutdown mode.
- 0: No BOR is monitored during Shutdown mode (default).
- 1: BOR is monitored during Shutdown mode (a POR reset happens if VDDIO goes below 1.6 V during Shutdown mode).

Note: Enabling this feature prevents blocking the device if VDDIO goes below supported voltages during Shutdown mode. However, it adds an overconsumption.

Bit 0 **LPMS**: Low Power Mode Selection.

This bit defines if the device enters Deepstop or Shutdown mode when both CPU and MR_SUBG requests a low power mode entry.
- 0: Deepstop mode (default).
- 1: Shutdown mode.

5.6.2 Control register 2 (PWRC_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RFREGON_STATUS	RFREGRDY	RFREGBYP	RFREGCEXT	RFREGEN	Res.	GPIORET	LPREG_VH_STATUS	LPREG_FORCE_VH	RAMRET1	DBGRET	PVDLS[2:0]			PVDE
	r	r	rw	rw	rw		rw	r	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RFREGON_STATUS**: RF Regulator On Status.
- 0: RF Regulator is disabled
- 1: RF Regulator is enabled

Bit 13 **RFREGRDY**: RF Regulator Ready flag
- 0: RF Regulator is not ready
- 1: RF Regulator is ready

Bit 12 **RFREGBYP**: RF Regulator Bypass Enable
- 0: internally generated 1.2 V
- 1: LDO output connected to VSMPS

Bit 11 **RFREGCEXT**: RF Regulator External Supply Bypass
- 0: Internal supply only
- 1: External supply bypass capability

Bit 10 **RFREGEN**: RF Regulator enable
- 0: Disable RF Regulator (Note: RF Regulator can still be enabled by the RFSUGB or RCC_CR.HSEON)
- 1: Enable RF Regulator

Bit 9 Reserved, must be kept at reset value.

Bit 8 **GPIORET**: GPIO retention enable.
- 0: GPIOs don't retain their configuration during and exiting from Deepstop (default)
- 1: GPIOs retain their configuration during and exiting from Deepstop.

Bit 7 **LPREG_VH_STATUS**: LPREG status
- 0: LPREG=1 V during Deepstop mode (default).
- 1: LPREG=1.2 V during Deepstop mode.

Bit 6 **LPREG_FORCE_VH**: force LPREG=1.2 V during Deepstop mode.
Set and clear by software. This bit allows to select the LPREG voltage. Note: When the LCD is enable (LCD_CR.LCDEN=1) or the Comparator SCALER is enable (COMP_CSR.SCALEEN=1 and COMP_CSR.EN=1) the LPREG voltage is set at 1.2 V by hardware
- 0: LPREG=1 V during Deepstop mode (default).
- 1: Force LPREG=1.2 V during Deepstop mode.

Bit 5 **RAMRET1**: RAM1 (STM32WL33xx only) bank retention in Deepstop mode.
- 0: SRAM1 bank is not retained during Deepstop mode (default).
- 1: SRAM1 bank is retained during Deepstop mode.

Bit 4 **DBGRET**: PA2 and PA3 retention enable after Deepstop
- 0: PA2, PA3 GPIOs don't retain their configuration exiting from Deepstop (default).
- 1: PA2, PA3 GPIOs retain their configuration exiting from Deepstop.

Bits 3:1 **PVDLS[2:0]**: Programmable Voltage Detector level selection:
- 000: 2.05V - Lowest level
- 001: 2.20 V
- 010: 2.36 V
- 011: 2.52 V
- 100: 2.64 V
- 101: 2.81 V
- 110: 2.91 V - Highest level
- 111: Reserved

Bit 0 **PVDE**: Programmable Voltage Detector enable.
- 0: the Power Voltage Detector feature is disabled (default).
- 1: the Power Voltage Detector feature is enabled.

Note: it is mandatory to ensure GPIORET bit is set before entering Deepstop in order GPIOs retain their configuration.

5.6.3 Internal source wakeup enable register (PWRC_IEWU)

This register manages the selection of the internal wakeup sources to get out of Deepstop mode.

Note: All wakeup sources are disabled by default after reset.

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	EWLPA WUR	EWMR SUBG HCPU	EWMR SUBG	Res.	Res.	Res.	EIWL4	EIWL3	EIWL2	EIWL1	EIWL0
					rw	rw	rw				rw	rw	rw	rw	rw

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

Bit 10 **EWLPAWUR**: Enable wakeup on LPAWUR event
- 0: Wakeup on LPAWUR event is disabled (default).
- 1: Wakeup on LPAWUR event is enabled.

Bit 9 **EWMRSUBGHCPU**: Enable wakeup on MRSUBG Host CPU event.
- 0: Wakeup on MRSUBG Host CPU event is disabled (default).
- 1: Wakeup on MRSUBG Host CPU event is enabled.

Bit 8 **EWMRSUBG**: Enable wakeup on MRSUBG RFIP event.
- 0: Wakeup on MRSUBG RFIP event is disabled (default).
- 1: Wakeup on MRSUBG RFIP event is enabled.

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

Bit 4 **EIWL4**: Enable wakeup on LCSC event (LCSC).
- 0: Wakeup on internal line is disabled (default).
- 1: Wakeup on internal line is enabled.

Bit 3 **EIWL3**: Enable wakeup on Internal event (COMP).
- 0: Wakeup on internal line is disabled (default).
- 1: Wakeup on internal line is enabled.

Bit 2 **EIWL2**: Enable wakeup on Internal event (LCD).
- 0: Wakeup on internal line is disabled (default).
- 1: Wakeup on internal line is enabled.

Bit 1 **EIWL1**: Enable wakeup on Internal event (RTC).
- 0: Wakeup on internal line is disabled (default).
- 1: Wakeup on internal line is enabled.

Bit 0 **EIWL0**: Enable wakeup on Internal event (LPUART).
- 0: Wakeup on internal line is disabled (default).
- 1: Wakeup on internal line is enabled.

5.6.4 Internal source wakeup polarity register (PWRC_IWUP)

This register manages the polarity for the internal wakeup sources to get out of Deepstop mode.

Note: The wakeup event are only edge detection, not level detection.

Address offset: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	WLPAWURP	WMRSUBGHCPUP	WMRSUBGP	Res.	Res.	Res.	IWUP4	IWUP3	IWUP2	IWUP1	IWUP0
					rw	rw	rw				rw	rw	rw	rw	rw

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

Bit 10 **WLPAWURP**: Wakeup polarity for LPAWUR event.
- 0: Detection of wakeup event on rising edge (default).
- 1: Detection of wakeup event on falling edge.

Bit 9 **WMRSUBGHCPUP**: Wakeup polarity for MRSUBG Host CPU event.
- 0: Detection of wakeup event on rising edge (default).
- 1: Detection of wakeup event on falling edge.

Bit 8 **WMRSUBGP**: Wakeup polarity for MRSUBG event.
- 0: Detection of wakeup event on rising edge (default).
- 1: Detection of wakeup event on falling edge.

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

Bit 4 **IWUP4**: Wakeup polarity for internal wakeup line 4 event (LCSC).
- 0: Detection of wakeup event on rising edge (default).
- 1: Detection of wakeup event on falling edge.

Bit 3 **IWUP3**: Wakeup polarity for internal wakeup line 3 event (COMP).
- 0: Detection of wakeup event on rising edge (default).
- 1: Detection of wakeup event on falling edge.

Bit 2 **IWUP2**: Wakeup polarity for internal wakeup line 2 event (LCD).
- 0: Detection of wakeup event on rising edge (default).
- 1: Detection of wakeup event on falling edge.

Bit 1 **IWUP1**: Wakeup polarity for internal wakeup line 1 event (RTC).
- 0: Detection of wakeup event on rising edge (default).
- 1: Detection of wakeup event on falling edge.

Bit 0 **IWUP0**: Wakeup polarity for internal wakeup line 0 event (LPUART).
- 0: Detection of wakeup event on rising edge (default).
- 1: Detection of wakeup event on falling edge.

5.6.5 Internal source wakeup status register (PWRC_IWUF)

This register provides the information concerning which internal source woke up the device after a Deepstop.

*A*ddress offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	WLPAWURF	WMRSUBGHCPUF	WMRSUBGF	Res.	Res.	Res.	IWUF4	IWUF3	IWUF2	IWUF1	IWUF0
					rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bit 10 **WLPAWURF**: LPAWUR wakeup flag
- 0: no wakeup from LPAWUR occurred since last clear.
- 1: a wakeup from LPAWUR occurred since last clear.

Cleared by writing 1 in this bit.

Bit 9 **WMRSUBGHCPUF**: MRSUBG Host CPU wakeup flag.
- 0: no wakeup from MRSUBG Host CPU occurred since last clear.
- 1: a wakeup from MRSUBG Host CPU occurred since last clear.

Cleared by writing 1 in this bit.

Bit 8 **WMRSUBGF**: MRSUBG wakeup flag.
- 0: no wakeup from MRSUBG occurred since last clear.
- 1: a wakeup from MRSUBG occurred since last clear.

Cleared by writing 1 in this bit.

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

Bit 4 **IWUF**4: Internal wakeup flag (LCSC).
- 0: no wakeup from LCSC occurred since last clear.
- 1: a wakeup from LCSC occurred since last clear.

Cleared by writing 1 in this bit.

Bit 3 **IWUF**3: Internal wakeup flag (COMP).
- 0: no wakeup from COMP occurred since last clear.
- 1: a wakeup from COMP occurred since last clear.

Cleared by writing 1 in this bit.

Bit 2 **IWUF**2: Internal wakeup flag (LCD).
- 0: no wakeup from LCD occurred since last clear.
- 1: a wakeup from LCD occurred since last clear.

Cleared by writing 1 in this bit.

Bit 1 **IWUF1**: Internal wakeup flag (RTC).
- 0: no wakeup from RTC occurred since last clear.
- 1: a wakeup from RTC occurred since last clear.

Cleared by writing 1 in this bit.

Bit 0 **IWUF0**: Internal wakeup flag (LPUART).
- 0: no wakeup from LPUART occurred since last clear.
- 1: a wakeup from LPUART occurred since last clear.

Cleared by writing 1 in this bit.

5.6.6 Status register 2 (PWRC_SR2)

This register provides some status flags related to the Power Voltage Detector and the SMPS blocks.

*A*ddress offset: 0x14

Reset value: 0x0000 X3X6

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOBOOTVAL[3:0]				PVDO	Res.	REGMS	REGLPS	IOBOOTVAL2[3:0]				Res.	SMPSRDY	SMPSENR	SMPSBYPR
r	r	r	r	r		r	r	r	r	r	r		r	r	r

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

Bits 15:12 **IOBOOTVAL**: I/Os value latched at POR.
- bit 3: PA11 input value latched at POR,
- bit 2: PA10 input value latched at POR,
- bit 1: PA9 input value latched at POR,
- bit 0: PA8 input value latched at POR.

Note: This information may be used by the boot loader to manage boot on serial interfaces for instance.

Bit 11 **PVDO**: Power voltage Detector Output.
When the Power voltage Detector is enabled (PWRC_CR2.PVDE=1), this bit indicates when the VDDIO is lower than the selected threshold (through PWRC_CR2.PVDLS bit field).
- 0: The VDDIO is not lower than threshold or PVD feature is not enabled.
- 1: The VDDIO is lower than the selected threshold.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **REGMS**: Main regulator ready status.
- 0: The Main regulator is not ready.
- 1: The Main regulator is ready.

Bit 8 **REGLPS**: Low Power regulator ready status.
- 0: The Low Power regulator is not ready.
- 1: The Low Power regulator is ready.

Bits 7:4 **IOBOOTVAL2**: I/Os value latched at POR.
- bit 3: PB15 input value latched at POR,
- bit 2: PB14 input value latched at POR,
- bit 1: PB13 input value latched at POR,
- bit 0: PB12 input value latched at POR.

Note: This information may be used by the boot loader to manage boot on serial interfaces for instance.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SMPSRDY**: SMPS ready status.
- 0: SMPS regulator is not ready.
- 1: SMPS regulator is ready.

Bit 1 **SMPSENR**: SMPS Run mode status.
This bit mirrors the internal ENABLE_3V3 control signal connected to the SMPS and driven by the hardware.
- 0: SMPS regulator is not regulating (in PRECHARGE or NOSMPS mode).
- 1: SMPS regulator is in Run mode.

Bit 0 **SMPSBYPR**: SMPS PRECHARGE mode status.
This bit mirrors the PRECHARGE control state of the SMPS.
- 0: SMPS regulator is not in PRECHARGE mode.
- 1: SMPS regulator is in PRECHARGE mode (VSMPS connected to VDDIO)

5.6.7 Control register 5 (PWRC_CR5)

This register is used to configure the SMPS.

*A*ddress offset: 0x1C

Reset value: 0x0000 6024

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPS_BOF_DYN	SMPS_PRECH_CUR_SEL[1:0]		CLKDETR_DISABLE	SMPS_ENA_DCM	NOSMPS	SMPSFBYP	SMPSLPOPEN	NOSMPS_BOF	SMPS_BOF_STATIC	SMPSBOMSEL[1:0]		SMPSLVL[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **SMPS_BOF_DYN**: SMPS Bypass on the Fly dynamic
- 0: disabled (by default)
- 1: SMPS Bypass on the fly dynamic is enabled (EN_LDO=1)

Bits 14:13 **SMPS_PRECH_CUR_SEL[1:0]**: Select SMPS PRECHARGE limit current.
- 00: 2.5 mA
- 01: 5 mA
- 10: 10 mA
- 11: 20 mA (default)

Bit 12 **CLKDETR_DISABLE**: disable the SMPS clock detection
The SMPS clock detection enables an automatic SMPS bypass switching in case of unexpected loss of the SMPS clock.
- 0: SMPS clock detection mechanism enabled (default).
- 1: SMPS clock detection mechanism disabled.

Bit 11 **SMPS_ENA_DCM**: Discontinuous conduction mode enable.
- 0: SMPS DCM is disabled (default).
- 1: SMPS DCM is enabled.

Bit 10 **NOSMPS**: No SMPS mode.

– 0: SMPS is enabled (default).

– 1: SMPS is disabled.

Note: this configuration (SMPS disabled) should be used only when the SMPS_FB pad is directly connected to the VBATT (external voltage), without L/C BOM.

Bit 9 **SMPSFBYP**: Force the SMPS in PRECHARGE mode.

– 0: no effect (default).

– 1: SMPS is disabled and bypassed.

Note: when this bit is set, the VSMPS output is connected to the VDDIO. The actual state of the SMPS is visible in the SMPS mode status bits in PWRC_SR2 register.

Bit 8 **SMPSLPOPEN**: Select OPEN mode instead of PRECHARGE mode for the SMPS during Deepstop.

– 0: in Deepstop, the SMPS is in PRECHARGE mode with output connected to VDDIO (default).

– 1: in Deepstop, the SMPS is disabled with floating output.

Bit 7 **NOSMPS_BOF**: No SMPS Mode in the case of BOF usage

When this bit is set, the SMPS regulator is disabled.

– 0: No effect, SMPS is enabled. (default)

– 1: SMPS is disabled;

Note: this configuration should be used only when SMPS_BOF_STATIC=1 or SMPS_BOF_DYN=1 and permits both Run and Deepstop operating modes.

Bit 6 **SMPS_BOF_STATIC**: SMPS Bypass on the Fly static
– 0: disabled (by default)
– 1: SMPS Bypass on the fly static is enabled (EN_SW=1)

Bits 5:4 **SMPSBOMSEL[1:0]**: Select the SMPS BOM.
– 00: BOM1
– 01: BOM2
– 10: BOM3 (default)
– 11: Not applicable
Note: BOM correspondence/details is available in Table 12: SMPS BOM information.

Bits 3:0 **SMPSLVL[3:0]**: Select the SMPS output voltage level.
This bit field selects the SMPS voltage output level with a granularity of 100 mV.
– 0000: 1.20 V (min VBAT = 1.95V)
– 0001: 1.20 V (min VBAT = 1.95V)
– 0010: 1.20 V (min VBAT = 1.95V)
– 0011: 1.30 V (min VBAT = 1.95V)
– 0100: 1.40 V (min VBAT = 2.0 V) (default)
– 0101: 1.50 V (min VBAT = 2.0 V)
– 0110: 1.60 V (min VBAT = 2.15V)
– 0111: 1.70 V (min VBAT = 2.2 V)
– 1000: 1.80 V (min VBAT = 2.3 V)
– 1001: 1.90 V (min VBAT = 2.45V)
– 1010: 2 V (min VBAT = 2.6 V)
– 1011: 2.1 V (min VBAT = 2.7V)
– 1100: 2.2 V (min VBAT = 2.8 V)
– 1101: 2.3 V (min VBAT = 2.8 V)
– 1110: 2.4 V (min VBAT = 2.9 V)
– 1111: 2.4 V (min VBAT = 2.9 V)
– Warning: The SMPS output voltage must not be changed while the SMPS is in use. The sequence to reprogram a new SMPS output voltage is described in *Section 5.8.2: SMPS output level re-programming*.

5.6.8 I/O Port A pull-up control register (PWRC_PUCRA)

This register is used to control the pull-up for the PA0 to PA15 I/O when the PWRC_CR1.APC bit is set.

Caution: If both pull-up and pull-down are enabled in the PWRC_PUCRA and PWRC_PDCRA registers for an I/O, then pull-down is applied.

The user must take care to disable the pull on I/O programmed in Analog mode.

Address offset: 0x20

Reset value: 0x0000 FFF7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUA15	PUA14	PUA13	PUA12	PUA11	PUA10	PUA9	PUA8	PUA7	PUA6	PUA5	PUA4	PUA3	PUA2	PUA1	PUA0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **PUAx** (x=15 to 0): pull-up enable for PAx I/O.
- 0: No pull-up.
- 1: Pull-up enabled (default).

5.6.9 I/O Port A pull-down control register (PWRC_PDCRA)

This register is used to control the pull-down for the PA0 to PA15 I/O when the PWRC_CR1.APC bit is set.

Caution: If both pull-up and pull-down are enabled in the PWRC_PUCRA and PWRC_PDCRA registers for an I/O, then pull-down is applied.

The user must take care to disable the pull on I/O programmed in Analog mode.

Address offset: 0x24

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDA15	PDA14	PDA13	PDA12	PDA11	PDA10	PDA9	PDA8	PDA7	PDA6	PDA5	PDA4	PDA3	PDA2	PDA1	PDA0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **PDAx** (x=15 to 0): pull-down enable for PAx I/O.
– 0: No pull-down (default).
– 1: pull-down enabled.

5.6.10 I/O Port B pull-up control register (PWRC_PUCRB)

This register is used to control the pull-up for the PB0 to PB15 I/O when the PWRC_CR1.APC bit is set.

Caution: If both pull-up and pull-down are enabled in the PWRC_PUCRA and PWRC_PDCRA registers for an I/O, then pull-down is applied.

The user must take care to disable the pull on I/O programmed in Analog mode.

Address offset: 0x28

Reset value: 0x0000 FFFF

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUB15	PUB14	PUB13	PUB12	PUB11	PUB10	PUB9	PUB8	PUB7	PUB6	PUB5	PUB4	PUB3	PUB2	PUB1	PUB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **PUBx** (x=15 to 0): Pull-up enable for PBx I/O.
- 0: No pull-up.
- 1: Pull-up enabled (default).

5.6.11 I/O Port B pull-down control register (PWRC_PDCRB)

This register is used to control the pull-down for the PB0 to PB15 I/O when the PWRC_CR1.APC bit is set.

Caution: If both pull-up and pull-down are enabled in the PWRC_PUCRA and PWRC_PDCRA registers for an I/O, then pull-down is applied.

The user must take care to disable the pull on I/O programmed in Analog mode.

Address offset: 0x2C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDB15	PDB14	PDB13	PDB12	PDB11	PDB10	PDB9	PDB8	PDB7	PDB6	PDB5	PDB4	PDB3	PDB2	PDB1	PDB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **PDBx** (x=15 to 0): pull-down enable for PBx I/O.
- 0: No pull-down (default).
- 1: pull-down enabled.

5.6.12 Port A wakeup enable register (PWRC_EWUA)

This register manages the selection of the wakeup sources related to IO port A to get out of Deepstop mode.

Note: All wakeup sources are disabled by default after reset.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EWU15	EWU14	EWU13	EWU12	EWU11	EWU10	EWU9	EWU8	EWU7	EWU6	EWU5	EWU4	EWU3	EWU2	EWU1	EWU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EWUx** (x = 15 to 0): Enable wakeup on PAx I/O event.
- 0: Wakeup on PAx I/O line is disabled (default).
- 1: Wakeup on PAx I/O line is enabled.

5.6.13 Port A wakeup polarity register (PWRC_WUPA)

This register manages the polarity for the port A I/Os wakeup sources to get out of Deepstop mode.

Note: The wakeup event are only edge detection, not level detection.

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WUP15	WUP14	WUP13	WUP12	WUP11	WUP10	WUP9	WUP8	WUP7	WUP6	WUP5	WUP4	WUP3	WUP2	WUP1	WUP0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **WUPx** (x = 15 to 0): Wakeup polarity for PAx IO event.
- 0: Detection of wakeup event on rising edge (default).
- 1: Detection of wakeup event on falling edge.

5.6.14 Port A wakeup status register (PWRC_WUFA)

This register provides the information concerning which port A I/O source woke up the device after a Deepstop.

*A*ddress offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUF15	WUF14	WUF13	WUF12	WUF11	WUF10	WUF9	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1	WUF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 15:0 **WUFx** (x = 15 to 0): PAx I/O wakeup flag.
- 0: no wakeup from PAx I/O occurred since last clear.
- 1: a wakeup from PAx I/O occurred since last clear.

Cleared by writing 1 in this bit.

5.6.15 Port B wakeup enable register (PWRC_EWUB)

This register manages the selection of the wakeup sources related to IO port B to get out of Deepstop mode.

Note: All wakeup sources are disabled by default after reset.

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWU14	EWU14	EWU13	EWU12	EWU11	EWU10	EWU9	EWU8	EWU7	EWU6	EWU5	EWU4	EWU3	EWU2	EWU1	EWU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **EWUx** (x = 15 to 0): Enable wakeup on PBx I/O event.
- 0: Wakeup on PBx I/O line is disabled (default).
- 1: Wakeup on PBx I/O line is enabled.

5.6.16 Port B wakeup polarity register (PWRC_WUPB)

This register manages the polarity for the port B I/Os wakeup sources to get out of Deepstop mode.

Note: The wakeup event are only edge detection, not level detection.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WUP15	WUP14	WUP13	WUP12	WUP11	WUP10	WUP9	WUP8	WUP7	WUP6	WUP5	WUP4	WUP3	WUP2	WUP1	WUP0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **WUPx** (x = 15 to 0): Wakeup polarity for PBx IO event.
– 0: Detection of wakeup event on rising edge (default).
– 1: Detection of wakeup event on falling edge.

5.6.17 Port B wakeup status register (PWRC_WUFB)

This register provides the information concerning which port B I/O woke up the device after a Deepstop.

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WUF15	WUF14	WUF13	WUF12	WUF11	WUF10	WUF9	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1	WUF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 15:0 **WUFx (x = 15 to 0)**: PBx I/O wakeup flag.
– 0: no wakeup from PBAx I/O occurred since last clear.
– 1: a wakeup from PBx I/O occurred since last clear.
Cleared by writing 1 in this bit.

5.6.18 Shutdown I/O wakeup enable register (PWRC_SDWN_WUEN)

Address offset: 0x4C

Reset value: 0x0000 0000

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

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

Bit 0 **WUEN**: PB0I/O wakeup from Shutdown modes.
- 0: PB0 disable pin wakeup from Shutdown modes.
- 1: PB0 enable pin wakeup from Shutdown modes.

5.6.19 Shutdown I/O wakeup polarity register (PWRC_SDWN_WUPOL)

Address offset: 0x50

Reset value: 0x0000 0000

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

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

Bit 0 **WUPOL**: PB0 I/O wakeup from shutdown polarity.
- 0: PB0 pin wakeup from Shutdown modes on high pulse or high level detection.
- 1: PB0 pin wakeup from Shutdown modes on low pulse or low level detection.

5.6.20 Shutdown I/O Wakeup flag register (PWRC_SDWN_WUF)

This register is reset on PORESETn.

*A*ddress offset: 0x54

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF
															rc_w0

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

Bit 0 **WUF**: PB0 pin I/O wakeup from Shutdown mode flags.
- 0: Shutdown wakeup from PB0 pin has not occurred.
- 1: Shutdown wakeup from PB0 pin has occurred.

5.6.21 Bypass on-the-fly tuning register (PWRC_BOF_TUNE)

*A*ddress offset: 0x58

Reset value: 0x0000 0004

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

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

Bits 3:0 **BOF_TUNE[3:0]**: SMPS Bypass on the fly (BOF) tuning bits.

– 0000: 1.2 V
– 0001: 1.2 V
– 0010: 1.2 V
– 0011: 1.3 V
– 0100: 1.4 V (Default)
– 0101: 1.5V
– 0110: 1.6 V
– 0111: 1.7 V
– 1000: 1.8 V
– 1001: 1.9 V
– 1010: 2 V
– 1011: 2.1 V
– 1100: 2.2 V
– 1101: 2.3 V
– 1110: 2.4 V
– 1111: 2.4 V

5.6.22 Debug register (PWRC_DBGR)

This register is used for debug features.

This register is reset on PORESETn.

*A*ddress offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEEPSTOP2
															rw

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

Bit 0 **DEEPSTOP2**: DEEPSTOP2 low power saving emulation enable.
- 0: normal Deepstop is applied
- 1: DEEPSTOP2 (debugger features not lost) is applied instead of Deepstop.

5.6.23 Extended status and reset register (PWRC_EXTSRR)

This register provides flags about RF SUBGHz radio activity start and Deepstop sequence occurrence or not.

*A*ddress offset: 0x88

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RFPHA SEF	DEEPS TOPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rc_w1	rc_w1									

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

Bit 10 **RFPHASEF**: RFPHASE Flag.
- 0: the RF SUBGHz IP does not require any attention.
- 1: the RF SUBGHz IP is awake and may require a system attention.

This bit is set by hardware when a radio wakeup event occurs.
This bit is reset by hardware when the RF SUBGHz IP raises the “ready to sleep” information.
The software can reset this bit by writing 1 in it.

Bit 9 **DEEPSTOPF:** System DEEPTSTOP Flag.
- 0: the device did not enter Deepstop mode.
- 1: the device entered a Deepstop mode.

This bit is set by hardware when a Deepstop sequence occurred.
The software can reset this bit by writing 1 in it.

Bits 8:0 Reserved, must be kept at reset value.

5.6.24 Debug SMPS register (PWRC_DBGSMPS)

This register drives some control signals for the SMPS.

*A*ddress offset: 0x8C

Reset value: 0x0000 8000

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

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

Bit 15:14 **BOF_CUR_SEL[1:0]**: Bypass On the Fly current limitation
– 00: 20mA
– 01: 40mA
– 10: 60mA (default)
– 11: no limit

Bit 13 **ILIM_BOOST:** SMPS current limitation Boost
– 0: Max current = 110 mA (default)
– 1: Max current = 130 mA

Bit 12 **DIS_ILIM**: DIS_ILIM_3V3 SMPS control signal.

Bit 11:0 Reserved, must be kept at reset value.

5.7 PWRC register map

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the PWRC base address location in STM32WL30xx/31xx/33xx devices.

Note: All the PWRC registers are retained during Deepstop mode.

The salmon cells indicates the bit fields located in the VDD33 power domain. This implies the associated feature is applied even during Shutdown state (but are lost at Shutdown mode exit as a PORESETn is generated).

Table 13. PWRC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	PWRC_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENBORL	Res.		ENBORH	APC	IBIAS_RUN_STATE	IBIAS_RUN_AUTO	ENSDNBOR	LPMS
	Reset value																								1			0	1	0	1	0	0
0x04	PWRC_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFREGON_STATUS	RFREGRDY	RFREGBYP	RFREGCEXT	RFREGEN	Res.	GPIO_RET	LPREG_VH_STATUS	LPREG_VORCE_VH	RAMRET1	DBGRET	PVDLS			PVDE
	Reset value																		0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x08	PWRC_IEWU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWLPAWUR	EWMRSUBGHCPU	EWMRSUBG	Res.	Res.	Res.	EIWL4	EIWL3	EIWL2	EIWL1	EIWL0
	Reset value																						0	0	0				0	0	0	0	0
0x0C	PWRC_IWUP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLPAWURP	WMRSUBGHCPUP	WMRSUBGP	Res.	Res.	Res.	IWUP4	IWUP3	IWUP2	IWUP1	IWUP0
	Reset value																						0	0	0				0	0	0	0	0
0x10	PWRC_IWUF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLPAWURF	WMRSUBGHCPUF	WMRSUBGF	Res.	Res.	Res.	IWUF4	IWUF3	IWUF2	IWUF1	IWUF0
	Reset value																						0	0	0				0	0	0	0	0
0x14	PWRC_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOBOOTVAL[3:0]				PVDO	Res.	Res.	REGLPS	IOBOOTVAL2[3:0]				Res.	Res.	SMPSENR	SMPSBYPR
	Reset value																	X	X	X	X	0			1	X	X	X	X			1	0

Table 13. PWRC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	Reserved																																
0x1C	PWRC_CR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPS_BOF_DYN	SMPS_PRECH_CUR_SEL[1:0]		CLKDETR_DISABLE	SMPS_ENA_DCM	NOSMPS	SMPSFBYP	SMPSLPOEN	NOSMPS_BOF	SMPS_BOF_STATIC	SMPSBOMSEL[1:0]		SMPSLVL[3:0]			
	Reset value																	0	1	1	0	0	0	0	0	0	0	1	0	0	1	0	0
0x20	PWRC_PUCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUA15	PUA14	PUA13	PUA12	PUA11	PUA10	PUA9	PUA8	PUA7	PUA6	PUA5	PUA4	PUA3	PUA2	PUA1	PUA0
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0x24	PWRC_PDCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDA15	PDA14	PDA13	PDA12	PDA11	PDA10	PDA9	PDA8	PDA7	PDA6	PDA5	PDA4	PDA3	PDA2	PDA1	PDA0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0x28	PWRC_PUCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUB15	PUB14	PUB13	PUB12	PUB11	PUB10	PUB9	PUB8	PUB7	PUB6	PUB5	PUB4	PUB3	PUB2	PUB1	PUB0
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x2C	PWRC_PDCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDB15	PDB14	PDB13	PDB12	PDB11	PDB10	PDB9	PDB8	PDB7	PDB6	PDB5	PDB4	PDB3	PDB2	PDB1	PDB0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	PWRC_EWUA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU15	EWU14	EWU13	EWU12	EWU11	EWU10	EWU9	EWU8	EWU7	EWU6	EWU5	EWU4	EWU3	EWU2	EWU1	EWU0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	PWRC_WUPA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUP15	WUP14	WUP13	WUP12	WUP11	WUP10	WUP9	WUP8	WUP7	WUP6	WUP5	WUP4	WUP3	WUP2	WUP1	WUP0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	PWRC_WUFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF15	WUF14	WUF13	WUF12	WUF11	WUF10	WUF9	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1	WUF0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	Reserved																																
0x40	PWRC_EWUB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU15	EWU14	EWU13	EWU12	EWU11	EWU10	EWU9	EWU8	EWU7	EWU6	EWU5	EWU4	EWU3	EWU2	EWU1	EWU0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	PWRC_WUPB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUP15	WUP14	WUP13	WUP12	WUP11	WUP10	WUP9	WUP8	WUP7	WUP6	WUP5	WUP4	WUP3	WUP2	WUP1	WUP0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	PWRC_WUFB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF15	WUF14	WUF13	WUF12	WUF11	WUF10	WUF9	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1	WUF0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	PWRC_SDWN_WUEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUEN
	Reset value																																0

Table 13. PWRC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	PWRC_SDWN_WUPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPOL
	Reset value																																0
0x54	PWRC_SDWN_WUF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF
	Reset value																																0
0x58	PWRC_BOF_TUNE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOF_TUNE[2:0]			
	Reset value																													0	1	0	0
0x5C - 0x80	Reserved																																
0x84	PWRC_DBGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			Res.	Res.	Res.			Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEEPSTOP2
	Reset value																																0
0x88	PWRC_EXTSRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFPHASEF	DEEPSTOPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																						0	0									
0x8C	PWRC_DBGSMPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOF_CUR_SEL[1:0]		ILIM_BOOST	DIS_ILIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	1	0	0	0												

5.8 Programmer's model

5.8.1 Reset reason management

The CPU has many reasons to be reset and executes its reset handler. *Table 14* provides an overview of the flags that can help the embedded software to get the root cause of the CPU restart.

Table 14. Flags versus CPU reboot reason

CPU reboot reason	RCC_CSR					PWRC_ISCR (in SYSCFG)	PWRC_EXT SRR
	LOCKUPRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	WAKEUP_ISC	DEEPSTOPF
POR/BOR reset	-	-	-	1	1	-	-
NRSTn pad reset	-	-	-	-	1	-	-
Watchdog reset	-	1	-	-	1	-	-
System reset (CPU request)	-	-	1	-	1	-	-
LOCKUP reset	1	-	-	-	1	-	-
Deepstop exit on wakeup event	-	-	-	-	-	1	1
Deepstop exit on watchdog reset	-	1	-	-	1	-	-
Deepstop exit on NRSTn pad reset	-	-	-	-	1	-	-
Deepstop exit on POR/BOR	-	-	-	1	1	-	-
Shutdown exit	-	-	-	1	1	-	-

If the reboot reason is a wakeup from Deepstop, then the wakeup source(s) can be read in the PWRC_IWUF/WUFA/WUFB register as shown in *Table 15*.

Table 15. Wakeup reason flags

Wakeup reason	PWRC_IWUF				PWRC_SR WUFA/B
	IWUFx	WLPAWUR	WMRSUB GHCPUF	WMRSUB GF	WUFx
Wakeup on MR_SUBG event	-	-	-	1	-
Wakeup on Host timer in MR_SUBG event	-	-	1	-	-
Wakeup on LPAWUR(1) event	-	1	-	-	-
Wakeup on RTC event	1	-	-	-	-
Wakeup on LPUART event	1	-	-	-	-
Wakeup on LCD(1) event	1	-	-	-	-
Wakeup on COMP(1) event	1	-	-	-	-
Wakeup on LCSC(1) event	1	-	-	-	-
Wakeup on I/Os	-	-	-	-	1

1. Comparator, LCSC, LCD, LPAWUR are available only on STM32WL33xx devices.

Note: If several (enabled) wakeup events occur, several bits are high in the PWRC_IWUF/WUFA/WUFB.

The wakeup flags are set as soon as a wakeup event (enabled in the corresponding PWRC_xEWUx register) occurs, the associated flag is set in the related PWRC_xWUFx register even if the device is in active mode or in the sleep exit sequence (initiated by another wakeup source).

Caution: Those flags have to be cleared by software knowing a Deepstop entry sequence cannot happen if a wakeup flag is already active when the system request a Deepstop mode.

5.8.2 SMPS output level re-programming

The SMPS output voltage cannot be modified on-the-fly when the modification implies to change more than one step (+/-100mV). In this case the following sequence must be respected:

- Set PWRC_CR5.SMPSBYP = 1
- Wait for PWRC_SR2.SMPSRDY = 0
- Program the new targeted value in PWRC_CR5.SMPSLVL[3:0]
- Clear PWRC_CR5.SMPSBYP = 0
- Wait for PWRC_SR2.SMPSRDY = 1

Caution: This sequence must be launched only when no radio activity / RF transfer is on- going.

6 Reset and clock controller (RCC)

The RCC block manages the clock and reset generation for all the STM32WL30xx/31xx/33xx device peripherals.

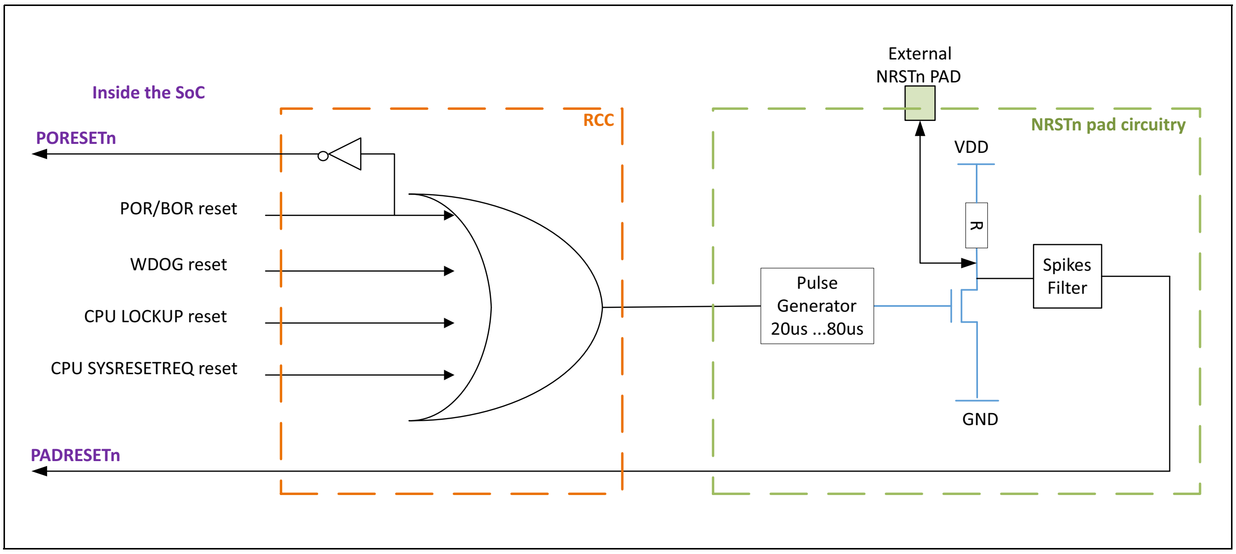
Note: Comparator, DAC, I2C2, LCSC, LCD, LPAWUR are available only on STM32WL33xx devices.
ADC, temperature sensor, I2C1, SPI1, I2S are available only on STM32WL31xx/33xx devices.
Any reference to these IPs on RCC chapter is valid within this perimeter.

6.1 Reset management

6.1.1 General description

The *Figure 12* shows the general principle of reset generation.

Figure 12. Reset generation



Note: The system reset information is output on the NRSTn pad to inform the external world and reset other elements on the board if needed.

Two different resets are available in the design:

- PORESETn: this reset is provided by the APMU analog block and corresponds to a POR or BOR root cause. It is linked to power voltage ramp-up or ramp-down.

The PORESETn reset impacts all the resources of the device.

Note: A Shutdown exits is equivalent to a POR/BOR situation and generates a PORESETn.

- PADRESETn (also termed system reset): this reset is built through several sources:
  - PORESETn,
  - the watchdog reset,
  - the CPU LOCKUP reset,
  - the CPU software system reset,
  - the NRSTn external pad.

Note: The system reset is called PADRESETn as when an internal reset source is activated (watchdog, software, and so on), the NRSTn pad toggles to inform the external world a reset occurs.

This system reset resets all the resources of the device except:
- Debug features (SWD, test registers...)
- flash controller key management part
- RTC timer
- power controller (PWRC)
- part of the RCC registers

The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of reset from the NRSTn external pad, the reset pulse is generated when the pad is asserted low.

6.1.2 Power reset

The PORESETn signal is active when the power supply of the device is below a threshold value or when the regulator does not provide the target voltage.

The PORESETn resets all the resources of the device

6.1.3 Watchdog reset

STM32WL30xx/31xx/33xx devices embed a watchdog timer which may be used to recover from software crashes.

See *Section 23: Independent watchdog (IWDG)* for details about watchdog usage and programming.

6.1.4 LOCKUP reset

The Cortex-M0+ generates a LOCKUP to indicate the core is in the lock-up state resulting from an unrecoverable exception.

The LOCKUP reset is masked if a debugger is connected to the Cortex-M0+. The user can use the SWD to reset or recover the code in this case.

6.1.5 System reset request

The system reset request is generated by the debug circuitry of the Cortex-M0+. The debugger sets the SYSRESETREQ bit of the Application Interrupt and Reset control Register (AIRCR). This system reset request through the AIRCR can also be done by the embedded software (in hard fault handler for instance).

For more details on the Cortex-M0+ system control and ID registers, refer to section B3.2.2 of the *ARMv6-M Architecture Reference Manual*.

6.1.6 Deepstop exit

The low power DEEPSTOP state leads to switch off a part of the 1.2 V (power domain called VDD12i), while keeping the rest of the 1.2 V at 1 V (power domain called VDD12o) and the 3.3 V (VDDIO).

When the device exits the Deepstop mode, only the VDD12i power domain is reset as it is the only power domain that lost the power supply.

6.2 Clock management

Three different clock sources may be used to drive the system clock (CLK_SYS) in the STM32WL30xx/31xx/33xx:

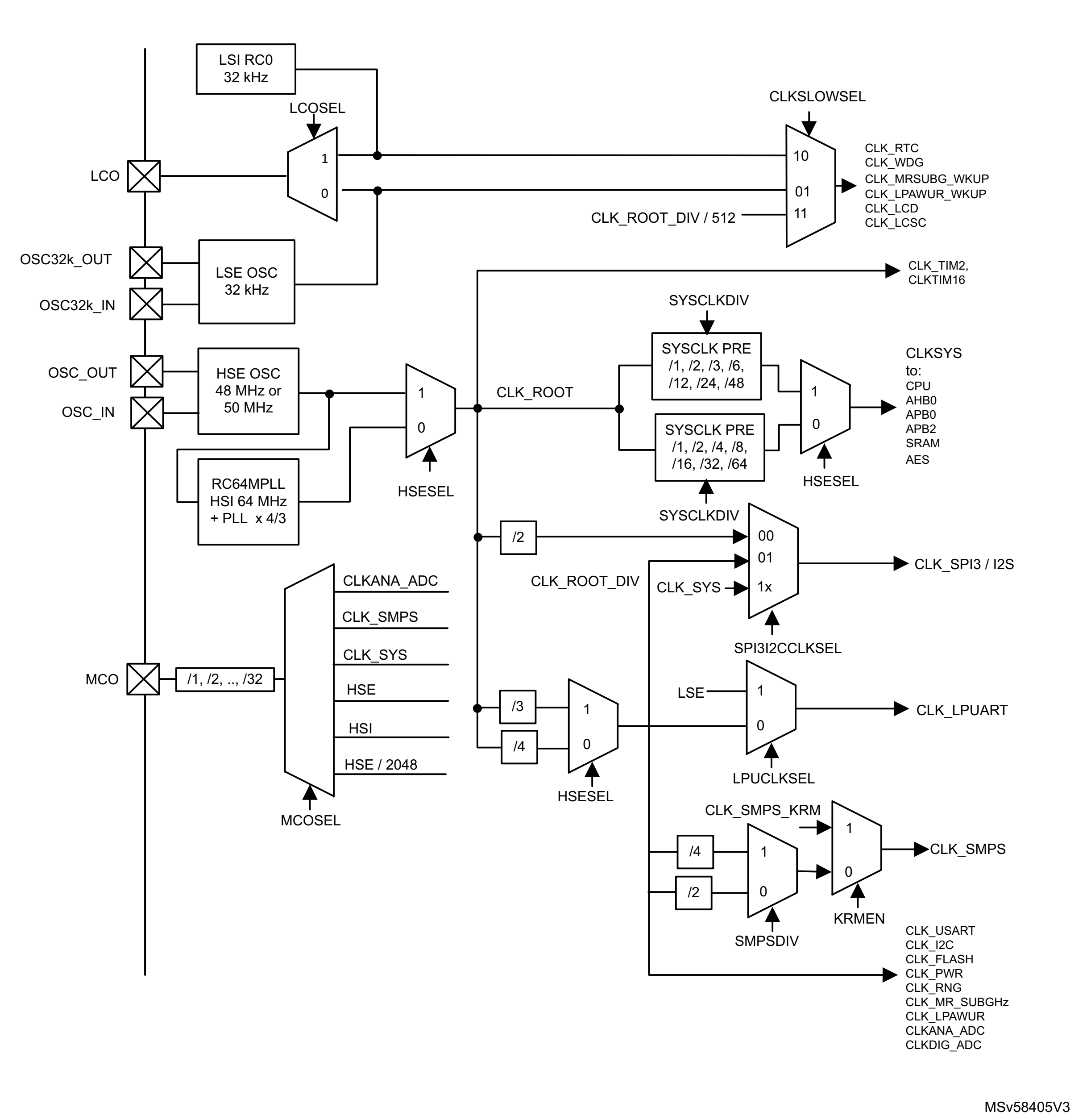
- HSI: high speed internal 64 MHz RC oscillator (provided by the RC64MPLL analog block),
- PLL: PLL clock (provided by the RC64MPLL analog block) equal to HSE x 4/3.
- HSE (High Speed External):
  - high speed 48 MHz or 50 MHz external crystal.
  - or provided by a single ended 48 MHz or 50 MHz input instead of a crystal.

STM32WL30xx/31xx/33xx devices also have a slow frequency clock tree used by some peripherals (RTC, watchdog, LPUART, LCDC, LPAWUR, LCSC and MR_SUBG radio timer). Three different clock sources can be used for this slow clock tree:

- LSI: low speed low drift internal RC with a fixed frequency between 24 kHz and 49 kHz depending on the sample. It is called 32 kHz clock inside this document to simplify.
- LSE:
  - 32.768 kHz low speed external crystal.
  - or provided by a single-ended 32.768 kHz input instead of a crystal.
- The CLK_ROOT_DIV divided by 512. In this case, the slow clock is not available in Deepstop low power mode and it has not be used for peripherals working in Deepstop low power mode.

The *Figure 13* provides an overview of the fast clock tree in STM32WL30xx/31xx/33xx devices.

Figure 13. System clock details



Note: 1 *Comparator, DAC, I2C2, LCSC, LCD, LPAWUR are available only on STM32WL33xx device.*

2 *ADC, temperature sensor, I2C1, SPI1, I2S are available only on STM32WL31xx/33xx devices.*

The HSI and the PLL clocks are provided by the same analog block called RC64MPLL. The output of this block can be:

– a non accurate clock (target is 1% typical) when no external XO provides an input clock to this block,

– an accurate clock when the external XO provides the 48 MHz or 50 MHz and once its internal PLL is locked.

Note: The usage of PLL or HSE as clock source is mandatory for SUBGHz radio operations (need of a high accuracy on the clock).

The software process to switch the system on the accurate clock is indicated in *Section 6.8: Programmer's model*.

This fast clock source is used to generate all the fast clock of the device through dividers as shown in the *Figure 13*.

After reset, the CLK_SYS is divided by four to provide a 16 MHz to the whole system (CPU, DMA, memories and peripherals).

Then the software can program another system clock frequency in the following using CLKSYSDIV bits (see *Clock configuration register (RCC_CFGR)*):

- 110 (forbidden when radio or ADC is in use)
- 101 (forbidden when radio or ADC is in use)
- 100 (forbidden when radio or ADC is in use)
- 011 (forbidden when radio is in use)
- 010
- 001
- 000

Note: Forbidden configuration means that the "in use" feature cannot work if the system clock runs at this frequency.

Note: Special care must be taken when programming the CLK_SYS as some constraints need to be respected:

CLK_SYS frequency must be an integer multiple of 16 MHz greater or equal than CLK_MR_SUBGHz, CLK_LPAWUR.

6.2.1 Peripheral clock details

This fast clock source is also used to generate several internal fast clocks in the system:

- A TIM2/TIM16 kernel clock that is the maximum reachable frequency of the system (output of RCPLL when HSESEL=0 or HSE when HSESEL=1).
- A fixed clock frequency CLK_ROOT_DIV requested by few peripherals like serial interfaces (to maintain fixed baud rate while system clock is switching from a frequency to another) or like flash controller, ADC (STM32WL31xx/33xx only), LPAWUR(STM32W33xx only), and MR_SUBG radio IP (to have a fixed reference clock to manage delays).

Most of the peripherals use only the system clock (CLK_SYS) except:

- I2C (STM32WL31xx/33xx only),USART:
  - In parallel of the system clock, they use a fixed frequency clock (CLK_ROOT_DIV) to have a fixed reference clock for baud rate management. The goal is to allow the CPU to boost or slow down the system clock (depending on on-going activities) without impacting a potential on-going serial interface transfer on external I/Os.
- LPUART:
  - In parallel of the system clock, it uses a fixed frequency clock (CLK_ROOT_DIV) to have a fixed reference clock for baud rate management and LSE clock when active in Deepstop. The goal is to allow the CPU to boost or slow down the system

clock (depending on on-going activities) without impacting a potential on-going serial interface transfer on external I/Os.

- SPI:
  - When using the I2S mode (STM32WL31xx/33xx only), the baud rate is managed through CLK_ROOT/2 clock, CLK_ROOT_DIV cock or system clock (SYS_CLK).

Note: The CPU/system clock frequency must be equal or slower than the I2S clock frequency.

  - When running in other modes than the I2S, the baud rate is managed by the system clock This implies there baud rate is impacted by dynamic system clock frequency changes.
- RNG:
  - In parallel of the system clock, the RNG uses a fixed frequency clock (CLK_ROOT_DIV) to generate at a constant frequency the random number whatever the system clock frequency.
- Flash controller:
  - In parallel of the system clock, the flash controller uses a fixed frequency clock (CLK_ROOT_DIV) to generate specific delays required by the flash memory during programming and erase operation for instance.
- Radio IP (MR_SUBG, and LPAWUR (STM32W33xx only))
  - the radio IPs does not use directly the system clock for their APB / AHB interfaces but the system clock with a potential divider (1 or 2 or 4). *Table 16* shows the supported configurations.
  - In parallel of the CLK_MR_SUBGHz, the MR_SUBG uses a fixed frequency clock (CLK_ROOT_DIV) for modulator, demodulator and to have a fixed reference clock to manage specific delays.

Table 16. System clock versus Radio IPs clock dependency

CLKSYSDIV bits	CLK_MR_SUBGHz
110 / 101 / 100 / 011	Not possible to use radio IPs
010	CLK_ROOT_DIV
001	CLK_ROOT_DIV
000	CLK_ROOT_DIV

- ADC (STM32WL31xx/33xx only):
  - In parallel of the system clock, the ADC uses a fixed clock frequency (called CLKANA_ADC, CLKDIG_ADC) see *Figure 13: System clock details*.

Note: When the ADC is used, the system clock must run at minimum 8 MHz to be able to read the ADC sample before they are overloaded by a new sample.

Note: To avoid SNR degradation of the ADC, SMPS and ADC clocks must be synchronous.

6.2.1.1 Sleep mode management

Sleep mode management enables the capability to clock gate each peripherals bus clock independently during CPU sleeping mode (WFI).

Peripheral Sleep mode is configured, before WFI, thanks to three dedicated RCC registers (peripherals are split between AHB, APB0, APB1 clock domains):

• AHB Sleep Mode Enable register (see *AHB sleep mode enable register (RCC_AHBSMENR)*)

• APB0 Sleep Mode Enable register (see *APB0 sleep mode enable register (RCC_APB0SMENR)*)

• APB1 Sleep Mode Enable register (see *APB1 sleep mode enable register (RCC_APB1SMENR)*)

APB2 clock domain are not impacted by this feature.

Note: Some restrictions can be applied depending of each register fields, and further power consumption reduction is expected when all APB1 peripherals are in SLEEP mode.

Note: FLASH SLEEP mode is enabled only when both RCC and flash controller are configured correctly.

6.2.2 Slow clock frequency details

As explained at the beginning of the clock management sub-chapter, three different clock sources can be used for this slow clock tree:

- LSI: low speed low drift internal RC with a fixed frequency between 24 kHz and 49 kHz depending on the sample. It is called 32 kHz clock inside this document to simplify.
- LSE: 32.768 kHz low speed external crystal (or single-ended input frequency).

Note: If the external oscillator is used, the PB12/PB13 I/Os are automatically connected to this feature when RCC_CR.LSEON bit is set (GPIO_MODERX configuration is overloaded).

Caution: If APC bit is set, the user has to disable by software the PUB12/PUB13/PDB12/PDB13 bits on PB12/PB13 *I/O Port B pull-up control register (PWRC_PUCRB)* and *I/O Port B pull-down control register (PWRC_PDCRB)* to have the feature working fine. Otherwise if APC is reset, the pull-up, pull-down of PB12/13 are automatically configured.

- An always 32 kHz clock equal to CLK_16MHz divided by 512 (see *Figure 13*). In this case, the slow clock is not available in Deepstop low power mode.

Only one source at a time drives the whole low speed clock tree.

Note: By default after a PORESETn, all low speed sources are OFF. After a PADRESETn, the slow clock configuration is the one programmed before the PADRESETn.

The slow clock activation and selection stays relevant during the Deepstop low power mode and at wakeup as it clocks the timers involved in wakeup events generation.

Note: If LSI configuration is used, the software must measure the slow clock frequency to know the associated period that is used by the timers. A slow clock measurement feature is available in the MR_SUBG IP.

6.3 System frequency switch while MR_SUBG is used

When the radio is used on the device, the system clock frequency selection must respect some rules:

- The system clock frequency must be an integer multiple of 16 MHz equal or greater than the MR_SUBG frequency. Other options make the radio non functional.
- Changing the frequency of the system must be done through the RCC_CSCMDR register mechanism to avoid any risk to crash the radio scenarios.

This proper system frequency switch is managed through the collaboration of several blocks:

- the RCC (see *Clock switch command register (RCC_CSCMDR)*)
- the AHBUPCONV and the AHBDOWNCONV blocks (see *Section 3: AHB up/down converter*)

Using this safe mechanism, the software requests a system clock frequency change and is informed by the hardware when the new frequency is really in place through a status bit (see *Clock switch command register (RCC_CSCMDR)*) and an associated interrupt line on the CPU (see *Section 2.3.2: Interrupts*).

The software sequence is described in the *Section 6.8.3: Changing the system clock frequency while the MR_SUBGHz is enabled* sub-chapter.

6.4 Clock observation on external pad

It is possible to output some internal clocks on external pads:

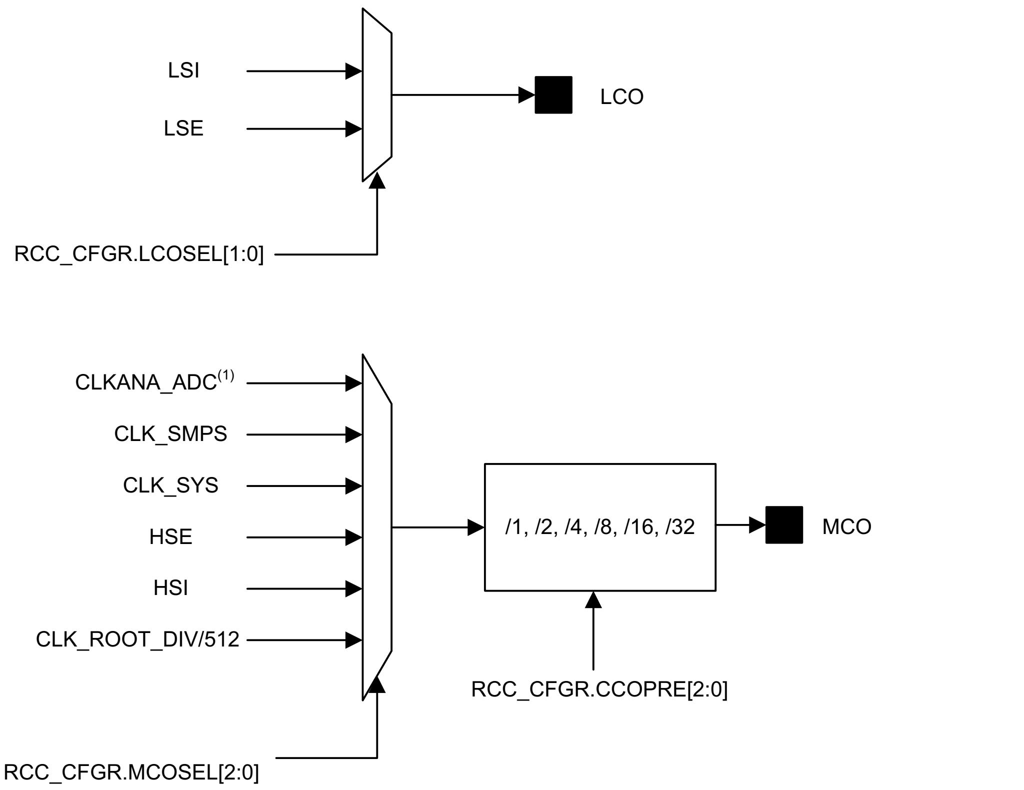
- the low speed clocks can be output on the LCO I/O,
- the high speed clocks can be output on the MCO I/O.

This is possible by programming the associated I/O in the good alternate function (see *Table 8: GPIO alternate options AF0 - AF3*).

The selection of the clock to output for each I/O is programmable through a RCC register (see *Section 6.6.2: Clock configuration register (RCC_CFGR)* for more details).

The *Figure 14* shows the possible configurations to output an internal clock.

Figure 14. LCO / MCO output clocks



1. ADC is available only on STM32WL31xx/33xx devices.

6.5 IO booster

Some analog switches are used to select the analog VINM/P pair input signals to be used by the ADC[a] or the COMP[b] or the DAC[b].

An IO booster block has been added to boost the voltage on the command of those analog switches when the VBAT goes below a threshold (2.7V) to guarantee the good behavior of those switches. This block has to be enabled by the software when needed through RCC_CFGR.IOBOOSTEN bit.

a. ADC is available only on STM32WL31xx/33xx devices.

b. Comparator and DAC are available only on STM32WL33xx devices.

6.6 RCC register descriptions

6.6.1 Clock source control register (RCC_CR)

This register controls the enable on the different clock sources (low and high speed).

Note: 1 The control bits linked to high speed clock source are reset on PADRESETn. The control bits linked to slow speed clock source are reset on PORESETn only (identified by the table footnote). As this register is in VDD12o power domain, its content is not modified after a wakeup from Deepstop and system clock is restored with configuration present before Deepstop mode entry.

2 Comparator, DAC, I2C2, LCSC, LCD, LPAWUR are available only on STM32WL33xx devices.
ADC, temperature sensor, I2C1, SPI1, I2S are available only on STM32WL31xx/33xx devices.
Any reference to these IPs on RCC registers is valid within this perimeter.

Address offset: 0x00

Reset value: 0x0000 1400

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HSIPLLRDY	HSIPLLON	HSEPLLBUFON	Res.	HSIRDY	LOCKDET_NSTOP[2:0]			LSEBYP	LSERDY	LSEON	LSIRDY	LSION	Res.	Res.
	r	rw	rw		r	rw	rw	rw	rw	r	rw	r	rw		

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **HSERDY**: External High Speed clock flag.
This bit is set by hardware to indicate that HSE oscillator (48 MHz or 50 MHz XO) is stable.
– 0: HSE oscillator is not ready.
– 1: HSE oscillator is ready.

Bit 16 **HSEON**: External High Speed clock enable.
The software has to set the bit to start the XO 48 MHz or 50 MHz and clear the bit to stop it.
– 0: HSE oscillator is OFF.
– 1: HSE oscillator is ON.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **HSIPLLRDY**: Internal High Speed clock PLL flag.
This bit is set by hardware to indicate that the RC64MPLL PLL is locked.
– 0: RC64MPLL PLL is unlocked.
– 1: RC64MPLL PLL is locked.

Bit 13 **HSIPLLON**: Internal High Speed clock PLL enable.
The software has to set the bit to request a RC64MPLL lock on HSE and clear the bit to stop it.
– 0: RC64MPLL PLL if OFF.
– 1: RC64MPLL PLL is ON.

Bit 12 **HSEPLLBUFON**: External high speed clock buffer for PLL RF enable.

The software has to set the bit when the radio is used (to have the RF PLL working).

– 0: HSE pll RF buffer is OFF.

– 1: HSE pll RF buffer is ON.

Warning: this bit must be set when the radio is used. The only reason to clear this bit would be to reduce power consumption in case the radio is not used on this device.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **HSIRDY**: Internal High Speed clock flag.

This bit is set by hardware to indicate that internal 64 MHz RC is stable.

This bit is activated only if the RC is enabled by HSION (it is not activated if the RC is enabled by an IP request).

– 0: internal 64 MHz RC is not ready.

– 1: internal 64 MHz RC is ready.

Bits 9:7 **LOCKDET_NSTOP**: define a time window target for the counter of the lock detector block in charge to manage the HSIPLLRDY information (PLL indicated as locked if the analog lock signal stays high and stable during this time window).

The formula to define the time window target is the following:

time window target = (LOCKDET_NSTOP + 1) x 64.

Bit 6(1) (2) **LSEBYP**: External Low Speed clock bypass.

This bit needs to be set when the slow clock is directly provided through SXTALI pin.

– 0: No LSE oscillator bypass.

– 1: LSE oscillator bypass is enabled.

Note: when this bit is set, ensure GPIOB_MODER[27:26] is not programmed as "11" and, if APC=1, PWRC_PUCRB[13] and PWRC_PDCRB[13] are not set.

Bit 5 (1) **LSERDY**: External Low Speed clock flag.

This bit is set by hardware to indicate that the slow clock has started.

– 0: LSE oscillator is not ready.

– 1: LSE oscillator is ready.

Note: this status bit is true whatever the chosen configuration (external 32 kHz oscillator -= LSEON or external clock provided on SXTALI = LSEBYP).

Bit 4(1) (2) **LSEON**: External Low Speed clock enable.

The software has to set the bit to start the XO 32 kHz and clear the bit to stop it.

– 0: LSE oscillator is OFF.

– 1: LSE oscillator is ON.

Bit 3(1) **LSIRDY**: Internal Low Speed clock flag.

This bit is set by hardware to indicate that internal low speed RC is stable.

– 0: internal low speed RC is not ready.

– 1: internal low speed RC is ready.

Bit 2(1) **LSION**: Internal low speed RC clock enable.

The software has to set the bit to start the internal slow clock RO and clear the bit to stop it.

– 0: LSI RC is OFF.

– 1: LSI RC is ON.

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

1. This bit is reset on PORESETn only.
2. The LSEBYP and LSEON bits must not be used at the same time. If the user decides to dynamically change the slow clock source between external XO and clock injection on SXTALI, he has to ensure both LSEON and LSEBYP are low at a time to reset the LSERDY flag.

6.6.2 Clock configuration register (RCC_CFGR)

Note: The control bits linked to high speed clock source are reset on PADRESETn. The control bits linked to slow speed clock source are reset on PORESETn only (identified by the table footnote).

Address offset: 0x08

Reset value: 0x0000 0240

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCOPRE[2:0]			MCOSEL[2:0]			LCOSEL[1:0]		SPI3I2SCLKSEL [1:0]		Res.	Res.	LCOEN	Res.	IOBOOSTEN	CLKSLOWSEL[1]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw		rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLKSLOWSEL[0]	Res.	LPUCLKSEL	SMPSDIV	Res.	CLKSYSDIV_STATUS[1:0]			CLKSYSDIV			Res.	HSESEL_STATUS	STOPHSI	HSESEL	Res.
rw		rw	rw		r	r	r	rw	rw	rw		r	rw	rw	

Bits 31:29 **CCOPRE**: Configurable Clock Output Prescaler.
- 000: CCO clock is divided by 1
- 001: CCO clock is divided by 2
- 010: CCO clock is divided by 4
- 011: CCO clock is divided by 8
- 100: CCO clock is divided by 16
- 101: CCO clock is divided by 32
- others: reserved

Note: Glitches propagation possible if CCOPRE[2:0] value is modified while MCO output is enabled on the IO.

Bits 28:26 **MCOSEL**: Main Configurable Clock Output Selection.
- 000: MCO output disabled. No clock on MCO pad.
- 001: system clock
- 010: reserved
- 011: RC64MPLL block output clock (can be internal RC 64 MHz or PLL output).
- 100: HSE (external 48 MHz or 50 MHz oscillator)
- 101: CLK_ROOT_DIV divided by 512 clock
- 110: SMPS clock
- 111: ADC clock

Note: Glitches propagation possible if MCOSEL[2:0] value is modified while MCO output is enabled on the IO.

Bits 25:24[1] **LCOSEL**: Low speed Configurable Clock Output Selection.
- 00: LCO output disabled. No clock on LCO pad.
- 01: not used
- 10: LSI (internal slow clock RC) clock
- 11: LSE (external 32 kHz)

Note: Glitches propagation possible if LCOSEL[1:0] value is modified while LCO output is enabled on the IO.

Bits 23:22 **SPI3I2SCLKSEL**: Selection of I2S clock for SPI3 IP.
– 00: CLK_ROOT/2 clock
– 01: CLK_ROOT_DIV clock (default)
– 1x: CLK_SYS
Note: the I2S clock frequency must be higher or equal to the system clock (configured through RCC_CFGR.CLKSYSDIV[2:0] bit field).

Bits 21:20 Reserved, must be kept at reset value.

Bits 19(1) **LCOEN**: LCO enable on PA10 also in Deepstop.
– 0: LCO output on PA10 is disabled
– 1: LCO output on PA10 is enabled.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **IOBOOSTEN**: IO BOOSTER enable (see *Section 6.5: IO booster* for details).
– 0: IO BOOSTER block is disabled
– 1: IO BOOSTER block is enabled.

Bits 16:15(1) **CLKSLOWSEL**: low speed clock source selection.
– 00: not used
– 01: LSE (external oscillator). This source can be kept during Deepstop mode.
– 10: LSI (internal RC). This source can be kept during Deepstop mode.
– 11: CLK_ROOT_DIV divided by 512
Note: No glitch mechanism has been added so glitches may appear on slow clock when the user changes its source.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **LPUCLKSEL**: Selection of LPUART clock
– 0: CLK_ROOT_DIV clock (default)
– 1: LSE clock

Bit 12 **SMPSDIV**: SMPS clock prescaling factor (when KRMEN=0)
– 0: SMPS clock is 8 MHz
– 1: SMPS clock is 4 MHz
Note: updating this field can cause issues if SMPS is running, it is recommended to first set SMPS in bypass mode (PWRC_CR5[9] SMPSBYP) before updating this value

Bit 11 Reserved, must be kept at reset value.

Bits 1110:8 **CLKSYSDIV_STATUS**: system clock frequency status
Set and cleared by hardware to indicate the actual system clock frequency. This register must be read to be sure that the new frequency, selected by CLKSYSDIV, has been applied.
– 000: CLK_SYS is CLK_ROOT
– 001: CLK_SYS is CLK_ROOT/2
– 010:CLK_SYS is CLK_ROOT/4 (HSESEL = 0) or CLK_ROOT/3 (HSESEL = 1)
– 011: CLK_SYS is CLK_ROOT/8 (HSESEL = 0) or CLK_ROOT/6 (HSESEL = 1)
– 100: CLK_SYS is CLK_ROOT/16 (HSESEL = 0) or CLK_ROOT/12 (HSESEL = 1)
– 101: CLK_SYS is CLK_ROOT/32 (HSESEL = 0) or CLK_ROOT/24 (HSESEL = 1)
– 110: CLK_SYS is CLK_ROOT/64 (HSESEL = 0) or CLK_ROOT/48 (HSESEL = 1)
– 111: not used.
The actual clock frequency switching can be delayed of up to 128 system clock cycles, depending on the RCC internal counter status at the moment the new CLKSYSDIV is applied

Bits 7:5 **CLKSYSDIV**: system clock frequency selection.
The division factor depends on RCC_CFGR.HSESEL
– 000: CLK_SYS is CLK_ROOT
– 001: CLK_SYS is CLK_ROOT/2
– 010:CLK_SYS is CLK_ROOT/4 (HSESEL = 0) or CLK_ROOT/3 (HSESEL = 1)
– 011: CLK_SYS is CLK_ROOT/8 (HSESEL = 0) or CLK_ROOT/6 (HSESEL = 1) *
– 100: CLK_SYS is CLK_ROOT/16 (HSESEL = 0) or CLK_ROOT/12 (HSESEL = 1) *
– 101: CLK_SYS is CLK_ROOT/32 (HSESEL = 0) or CLK_ROOT/24 (HSESEL = 1) *
– 110: CLK_SYS is CLK_ROOT/64 (HSESEL = 0) or CLK_ROOT/48 (HSESEL = 1) *
– 111: not used.
*: If RCC_APB2ENR.MRSUBGEN bit or RCC_APB2ENR.LPAWUREN bit are set, writing in CLKSYSDIV one of above values is replaced by 010b by hardware.
Warning:
– **if the software programs the CLSYSDIV= 000 while the RCC_CFGR.HSESEL=1, the system clock source is switched to PLL by hardware (and restart PLL analog block if RCC_CFGR.STOPHSI=1)**
– ***To switch CLKSYSDIV between 000 / 001 / 010 without risk when either the MR_SUBG or the LPAWUR is used, prefer the RCC_CSCMDR register to change the system frequency.***
– **The bus frequency of the radio IPs (MRSUBG, LPAWUR) must always be equal or less than the CPU/system clock to have functional radio and system clock must be an integer multiple of 16 MHz, this means that CLKSYSDIV=001 is forbidden when HSESEL=1.**

Bit 4 Reserved, must be kept at reset value.

Bit 3 **HSESEL_STATUS**: Clock source selection status.
– 0: RC64MPLL clock source is selected (default).
– 1: Direct HSE clock source is selected.

Bit 2 **STOPHSI**: RC64MPLL clock source stop request
– 0: RC64MPLL is enabled (default)
– 1: RC64MPLL disable requested.
Note: if the CLKSYSDIV (from RCC_CFGR or RCC_CSCMDR registers) is 000 the hardware automatically restart the RC64MPLL block and switch on the RC64MPLL clock source.

Bit 1 **HSESEL**: Clock source selection request.
– 0: RC64MPLL clock source is requested (default).
In this case, the fast clock tree is sourced by the RC64MPLL block. The clock can be either the HSI or the PLL if the HSI PLL is locked
– 1: Direct HSE clock source is requested.
In this case, the RC64MPLL block is not used and the maximum available frequency for the system clock tree is 48 MHz or 50 MHz.

Bit 0 Reserved, must be kept at reset value.

1. This bit is reset on PORESETn only.

6.6.3 Clock source software calibration register (RCC_CSSWCR)

This register allows overloading the trimming values loaded automatically by hardware with other values.

Note: The control bits linked to high speed clock source are reset on PADRESETn. The control bits linked to high speed clock source are reset on PORESETn only (identified by the table footnote).

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bits 29:24 **HSITRIMSW**: High speed clock trimming set by software.
This value is taken into account instead of the trimming value loaded by HW at reset if HSISWTRIMEN bit is set.
Note: when selected/active, this value must be modified either in JTAG mode (with CPU halted) or with CPU/system running on HSE clock tree

Bit 23 **HSISWTRIMEN**: High speed clock software trimming enable.
– 0: HW trimming value is used as trimming value on RC64MPLL block.
– 1: trimming value written in RCC_CSSWCR.HSITRIMSW[3:0] bit field is used as trimming value on RC64MPLL block.
Note: this bit must be modified either in JTAG mode (with CPU halted) or with CPU/system running on HSE clock tree

Bits 22:7 Reserved, must be kept at reset value.

Bits 6:5[1] **LSEDRV**: external 32 kHz crystal GM.
– 00: low drive capability
– 01: medium low drive capability
– 10: medium high drive capability
– 11: high drive capability

Bits 4:1[1] **LSISWBW**: Low speed internal RC trimming value set by software.
This value is taken into account instead of the trimming value loaded by HW at reset if LSISWTRIMEN bit is set.

Bit 0 [1] **LSISWTRIMEN**: Low speed internal RC software trimming enable.
– 0: HW trimming value is used as trimming value on the LSI.
– 1: trimming value written in RCC_CSSWCR.LSISWBW[3:0] bit field is used as trimming value on LSI.

1. This bit is reset on PORESETn only.

6.6.4 SMPS clock variable rate multiplier register (RCC_KRMR)

This register controls the enable and variable rate multiplier for SMPS clock.

This register is reset on PORESETn.

Address offset: 0x10

Reset value: 0x0000 0000

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

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

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

Bits 15:1 **KRM[4:0]**: SMPS clock dividing Ratio (CLK_SPMS_KRM frequency= CLK_ROOT frequency (depending on RCC_CFGR.HSESEL) divided by KRM when KRMEN=1)

- 0x00 to 0x08: SMPS clock frequency equals CLK_ROOT/8 (8.00 MHz / 6.00 MHz)
- 0x09: SMPS clock frequency equals CLK_ROOT/9 (7.11 MHz / 5.33 MHz)
- 0x0A: SMPS clock frequency equals CLK_ROOT/10 (6.40 MHz / 4.80 MHz)
- 0x0B: SMPS clock frequency equals CLK_ROOT/11 (5.82 MHz / 4.36 MHz)
- 0x0C: SMPS clock frequency equals CLK_ROOT/12 (5.33 MHz / 4.00 MHz)
- 0x0D: SMPS clock frequency equals CLK_ROOT/13 (4.92 MHz / 3.69 MHz)
- 0x0E: SMPS clock frequency equals CLK_ROOT/14 (4.57 MHz / 3.43 MHz)
- 0x0F: SMPS clock frequency equals CLK_ROOT/15 (4.27 MHz / 3.20 MHz)
- 0x10: SMPS clock frequency equals CLK_ROOT/16 (4.00 MHz / 3.00 MHz)
- 0x1x: Reserved

Note: SMPS clock frequency must be selected in a range [4-8] MHz (depending on RCC_KRMR.KRM and RCC_CFGR.HSESEL).

Bit 0 **KRMEN**: Variable rate multiplier

– 0: KRM is disabled (SMPS clock frequency depends **only** on RCC_CFGR.SMPSDIV)

– 1: KRM is enabled (SMPS clock frequency depends on source clock selection RCC_CFGR.HSESEL and KRM[4:0] dividing ratio)

Note: this bit must be 0 when SMPS is in precharge mode (PWRC_CR5. SMPSFBYP=1)

Note: before setting this bit ensure that PWRC_SR2.SMPSRDY=1

6.6.5 Clock interrupt enable register (RCC_CIER)

This register controls the enable on interrupt sources.

This register is reset on PADRESETn.

Address offset: 0x18

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	LCSCRSTIE	Res.	Res.	LCDRSTIE	LPURSTIE	WDGRSTIE	RTCRSTIE	HSIPLLUNLOCKDETIE	HSIPLLRDYIE	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSIRDYIE.
		rw			rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **LCSCRSTIE**: LCSC reset release interrupt enable.
- 0: LCSC reset release interrupt is disabled.
- 1: LCSC reset release interrupt is enabled.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **LCDRSTIE**: LCD reset release interrupt enable.
- 0: LCD reset release interrupt is disabled.
- 1: LCD reset release interrupt is enabled.

Bit 9 **LPURSTIE**: LPUART reset release interrupt enable.
- 0: LPUART reset release interrupt is disabled.
- 1: LPUART reset release interrupt is enabled.

Bit 8 **WDGRSTIE**: Watchdog reset release interrupt enable.
- 0: Watchdog reset release interrupt is disabled.
- 1: Watchdog reset release interrupt is enabled.

Bit 7 **RTCRSTIE**: RTC reset release interrupt enable.
- 0: RTC reset release interrupt is disabled.
- 1: RTC reset release interrupt is enabled.

Bit 6 **HSIPLLUNLOCKDETIE**: HSI PLL unlock detection interrupt enable.
- 0: HSI PLL unlocked detection interrupt is disabled.
- 1: HSI PLL unlocked detection is enabled.

Bit 5 **HSIPLLRDYIE**: HSI PLL ready interrupt enable.
- 0: HSI PLL locked interrupt is disabled.
- 1: HSI PLL locked interrupt is enabled.

Bit 4 **HSERDYIE**: HSE ready interrupt enable.
- 0: HSE ready interrupt is disabled.
- 1: HSE ready interrupt is enabled.

Bit 3 **HSIRDYIE**: HSI ready interrupt enable.
- 0: HSI ready interrupt is disabled.
- 1: HSI ready interrupt is enabled.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYIE**: LSE ready interrupt enable.
- 0: LSE ready interrupt is disabled.
- 1: LSE ready interrupt is enabled.

Bit 0 **LSIRDYIE**: LSI ready interrupt enable.
- 0: LSI ready interrupt is disabled.
- 1: LSI ready interrupt is enabled.

6.6.6 Clock Interrupt flag register (RCC_CIFR)

This register provides the status flag linked to clock source ready state or not. It is also used to clear the flags.

This register is reset on PADRESETn.

Address offset: 0x1C

Reset value: 0x0000 0008

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	LCSCRSTF	Res.	Res.	LCDRSTF	LPURSTF	WDGRSTF	RTCRSTF	HSIPLLUNLOCKDETF	HSIPLLRDYF	HSERDYF	HSIRDYF	Res.	LSERDYF	LSIRDYF.
		rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **LCSCRSTF**: LCSC reset release flag.

– 0: no LCSC reset release event occurred.

– 1: LCSC reset release event occurred.

Cleared by writing 1 in this bit.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the LCSC reset by writing in the RCC_APB0RSTR.LCSCRST, the reset release is effective only 2 slow clock periods after the APB writing. This interrupt allows informing the software when the reset release is really done.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **LCDRSTF**: LCD reset release flag.

– 0: no LCD reset release event occurred.

– 1: LCD reset release event occurred.

Cleared by writing 1 in this bit.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the LCD reset by writing in the RCC_APB0RSTR.LCDCRST, the reset release is effective only 2 slow clock periods after the APB writing. This interrupt allows informing the software when the reset release is really done.

Bit 9 **LPURSTF**: LPUART reset release flag.

– 0: no LPUART reset release event occurred.

– 1: LPUART reset release event occurred.

Cleared by writing 1 in this bit.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the LPUART reset by writing in the RCC_APB0RSTR.LPUARTRST, the reset release is effective only 2 slow clock periods or 2 CLK_ROOT_DIV clock periods after the APB writing, depending on how it's configured LPUCLKSEL bit in *Clock configuration register (RCC_CFGR) on page 118**. This interrupt allows informing the software when the reset release is really done.*

Bit 8 **WDGRSTF**: Watchdog reset release flag.
- 0: no Watchdog reset release event occurred.
- 1: Watchdog reset release event occurred.

Cleared by writing 1 in this bit.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the Watchdog reset by writing in the RCC_APB0RSTR.WDGRST, the reset release is effective only 2 slow clock periods after the APB writing. This interrupt allows informing the software when the reset release is really done.

Bit 7 **RTCRSTF**: RTC reset release flag.
- 0: no RTC reset release event occurred.
- 1: RTC reset release event occurred.

Cleared by writing 1 in this bit.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the RTC reset by writing in the RCC_APB0RSTR.RTCRST, the reset release is effective only 2 slow clock periods after the APB writing. This interrupt allows informing the software when the reset release is really done.

Bit 6 **HSIPLLUNLOCKDETF**: HSI PLL unlock detection flag.
- 0: no HSI PLL unlock event occurred.
- 1: HSI PLL unlock event occurred.

Cleared by writing 1 in this bit.

Bit 5 **HSIPLLRDYF**: HSI PLL ready flag.
- 0: no HSI PLL locked event occurred.
- 1: HSI PLL locked event occurred.

Cleared by writing 1 in this bit.

Bit 4 **HSERDYF**: HSE ready flag.
- 0: no HSE ready event occurred.
- 1: HSE ready event occurred.

Cleared by writing 1 in this bit.

Bit 3 **HSIRDYF**: HSI ready flag.
- 0: no HSI ready event occurred.
- 1: HSI ready event occurred.

Cleared by writing 1 in this bit.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYF**: LSE ready flag.
- 0: no LSE ready event occurred.
- 1: LSE ready event occurred.

Cleared by writing 1 in this bit.

Bit 0 **LSIRDYF**: LSI ready flag.
- 0: no LSI ready event occurred.
- 1: LSI ready event occurred.

Cleared by writing 1 in this bit.

6.6.7 Clock switch command register (RCC_CSCMDR)

This register allows safe CPU/system clock frequency switching while the MR_SUBG or the LPAWUR is active.

Requesting a frequency clock switch holds the AHB/APB transfers between the MR_SUBG or the LPAWUR and the rest of the system to execute safely the clock switching and release AHB / APB transfers as soon as the new frequency is in place.

A dedicated line of interrupt (instead of the RCC line) is used on the NVIC for the EOFSEQ_IRQ information (see *Table 7: Interrupt vectors*).

Note: Anyway, the user must keep the CPU/system frequency at minimum 16 MHz clock when the radio is used.

This register is reset on PADRESETn.

Address offset: 0x20

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOFSEQ_IRQ	EOFSEQ_IE	STATUS[1:0]		CLKSYSDIV_REQ[2:0]			REQUEST
								rc_w1	rw	r	r	rw	rw	rw	rw

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

Bit 7 **EOFSEQ_IRQ**: End of sequence flag.
- 0: No end of sequence event occurred.
- 1: End of sequence event occurred.

Cleared by writing 1 in this bit.

Bit 6 **EOFSEQ_IE**: End of sequence interrupt enable.
- 0: End of sequence interrupt is disabled.
- 1: End of sequence interrupt is enabled.

Bits 5:4 **STATUS**: Status of the switching sequence.

– 00: IDLE = no switch sequence requested / on-going.

– 01: ONGOING = a system clock frequency switch is on-going

– 10: DONE = a system clock frequency switch is done.

– 11: reserved.

This bit field is cleared when EOFSEQ_IRQ bit is cleared.

Bits 3:1 **CLKSYSDIV_REQ**: System clock requested/targeted frequency.

Same format and same notes/warnings as SYSCLKDIV[2:0] bit field described in *Clock configuration register (RCC_CFGR)*.

Bit 0 **REQUEST**: Request to switch the system clock frequency.

Write 1 in this bit to request a system clock frequency switch (using CLKSYSDIV_REQ[2:0] information).

This bit is cleared by hardware when the clock frequency switch is done.

Note: writing 0 in this bit aborts the frequency switch sequence if it is not yet finished. This action must not be used in the normal life of the application except if the end of sequence does not occurs after a long time (to unblock the situation) but this is not supposed to occur.

6.6.8 AHB0 macrocell reset register (RCC_AHBRSTR)

This register allows individual software reset of each IP located in the AHB0 mapping.

This register is reset on PADRESETn.

Address offset: 0x30

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOBRST	GPIOARST	Res.	DMARST
			rw									rw	rw		rw

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

Bit 20 **AESRST**: AES reset.
- 0: AES IP is not under reset.
- 1: AES IP is under reset.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **RNGRST**: RNG reset.
- 0: RNG IP is not under reset.
- 1: RNG IP is under reset.

Bit 17:16 Reserved, must be kept at reset value.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC reset.
- 0: CRC IP is not under reset.
- 1: CRC IP is under reset.

Bits 11:4 Reserved, must be kept at reset value.

Bit 3 **GPIOBRST**: IO controller for port B reset.
- 0: GPIOB IP is not under reset.
- 1: GPIOB IP is under reset.

Bit 2 **GPIOARST**: IO controller for port A reset.
- 0: GPIOA IP is not under reset.
- 1: GPIOA IP is under reset.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMARST**: DMA and DMAMUX reset.
- 0: DMA and DMAMUX IPs are not under reset.
- 1: DMA and DMAMUX IPs are under reset.

6.6.9 APB0 macrocell reset register (RCC_APB0RSTR)

This register allows individual software reset of each IP located in the APB0 mapping.

This register is reset on PADRESETn.

Note: Each bit is set and reset by the software.

Address offset: 0x34

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBGMCURST	WDGRST	LCSCRST	RTCRST	DACRST	COMPRST	LCDCRST	SYSCFGRST	Res.	Res.	Res.	Res.	Res.	Res.	TIM16RST	TIM2RST
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw

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

Bit 15 **DBGMCURST**: DBGMCU reset.
- 0: DBGMCU IP is not under reset.
- 1: DBGMCU IP is under reset.

Bit 14 **WDGRST**: Watchdog reset.
- 0: Watchdog IP is not under reset.
- 1: Watchdog IP is under reset.

Note: due to asynchronism slow clock/fast clock management, when the software requests to release the WDG reset by writing 0 in the RCC_APB0RSTR.WDGRST, the reset release is effective only 2 slow clock periods after the APB writing. An interrupt/status flag is available to inform the software when the reset release is really done (see RCC_CIFR/RCC_CIFR registers).

Bit 13 **LCSCRST**: LCSC reset.
- 0: LCSC IP is not under reset.
- 1: LCSC IP is under reset.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the LCSC reset by writing in the RCC_APB0RSTR.LCSCRST, the reset release is effective only 2 slow clock periods after the APB writing. An interrupt/status flag is available to inform the software when the reset release is really done (see RCC_CIFR/RCC_CIFR registers).

Bit 12 **RTCRST**: RTC reset.
- 0: RTC IP is not under reset.
- 1: RTC IP is under reset.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the RTC reset by writing in the RCC_APB0RSTR.RTCRST, the reset release is effective only 2 slow clock periods after the APB writing. An interrupt/status flag is available to inform the software when the reset release is really done (see RCC_CIFR/RCC_CIFR registers).

Bit 11 **DACRST**: DAC controller reset.
- 0: DAC controller IP is not under reset.
- 1: DAC controller IP is under reset.

Bit 10 **COMPRST**: Comparator controller reset.
- 0: Comp controller IP is not under reset.
- 1: Comp controller IP is under reset.

Bit 9 **LCDCRST**: LCD controller reset.
- 0: LCD controller IP is not under reset.
- 1: LCD controller IP is under reset.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the LCD reset by writing in the RCC_APB0RSTR.LCDCRST, the reset release is effective only 2 slow clock periods after the APB writing. An interrupt/status flag is available to inform the software when the reset release is really done (see RCC_CIFR/RCC_CIFR registers).

Bit 8 **SYSCFGRST**: system controller reset.
- 0: system controller IP is not under reset.
- 1: system controller IP is under reset.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **TIM16RST**: TIM16 reset.
- 0: TIM16 IP is not under reset.
- 1: TIM16 IP is under reset.

Bit 0 **TIM2RST**: TIM2 reset.
- 0: TIM2 IP is not under reset.
- 1: TIM2 IP is under reset.

6.6.10 APB1 macrocell reset register (RCC_APB1RSTR)

This register allows individual software reset of each IP located in the APB1 mapping.

This register is reset on PADRESETn.

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C2RST	Res.	I2C1RST	Res.	Res.	Res.	Res.	Res.
								rw		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI3RST	Res.	Res.	Res.	USARTRST	Res.	LPUARTRST	Res.	Res.	Res.	ADCRST	Res.	Res.	Res.	Res. SPI1RST
	rw				rw		rw				rw				rw

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

Bit 23 **I2C2RST**: I2C2 reset.
- 0: I2C2 IP is not under reset.
- 1: I2C2 IP is under reset.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **I2C1RST**: I2C1 reset.
- 0: I2C1 IP is not under reset.
- 1: I2C1 IP is under reset.

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI3RST**: SPI3 reset.
- 0: SPI3 IP is not under reset.
- 1: SPI3 IP is under reset.

Bit 13:11 Reserved, must be kept at reset value.

Bit 10 **USARTRST**: USART reset.
- 0: USART IP is not under reset.
- 1: USART IP is under reset.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **LPUARTRST**: LPUART reset.
- 0: LPUART IP is not under reset.
- 1: LPUART IP is under reset.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the LPUART reset by writing in the RCC_APB1RSTR.LPUARTRST, the reset release is effective only 2 slow clock periods after the APB writing. An interrupt/status flag is available to inform the software when the reset release is really done (see RCC_CIFR/RCC_CIFR registers).

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

Bit 4 **ADCRST**: ADC reset.
– 0: ADC IP is not under reset.
– 1: ADC IP is under reset.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SPI1RST**: SPI1 reset.
– 0: SPI1 IP is not under reset.
– 1: SPI1 IP is under reset.

6.6.11 APB2 macrocell reset register (RCC_APB2RSTR)

This register allows individual software reset of each IP located in the APB2 mapping (radio).

This register is reset on PADRESETn.

Address offset: 0x40

Reset value: 0x0000 0000

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

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

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

Bit 3 **LPAWURRST**: LPAWUR reset.
– 0: LPAWUR is not under reset.
– 1: LPAWUR is under reset.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **MRSUBGRST**: MR_SUBG (SubG radio) reset.
– 0: MR_SUBG IP is not under reset.
– 1: MR_SUBG IP is under reset.

6.6.12 AHB0 macrocell clock enable register (RCC_AHBENR)

This register allows individual software gating of the clock of each IP located in the AHB0 mapping.

Note: Each IP clock gating is controlled by only 1 bit which gates both AHB clock and kernel clock when the IP uses one.

This register is reset on PADRESETn.

Address offset: 0x50

Reset value: 0x0000 000C

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

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

Bit 20 **AESEN**: AES clock enable.
– 0: AES IP is clock gated.
– 1: AES IP is clocked.

Bits 19 Reserved, must be kept at reset value.

Bit 18 **RNGEN**: RNG clock enable.
– 0: RNG IP is clock gated.
– 1: RNG IP is clocked.

Bit 17:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC enable.
– 0: CRC IP is clock gated.
– 1: CRC IP is clocked.

Bits 11:4 Reserved, must be kept at reset value.

Bit 3 **GPIOBEN**: IO controller for port Benable.
– 0: GPIOB IP is clock gated.
– 1: GPIOB IP is clocked (default).

Bit 2 **GPIOAEN**: IO controller for port A enable.
– 0: GPIOA IP is clock gated.
– 1: GPIOA IP is clocked (default).

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMAEN**: DMA and DMAMUX enable.
– 0: DMA and DMAMUX IPs are clock gated.
– 1: DMA and DMAMUX IPs are clocked.

6.6.13 APB0 macrocell clock enable register (RCC_APB0ENR)

This register allows individual software gating of the clock of each IP located in the APB0 mapping.

Note: Each IP clock gating is controlled by only 1 bit which gates both APB clock and kernel clock when the IP uses one.

This register is reset on PADRESETn (except one bit identified with a table footnote).

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DBGMCUEN	WDGEN	LCSCEN	RTCEN	DACEN	COMPEN	LCDCEN	SYSCFGEN	Res.	Res.	Res.	Res.	Res.	Res.	TIM16EN	TIM2EN
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw

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

Bit 15 **DBGMCUEN**: DBGMCU enable.
- 0: DBGMCU IP is clock gated.
- 1: DBGMCU IP is clocked.

Bit 14 **WDGEN**: Watchdog enable.
- 0: Watchdog IP is clock gated.
- 1: Watchdog IP is clocked.

WARNING: the software has to wait 2 slow clock cycles before to use the IWDOG IP after setting this bit due to a double resynchronization on slow clock.

Bit 13 **LCSCEN**: LCSC enable.
- 0: LCSC IP is clock gated.
- 1: LCSC IP is clocked.

WARNING: the software has to wait 2 slow clock cycles before to use the LCSC IP after setting this bit due to a double resynchronization on slow clock.

Bit 12[(1)] **RTCEN**: RTC enable.
- 0: RTC IP is clock gated.
- 1: RTC IP is clocked.

WARNING: the software has to wait 2 slow clock cycles before to use the RTC IP after setting this bit due to a double resynchronization on slow clock.

Bit 11 **DACEN**: DAC controller enable.
- 0: DAC controller is clock gated.
- 1: DAC controller is clocked.

Bit 10 **COMPEN**: Comparator controller enable.
- 0: COMP controller is clock gated.
- 1: COMP controller is clocked.

Bit 8 **SYSCFGEN**: system controller enable.
- 0: system controller IP is clock gated.
- 1: system controller IP is clocked.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **TIM16EN**: TIM16 enable.
- 0: TIM16 IP is clock gated.
- 1: TIM16 IP is clocked.

Bit 0 **TIM2EN**: TIM2 enable.
- 0: TIM2 IP is clock gated.
- 1: TIM2 IP is clocked.

1. This bit is reset on PORESETn only.

6.6.14 APB1 macrocell clock enable register (RCC_APB1ENR)

This register allows individual software gating of the clock of each IP located in the APB1 mapping.

Note: Each IP clock gating is controlled by only 1 bit which gates both APB clock and kernel clock when the IP uses one.

This register is reset on PADRESETn.

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C2EN	Res.	I2C1EN	Res.	Res.	Res.	Res.	Res.
								rw		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI3EN	Res.	Res.	Res.	USARTEN	Res.	LPUARTEN	Res.	Res.	ADCANAEN	ADCDIGEN	Res.	Res.	Res.	SPI1EN
	rw				rw		rw			rw	rw				rw

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

Bit 23 **I2C2EN**: I2C2 enable.
- 0: I2C21 IP is clock gated.
- 1: I2C2 IP is clocked.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN**: I2C1 enable.
- 0: I2C1 IP is clock gated.
- 1: I2C1 IP is clocked.

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI3EN:** SPI3 enable.
- 0: SPI3 IP is clock gated.
- 1: SPI3 IP is clocked.

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **USARTEN**: USART enable.
- 0: USART IP is clock gated.
- 1: USART IP is clocked.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **LPUARTEN** LPUART enable.
- 0: LPUART IP is clock gated.
- 1: LPUART IP is clocked.

WARNING: the software has to wait 2 slow clock cycles before to use the LPUART IP after setting this bit due to a double resynchronization on slow clock.

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

Bit 5 **ADCANAEN** ADC clock enable for the analog part of the ADC block.
- 0: ADC analog IP is clock gated.
- 1: ADC analog IP is clocked.

Bit 4 **ADCDIGEN** ADC clock enable for digital part of the ADC block.
- 0: ADC digital IP is clock gated.
- 1: ADC digital IP is clocked.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SPI1EN**: SPI1 enable.
- 0: SPI1 IP is clock gated.
- 1: SPI1 IP is clocked.

6.6.15 APB2 macrocell clock enable register (RCC_APB2ENR)

This register allows software gating of the clock of each IP located in the APB2 mapping (radio).

Note: Gating the MR_SUBG or LPAWUR clocks mean that both their fast and slow clocks are gated (so including their WAKEUP block clocks).Gating the MR_SUBG clock means that the fast and slow clocks are gated (so including its WAKEUP block clock).

This register is reset on PADRESETn.

Address offset: 0x60

Reset value: 0x0000 0000

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

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

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

Bit 3 **LPAWUREN**: LPAWUR enable.
- 0: LPAWUR is clock gated.
- 1: LPAWUR is clocked.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **MRSUBGEN**: MR_SUBG enable.
- 0: MR_SUBG IP is clock gated.
- 1: MR_SUBG IP is clocked.

6.6.16 V33 reset status register (RCC_CSR)

This register provides the reset reason flags. It is set automatically by hardware on any new reset event and must be cleared by software.

The *Figure 12: Reset generation* provides a summary of active flags versus reset reason.

This register is reset on PORESETn.

Address offset: 0x94

Reset value: 0x0C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LOCKUPRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	r	r	r	r	r			rc_w1							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LOCKUPRSTF**: CPU lockup reset flag.
Set by the hardware when a CPU lockup reset occurs.
Reset by writing 1 in RMVF bit.

Bit 29 **WDGRSTF**: Watchdog reset flag.
Set by the hardware when a Watchdog reset occurs.
Reset by writing 1 in RMVF bit.

Bit 28 **SFTRSTF**: Software reset flag.
Set by the hardware when a CPU system reset occurs.
Reset by writing 1 in RMVF bit.

Bit 27 **PORRSTF**: Power-On reset flag.
Set by the hardware when a PORESETN or a BOR reset occurs.
Reset by writing 1 in RMVF bit.

Bit 26 **PADRSTF**: NRSTn pad reset flag.
Set by the hardware when a reset from external NRSTn pad occurs but also after any reset.
This means the source of the reset is the NRSTn pad only if all flags are low except this one.
Reset by writing 1 in RMVF bit.

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: Remove Flag reset.
Writing 1 in this bit clears all the reset flags of this register.
This bit is auto-cleared by the hardware.

Bits 22:0 Reserved, must be kept at reset value.

6.6.17 RF software high-speed external register (RCC_RFSWHSECR)

This register is reset on PADRESETn.

Address offset: 0x98

Reset value: 0x0000 803F

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **AMPLTHRESH**: RF-HSE Amplitude Control threshold.
Set by software

Bits 15:14 **ISTARTUP**: RF-HSE Startup current
Set by software

Bits 13:8 **SWXOTUNE**: RF HSE capacitor bank tuning value set by software.
This value is taken into account instead of the trimming value loaded by HW at reset if SWXOTUNEEN bit is set.

Bit 7 **SWXOTUNEEN**: RF HSE software capacitor bank tuning enable.
– 0: trimming value readable in RCC_RFHSECR.XOTUNE[5:0] bit field is used as trimming value on HSE.
– 1: trimming value written in RCC_RFSWHSECR.SWXOTUNE[5:0] bit field is used as trimming value on HSE.

Bits 6:5 **GMC[6:5]**: High speed external XO current control reference
- 00: 10 µA
- 01: 20 µA
- 1x: 40 µA
Note: this value is set only by software.

Bits 4:0 **GMC[4:0]**: High speed external XO current control multiplying factor
IcoreHSE= GMC[4:0] * GMC[6:5]

Example: GMC[6:0]=0x1111001 -> IcoreHSE=25*40uA
Note: this value is set only by software.

6.6.18 RF high speed external register (RCC_RFHSECR)

This register is reset on PADRESETn.

Address offset: 0x9C

Reset value: 0x0000 0000 when the STM32WL30xx/31xx/33xx device flash is empty, else depends on trimmed values flashed in the sample.

Note: The XOTUNE value depends on the choice of the external XO component. For this reason, the RCC_RFSWHSECR.SWXOTUNE bit field is the one to program and select before to start the XO.

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

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **AMPLREADY**: RF-HSE Amplitude Control Ready output

Bits 5:0 **XOTUNE**: RF-HSE capacitor bank tuning.
This value is loaded by HW at reset as soon as the flash controller achieves the reading of the information in flash memory.

6.6.19 AHB sleep mode enable register (RCC_AHBSMENR)

This register is reset on PADRESETn.

Address offset: 0xA0

Reset value: 0x0014 160F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCSMEN	Res.	SRAM1SMEN	SRAM0SMEN	Res.	Res.	Res.	Res.	Res.	GPIOBSMEN	GPIOASMEN	FLASHSMEN	DMASMEN
			rw		rw	rw						rw	rw	rw	rw

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

Bit 20 **AESSMEN:** AES clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: AES clock disabled in Sleep mode
- 1: AES clock enabled in Sleep mode (if enabled in AESEN)

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **RNGSMEN:** RNG bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: RNG bus clock disabled in Sleep mode
- 1: RNG bus clock enabled in Sleep mode (if enabled in RNGEN)

Bits 17:13 Reserved, must be kept at reset value.

Bit 12 **CRCSMEN:** CRC clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: CRC clock disabled in Sleep mode
- 1: CRC clock enabled in Sleep mode (if enabled in CRCEN)

Bit 11 Reserved, must be kept at reset value.

Bit 10 **SRAM1SMEN:** SRAM1 clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: SRAM1 clock disabled in Sleep mode
- 1: SRAM1 clock enabled in Sleep mode

Bit 9 **SRAM0SMEN:** SRAM0 clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: SRAM0 clock disabled in Sleep mode
- 1: SRAM0 clock enabled in Sleep mode

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

Bit 3 **GPIOASMEN**: GPIOB clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: GPIOB clock disabled in Sleep mode
- 1: GPIOB clock enabled in Sleep mode (if enabled by GPIOBEN)

Bit 2 **GPIOASMEN**: GPIOA clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: GPIOA clock disabled in Sleep mode
- 1: GPIOA clock enabled in Sleep mode (if enabled by GPIOAEN)

Bit 1 **FLASHSMEN**: Flash clocks enable during flash Sleep mode and CPU WFI bit
This bit is set and reset by software.
- 0: flash clocks are disabled in flash Sleep mode and CPU WFI mode
- 1: flash clocks are enabled in Sleep mode

Note: Flash Sleep mode is enabled through flash controller register CONFIG.SLEEP_SM (see Configuration register (CONFIG))

Bits 0 **DMASMEN**: DMA clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: DMA clock disabled in Sleep mode
- 1: DMA clock enabled in Sleep mode (if enabled in DMAEN)

6.6.20 APB0 sleep mode enable register (RCC_APB0SMENR)

This register is reset on PADRESETn.

Address offset: 0xA4

Reset value: 0x0000 FF03

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBGMCUSMEN	WDGSMEN	LCSCSMEN	RTCSMEN	DACSMEN	COMPSMEN	LCDCSMEN	SYSCFGSMEN	Res.	Res.	Res.	Res.	Res.	Res.	TIM16SMEN	TIM2SMEN
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw

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

Bit 15 **DBGMCUSMEN**: DBGMCU clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: DBGMCU clock disabled in Sleep mode
- 1: DBGMCU clock enabled in Sleep mode (if enabled in DBGMCUEN)

Bit 14 **WDGSMEN**: WDG bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: WDG bus clock disabled in Sleep mode
- 1: WDG bus clock enabled in Sleep mode (if enabled in WDGEN)

Bit 13 **LCSCSMEN**: LCSC bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: LCSC bus clock disabled in Sleep mode
- 1: LCSC bus clock enabled in Sleep mode (if enabled in LCSCEN)

Bit 12 **RTCSMEN**: RTC bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: RTC bus clock disabled in Sleep mode
- 1: RTC bus clock enabled in Sleep mode (if enabled in RTCEN)

Bit 11 **DACSMEN**: DAC clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: DAC clock disabled in Sleep mode
- 1: DAC clock enabled in Sleep mode (if enabled in DACEN)

Bit 10 **COMPSMEN**: COMP clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: COMP clock disabled in Sleep mode
- 1: COMP clock enabled in Sleep mode (if enabled in COMPEN)

Bit 9 **LCDCSMEN**: LCDC bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: LCDC bus clock disabled in Sleep mode
- 1: LCDC bus clock enabled in Sleep mode (if enabled in LCDCEN)

Bit 8 **SYSCFGSMEN**: SYSCFG bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: SYSCFG bus clock disabled in Sleep mode
- 1: SYSCFG bus clock enabled in Sleep mode (if enabled in SYSCFGEN)

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **TIM16SMEN**: TIM16 bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: TIM16 bus clock disabled in Sleep mode
- 1: TIM16 bus clock enabled in Sleep mode (if enabled in TIM16EN)

Bit 0 **TIM2SMEN**: TIM2 bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: TIM2 bus clock disabled in Sleep mode
- 1: TIM2 bus clock enabled in Sleep mode (if enabled in TIM2EN)

6.6.21 APB1 sleep mode enable register (RCC_APB1SMENR)

This register is reset on PADRESETn.

Address offset: 0xA8

Reset value: 0x00A0 4511

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C2SMEN	Res.	I2C1SMEN	Res.	Res.	Res.	Res.	Res.
								rw		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI3SMEN	Res.	Res.	Res.	USARTSMEN	Res.	LPUARTSMEN	Res.	Res.	Res.	ADCDIGSMEN	Res.	Res.	Res.	SPI1SMEN
	rw				rw		rw				rw				rw

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

Bit 23 **I2C2SMEN**: I2C2 bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: I2C2 bus clock disabled in Sleep mode
- 1: I2C2 bus clock enabled in Sleep mode (if enabled in I2C2EN)

Bit 22 Reserved, must be kept at reset value.

Bit 21 **I2C1SMEN**: 2C1 bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: I2C1 bus clock disabled in Sleep mode
- 1: I2C1 bus clock enabled in Sleep mode (if enabled in I2C1EN)

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI3SMEN:** SPI3 bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: SPI3 bus clock disabled in Sleep mode
- 1: SPI3 bus clock enabled in Sleep mode (if enabled in SPI3EN)

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **USARTSMEN**: USART bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: USART bus clock disabled in Sleep mode
- 1: USART bus clock enabled in Sleep mode (if enabled in USARTEN)

Bit 9 Reserved, must be kept at reset value.

Bit 8 **LPUARTSMEN**: LPUART bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: LPUART bus clock disabled in Sleep mode
- 1: LPUART bus clock enabled in Sleep mode (if enabled in LPUARTEN)

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

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ADCDIGSMEN:** ADCDIG bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: ADCDIG bus clock disabled in Sleep mode
- 1: ADCDIG bus clock enabled in Sleep mode (if enabled by ADCDIGEN)

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SPI1SMEN**: SPI1 bus clock enable during Sleep mode bit
This bit is set and reset by software.
- 0: SPI1 bus clock disabled in Sleep mode
- 1: SPI1 bus clock enabled in Sleep mode (if enabled in SPI1EN)

6.7 RCC register map

Refer to *Table 2.2.2: Memory map and register boundary addresses* for the RCC base address location in the STM32WL30xx/31xx/33xx.

The green cells indicate that the register is in the VDD12o power domain, and the salmon cells indicate that the register is in the VDDIO (V33) power domain. This implies that those registers are not reset on Deepstop exit.

Table 17. RCC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERDY	HSEON	Res.	HSIPLLRDY	HSIPLLON	HSEPLLBUFON	Res.	HSIRDY		LOCKDET_NSTOP		LSEBYP	LSERDY	LSEON	LSIRDY	LSION	Res.	Res.
	Reset value															0	0		0	0	1		1	0	0	0	0	0	0	0	0		
0x04	Reserved																																
0x08	RCC_CFGR		CCOPRE			MCOSEL		LCOSEL		SPI3I2SCLKSEL		Res.	Res.	LCOEN	Res.	IOBOOSTEN	CLKSLOWSEL		Res.	LPUCLKSEL	SMPSDIV	Res.		CLKSYSDIV_STATUS			CLKSYSDIV		Res.	HSESEL_STATUS	STOPHSI	HSESEL	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0		0	0	0		0	0		0	1	0	0	1	0		0	0	0	
0x0C	RCC_CSSWCR	Res.	Res.				HSITRIMSW[5:0]			HSISWTRIMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSEDRV			LSISWBW[3:0]			LSISWTRIMEN
	Reset value			0	0	0	0	0	0	0																	0	0	0	0	0	0	0
0x10	RCC_KRMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			KRM[4:0]			KRMEN
	Reset value																											0	0	0	0	0	0
0C0x14	Reserved																																
0x18	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCSCRSTIE	Res.	Res.	LCDRSTIE	LPURSTIE	WDGRSTIE	RTCRSTIE	HSIPLLUNLOCKDETIE	HSIPLLRDYIE	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSIRDYIE
	Reset value																			0			0	0	0	0	0	0	0	0		0	0

Table 17. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCDRSTF	LPURSTF	WDGRSTF	RTCRSTF	HSIPLLUNLOCKDETF	HSIPLLRDYF	HSERDYF	HSIRDYF		LSERDYF	LSIRDYF
	Reset value																						0	0	0	0	0	0	0	1		0	0
0x20	RCC_CSCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOFSEQ_IRQ	EOFSEQ_IE	STATUS[1:0]		CLKSYSDIV_REQ[2:0]			REQUEST
	Reset value																									1	0	0	0	0	0	0	0
0x24 - 0x2C	Reserved																																
0x30	RCC_AHBRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AESRST	Res.	RNGRST	Res.	Res.	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOBRST	GPIOARST	Res.	DMARST
	Reset value												0		0						0									0	0		0
0x34	RCC_APB0RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGMCURS	WDGRST	LCSCRST	RTCRST	DACRST	COMPRST	LCDCRST	SYSCFGRST	Res.	Res.	Res.	Res.	Res.	Res.	TIM16RST	TIM2RST
	Reset value																	0	0	0	0	0	0	0	0							0	0
0x38	RCC_APB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C2RST	Res.	I2C1RST	Res.	Res.	Res.	Res.	Res.	Res.	SPI3RST	Res.	Res.	Res.	USARTRST	Res.	LPUARTRST	Res.	Res.	Res.	ADCRST	Res.	Res.	Res.	SPI1RST
	Reset value									0		0							0				0		0				0				0
0x3C	Reserved																																
0x40	RCC_APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPAWURRST	Res.	Res.	MRSUBGRST
	Reset value																													0			0
0x44 - 0x4C	Reserved																																
0x50	RCC_AHBENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AESEN	Res.	RNGEN	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOBEN	GPIOAEN	Res.	DMAEN
	Reset value												1		1						1									1	1		1

Table 17. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54	RCC_APB0ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGMCUEN	WDGEN	LCSCEN	RTCEN	DACEN	COMPEN	LCDCEN	SYSCFGEN	Res.	Res.	Res.	Res.	Res.	Res.	TIM16EN	TIM2EN
	Reset value																	0	0	0	0	0	0	0	0							0	0
0x58	RCC_APB1ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C2EN	Res.	I2C1EN	Res.	Res.	Res.	Res.	Res.	Res.	SPI3EN	Res.	Res.	Res.	USARTEN	Res.	LPUARTEN	Res.	Res.	ADCANAEN	ADCDIGEN	Res.	Res.	Res.	SPI1EN
	Reset value									0		0							0				0		0			0	0				0
0x5C	Reserved																																
0x60	RCC_APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPAWUREN	Res.	Res.	MRSUBGEN
	Reset value																													0			0
0x6464 - 0x90	Reserved																																
0x94	RCC_CSR	Res.	LOCKUPRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0	0	0	1	1			0																							
0x98	RCC_RFSWHSECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	AMPTHRESH			ISTARTUP		Res.	Res.	Res.	Res.	Res.	Res.	Res.	GMC						
	Reset value									0					0	0	0	1	0								0	1	1	1	1	1	1
0x9C	RCC_RFHSECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AMPLREADY.	XOTUNE					
	Reset value																										-	-	-	-	-	-	-
0xA0	RCC_AHBSMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AESSMEN	Res.	RNGSMEN	Res.	Res.	Res.	Res.	Res.	CRCSMEN	Res.	SRAM1SMEN	SRAM0SMEN	Res.	Res.	Res.	Res.	Res.	GPIOBSMEN	GPIOASMEN	FLASHSMEN	DMASMEN
	Reset value												1		1						1		1	1						1	1	1	1
0xA4	RCC_APB0SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGMCUSMEN	WDGSMEN	SCSMIEN	RTCSMEN	DACSMEN	COMPSMEN	LCDCSMEN	SYSCFGSMEN	Res.	Res.	Res.	Res.	Res.	Res.	TIM16SMEN	TIM2SMEN
	Reset value																	1	1	1	1	1	1	1	1							1	1

Table 17. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA8	RCC_APB1SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C2SMEN	Res.	I2C1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	SPI3SMEN	Res.	Res.	Res.	USARTSMEN	Res.	LPUARTSMEN	Res.	Res.	Res.	ADCDIGSMEN	Res.	Res.	Res.	SPI1SMEN
	Reset value									1		1							1				1		1				1				1

6.8 Programmer's model

6.8.1 Switch the system on the PLL64M clock tree

To switch the system from the HSI clock source to the PLL clock source, the user has to:

1. Enable the HSE (48 MHz or 50 MHz external crystal)
2. Wait for the HSE ready flag information (through interrupt or by polling)
3. Request to enable the PLL
4. Wait for the PLL ready flag information (through interrupt or by polling). From this point, the clock source for the whole fast clock tree is the accurate PLL source.

Note: A status flag and an associated interrupt are available to inform the software in case of HSIPLL64M unlock event. See RCC_CIER and RCC_CIFR registers.

6.8.2 Use the direct HSE instead of the RC64MPLL block

If the application does not target to use the maximum system clock frequency, the system can be configured to use directly the 48 MHz or 50 MHz provided by the external XO (HSE).

This configuration choice is supposed to be static and to be used when maximum frequency is never used.

In this case, the software has to:

1. Ensure the RCC_CR.CLKSYSDIV bit field is programmed with a value greater then 000
2. Enable the external XO (by setting the RCC_CR.HSEON if not yet done)
3. Wait for the HSE ready flag information (through interrupt or by polling)
4. Set the RCC_CFGR.HSESEL bit to switch the fast clock tree on HSE path. If both clocks (HSI and HSE) are present, the switch should take around 4 clock cycles.
5. To safe power, the software can stop the RC64MPLL analog block by setting the RCC_CFGR.STOPHSI bit.

Note: A hardware mechanism is in place to restart the RC64MPLL and switch back the clock tree on it if the HSERDY is low or if the CLKSYSDIV bit field has been programmed to 000.

The HSE configuration is not lost on a Deepstop sequence. So at wakeup, the system restarts the HSE (thanks to HEON bit) and clock tree switches back on HSE as soon as the HSERDY flag is set. In the meantime, the clock tree run on RC64MPLL block. If the STOPHSI was high before the Deepstop, the RC64MPLL analog is switched off as soon as the clock tree is back on HSE path.

- The HSE configuration is not lost on a Deepstop sequence. So at wakeup, the system restarts the HSE (thanks to HEON bit). So at wakeup:the clock tree runs on RC64MPLL block (HSI or PLL)
- the clock is automatically switched from HSI to HSE

6.8.3 Changing the system clock frequency while the MR_SUBGHz is enabled

As long as the MR_SUBG is enabled (by setting the RCC_APB2ENR.MRBLEEN), the application software has no guarantee the radio is running or about to start a sequence that makes the MR_SUBG IP does AHB access to the RAM. Changing the system clock and by this action changing the ratio between MR_SUBG clock domain and system clock domain could create crash if not managed carefully.

For this reason, a hardware mechanism has been put in place and must be used to change the system frequency when MR_SUBG is ON.

The sequence to execute to change the system clock is the following:

1. Ensure the targeted frequency is a integr multiple of 16 MHz greater than or equal to the MR_SUBG frequency (visible in RCC_APB2ENR.CLKBLEDIV)
2. Program the wanted frequency in the RCC_CSCMDR.CLKSYDIV bit field and set the RCC_CSCMDR.REQUEST bit.
3. Wait for the RCC_CSCMDR.EOFSEQ_IRQ flag information (through interrupt or by polling)
4. When the flag (and the interrupt if enabled) is set, the system is running on the new frequency.
5. The RCC_CFGR.CLKSYSDIV[1:0] bit field has been updated by hardware to the new frequency value.

Note: If the software requested a frequency below 16 MHz, the actual final frequency is 16 MHz (associated value is readable in the RCC_CFGR.CLKSYSDIV[1:0].

7 General-purpose I/Os (GPIO)

7.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR) and a 32-bit set/reset register (GPIOx_BSRR). In addition all GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

7.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every clock cycle
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

7.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the *Section 4: I/O operating modes*, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

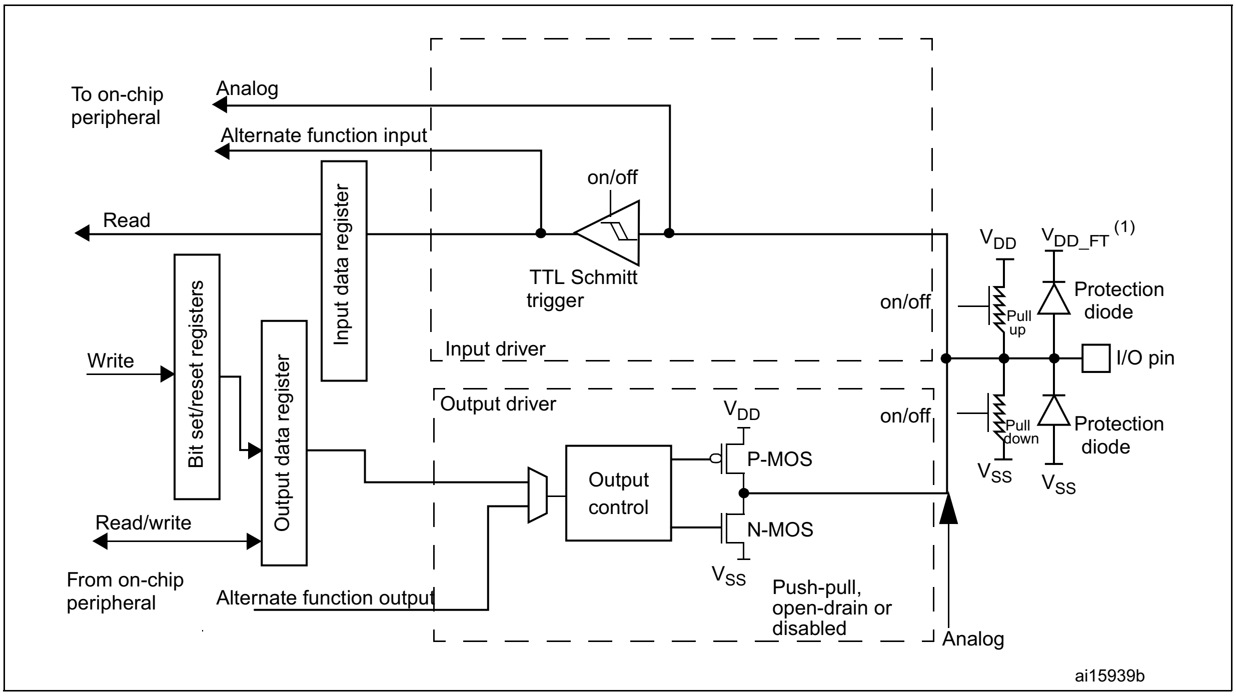
- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Note: Open-drain and analog features are not available on all the I/Os of the STM32WL30xx/31xx/33xx. *Refer to Table 8: GPIO alternate options AF0 - AF3 and Table 9: GPIO alternate options AF4 - AF6 footnotes.*

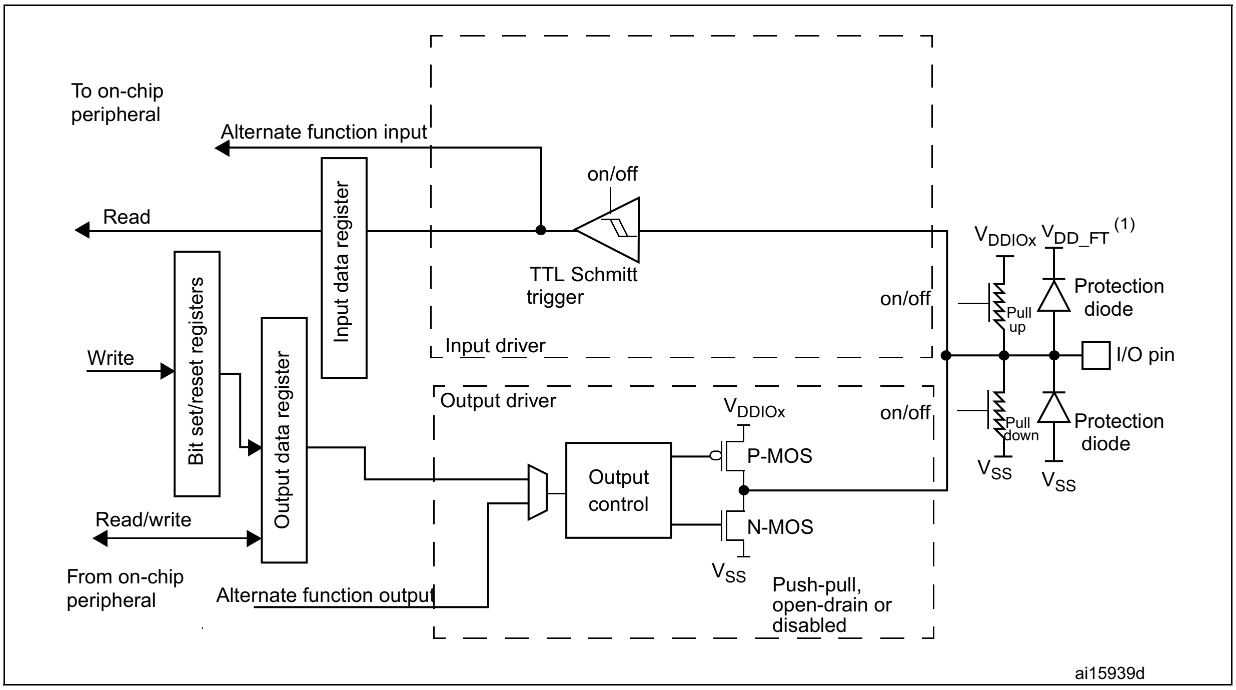
Figure 15 and *Figure 16* show the basic structures of a mixed analog/digital 5V tolerant I/O port bit and a digital only 5V tolerant, respectively. *Table 18* gives the possible port bit configurations.

Figure 15. Basic structure of a mixed analog/digital five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Figure 16. Basic structure of a digital only five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Table 18. Port bit configuration table(1)

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]	PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]	0	0	AF	PP
	0		0	1	AF	PP + PU
	0		1	0	AF	PP + PD
	0		1	1	Reserved	
	1		0	0	AF	OD
	1		0	1	AF	OD + PU
	1		1	0	AF	OD + PD
	1		1	1	Reserved	

Table 18. Port bit configuration table(1) (continued)

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

Note: Open-drain and analog features are not available on all the I/Os of the STM32WL30xx/31xx/33xx. Refer to Table 8: GPIO alternate options AF0 - AF3 and Table 9: GPIO alternate options AF4 - AF6 footnotes.

7.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in GPIO input pull-up mode except the SWD debug pins.

The debug pins are in AF0 pull-up/pull-down after reset:

- PA2: SWDIO in pull-up
- PA3: SWDCLK in pull-down

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

7.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to seven alternate function inputs (AF0 to AF6) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the *Section 4: I/O operating modes*.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:

- **Debug function:** after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **System function:** MCO and LCO pins have to be configured in alternate function mode.
- **GPIO:** configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Alternate function:**
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Cortex-M0+ alternate function (EVENTOUT):** The Cortex®-M0+ output EVENTOUT signal can be output as alternate function on several I/Os. An event can be signaled through the configured pin after executing SEV instruction.
- **Additional functions:**
  - For the ADC (STM32WL33xx/31xx only), configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADCregisters.

Note: The user has to disable the pulls by software on I/Os he configures in analog mode.

  - For the additional functions like Wakeup I/Os, LSE oscillator, LCO and LCSC (STM32WL33xx only) configure the required function in the related PWRC and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the for the detailed mapping of the alternate function I/O pins.

7.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

7.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

See *GPIOA port input data register (GPIOA_IDR)* and *GPIOA port output data register (GPIOA_ODR)* for the register descriptions.

7.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, bit BS(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BR(i) **resets** the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

7.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence (refer to *GPIOB port bit set/reset register (GPIOB_BSRR)*) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in *GPIOB port bit set/reset register (GPIOB_BSRR)*.

7.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, you can connect an alternate function to some other pin as required by your application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the *Section 4: I/O operating modes*.

7.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the IO port must be configured in input mode. The interruption configuration (level/edge, polarity, mask)has to be done in the system controller (SYSCFG). See *Section 8: System controller (SYSCFG)*.

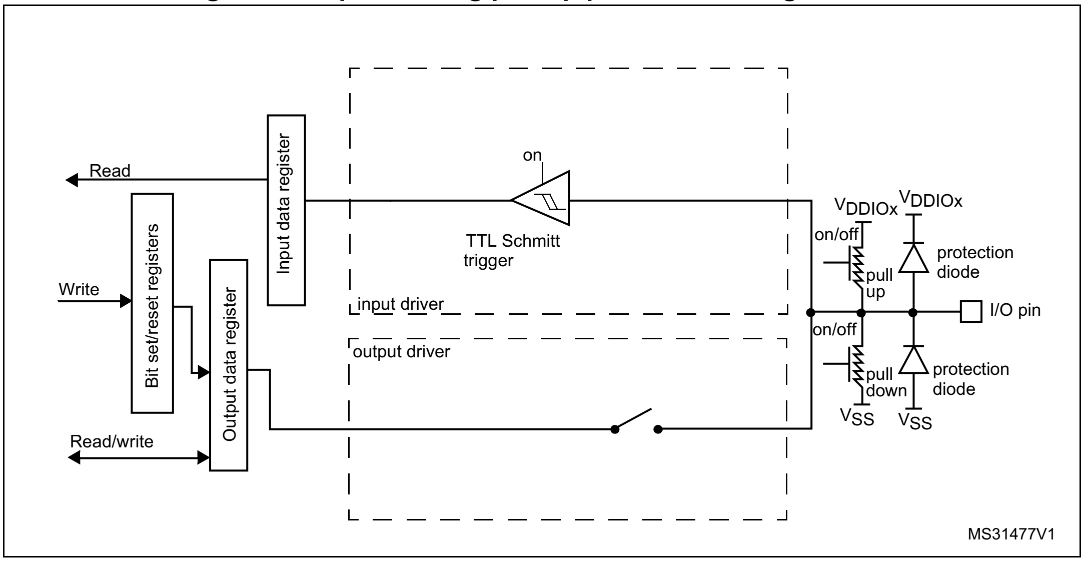
7.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

Figure 17 shows the input configuration of the I/O port bit.

Figure 17. Input floating/pull up/pull down configurations



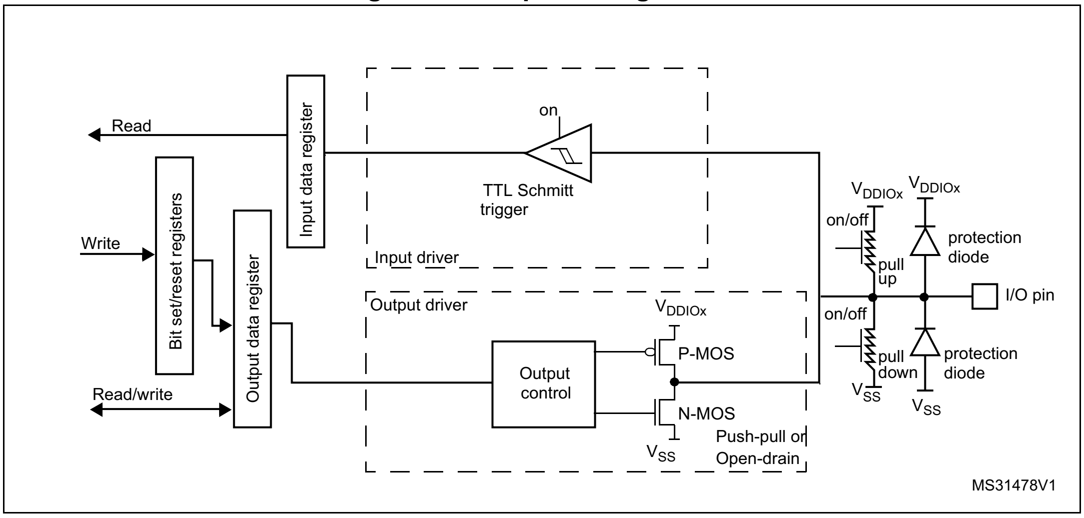
7.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  – Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  – Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

Figure 18 shows the output configuration of the I/O port bit.

Figure 18. Output configuration



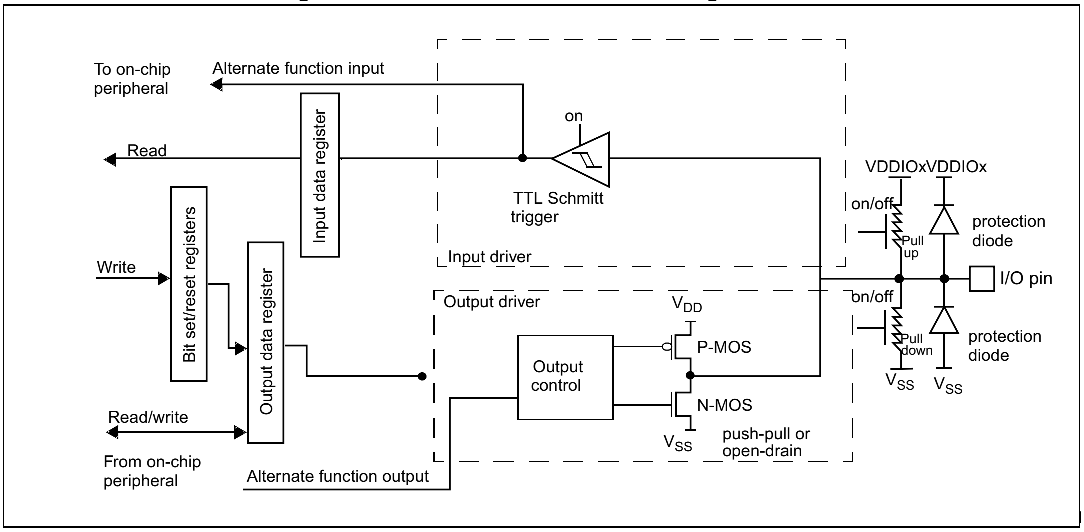
7.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

Figure 19 shows the Alternate function configuration of the I/O port bit.

Figure 19. Alternate function configuration



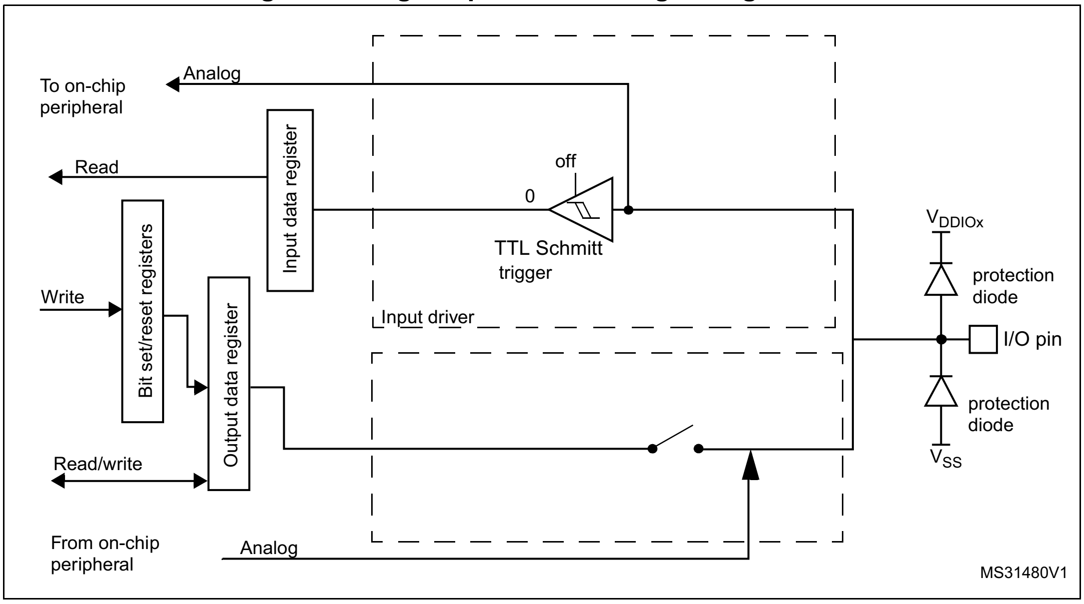
7.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The pull-up and pull-down resistors have to be disabled by the software else the associated analog feature does not work as expected.
- Read access to the input data register gets the value “0”

Figure 20 shows the high-impedance, analog-input configuration of the I/O port bit.

Figure 20. High impedance-analog configuration



7.3.13 Using the LSE oscillator pins as GPIOs

When the LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the oscillator is configured in a user external clock mode, only the OSC32_IN pin is reserved for clock input and the OSC32_OUT pin can still be used as normal GPIO.

Caution: There is no hardware mechanism to isolate software configuration automatically for the I/Os shared with OSC32_IN (PB13) and OSC32_OUT (PB12) when the external low speed oscillator (LSE) is used. The user has to take care to program the concerned I/O as input floating.

Note: The high speed oscillator (HSE) OSC_IN and OSC_OUT pins are dedicated oscillator pins and can not be used as GPIO.

7.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to *Section 7.5: GPIO register map*.

The peripheral registers can be written in word, half word or byte mode.

7.4.1 GPIOA port mode register (GPIOA_MODER)

Address offset:0x00

Reset values:0x0000 00A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODEy[1:0]:** Port A configuration bits (y = 0..15)

These bits are written by software to configure the I/O mode.

00: Input mode

01: output mode

10: Alternate function mode

11: Analog mode

Note: when configuring a pad in Analog mode, the user must take care to disable the associated pull-up/down to avoid pollution on the analog signal.

7.4.2 GPIOB port mode register (GPIOB_MODER)

Address offset:0x00

Reset values:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODEy[1:0]:** Port B configuration bits (y = 0..15)

These bits are written by software to configure the I/O mode.

00: Input mode

01: output mode

10: Alternate function mode

11: Analog mode

Note: when configuring a pad in Analog mode, the user must take care to disable the associated pull-up/down to avoid pollution on the analog signal.

7.4.3 GPIOA port output type register (GPIOA_OTYPER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **OTy:** Port A configuration bits (y = 0..15)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

7.4.4 GPIOB port output type register (GPIOB_OTYPER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **OTy:** Port B configuration bits (y = 0..15)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

7.4.5 GPIOA port output speed register (GPIOA_OSPEEDR)

Address offset: 0x08

Reset value: 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **OSPEEDy[1:0]**: Port A configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

7.4.6 GPIOB port output speed register (GPIOB_OSPEEDR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **OSPEEDy[1:0]**: Port B configuration bits (y = 0..15)
These bits are written by software to configure the I/O output speed.

7.4.7 GPIOA port pull-up/pull-down register (GPIOA_PUPDR)

Address offset: 0x0C

Reset values: 0x5555 5595

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **PUPDy[1:0]:** Port A configuration bits (y = 0..15)
These bits are written by software to configure the I/O pull-up or pull-down
- 00: No pull-up, pull-down
- 01: Pull-up
- 10: Pull-down
- 11: Reserved

7.4.8 GPIOB port pull-up/pull-down register (GPIOB_PUPDR)

Address offset: 0x0C

Reset values: 0x5555 5555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **PUPDy[1:0]:** Port B configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

7.4.9 GPIOA port input data register (GPIOA_IDR)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **IDy:** Port A input data bit (y = 0..15)

These bits are read-only. They contain the input value of the corresponding I/O port.

7.4.10 GPIOB port input data register (GPIOB_IDR)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **IDy:** Port B input data bit (y = 0..15)
These bits are read-only. They contain the input value of the corresponding I/O port.

7.4.11 GPIOA port output data register (GPIOA_ODR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD5	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **ODy:** Port A output data bit (y = 0..15)
These bits can be read and written by software.
Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A, B)

7.4.12 GPIOB port output data register (GPIOB_ODR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **ODy:** Port B output data bit (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A, B)

7.4.13 GPIOA port bit set/reset register (GPIOA_BSRR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port A reset bit y (y = 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Resets the corresponding ODx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port A set bit y (y= 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Sets the corresponding ODx bit

7.4.14 GPIOB port bit set/reset register (GPIOB_BSRR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port B reset bit y (y = 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Resets the corresponding ODx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port B set bit y (y= 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Sets the corresponding ODx bit

7.4.15 GPIOA port configuration lock register (GPIOA_LCKR)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = ‘1’ + LCKR[15:0]

WR LCKR[16] = ‘0’ + LCKR[15:0]

WR LCKR[16] = ‘1’ + LCKR[15:0]

RD LCKR

RD LCKR[16] = ‘1’ (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns ‘1’ until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port A lock bit y (y= 0..15)

These bits are read/write but can only be written when the LCKK bit is ‘0, using the specific sequence described in LCKK bit description.

0: Port configuration not locked

1: Port configuration locked

7.4.16 GPIOB port configuration lock register (GPIOB_LCKR)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = ‘1’ + LCKR[15:0]

WR LCKR[16] = ‘0’ + LCKR[15:0]

WR LCKR[16] = ‘1’ + LCKR[15:0]

RD LCKR

RD LCKR[16] = ‘1’ (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns ‘1’ until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port B lock bit y (y= 0..15)

These bits are read/write but can only be written when the LCKK bit is ‘0, using the specific sequence described in LCKK bit description.

0: Port configuration not locked

1: Port configuration locked

7.4.17 GPIOA alternate function low register (GPIOA_AFRL)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **y[3:0]:** Alternate function selection for port A pin y (y = 0..7)
These bits are written by software to configure alternate function I/Os

AFSELy selection:
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
Other: Reserved

7.4.18 GPIOB alternate function low register (GPIOB_AFRL)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **y[3:0]:** Alternate function selection for port B pin y (y = 0..7)
These bits are written by software to configure alternate function I/Os

AFSELy selection:
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
Other: Reserved

7.4.19 GPIOA alternate function high register (GPIOA_AFRH)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **y[3:0]:** Alternate function selection for port A pin y (y = 8..15)
These bits are written by software to configure alternate function I/Os

AFSELy selection:
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
Other: Reserved

7.4.20 GPIOB alternate function high register (GPIOB_AFRH)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **y[3:0]:** Alternate function selection for port B pin y (y = 8..15)
These bits are written by software to configure alternate function I/Os

AFSELy selection:
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
Other: Reserved

7.4.21 GPIOA port bit reset register (GPIOA_BRR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port A reset bit y (y = 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Resets the corresponding ODx bit

7.4.22 GPIOB port bit set/reset register (GPIOB_BSRR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 15:0 **BRy:** Port B reset bit y (y = 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Resets the corresponding ODx bit

7.5 GPIO register map

The following table gives the GPIO register map and reset values.

Table 19. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
0x00	GPIOB_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPIOA_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPIOB_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOA_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
0x08	GPIOB_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOA_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	0	1	0	1
0x0C	GPIOB_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1
0x10	GPIOA_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x10	GPIOB_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	GPIOA_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19. GPIO register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	GPIOB_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOA_BSRRA	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOB_BSRR	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOA_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOB_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOA_AFRL	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOB_AFRL	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOA_AFRH (where x = A, B)	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOB_AFRH (where x = A, B)	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOA_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOB_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

8 System controller (SYSCFG)

The System Controller is a set of registers (configuration, control and status) linked to system features of the STM32WL30xx/31xx/33xx.

8.1 SYSCFG main features

The System Controller set of registers are mainly linked to:

- Provide the JTAG_ID, Die ID and cut version
- Manage the interrupts linked to:
  - GPIOs
  - Power Controller (PWRC)
  - Comparator (COMP (STM32WL33xx only))
  - MR_SUBG
  - LCSC (STM32WL33xx only)
- Enable/disable I2C Fast-mode Plus driving capability on some I/Os

Note: This peripheral is in the non-retained power domain so all settings done in the associated registers are lost after a Deepstop.

8.2 System controller register descriptions

Note: Comparator, DAC, I2C2, LCSC, LCD, LPAWUR are available only on STM32WL33xx devices.
ADC, temperature sensor, I2C1, SPI1, I2S are available only on STM32WL31xx/33xx devices.
Any reference to these IPs on System control registers is valid within this perimeter.

8.2.1 Die ID register (DIE_ID)

This register provides the device version and cut information.

Address offset: 0x00

Reset value: 0x0000 0120

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **PRODUCT**: Product version.

Bits 7:4 **VERSION**: Cut version.

Bits 3:0 **REVISION**: Cut revision (metal fix)

8.2.2 JTAG ID register (JTAG_ID)

This register provides the JTAG ID of the STM32WL30xx/31xx/33xx.

Note: The same information is also available through direct JTAG access to test registers.

Address offset: 0x04

Reset value: 0x0202 7041

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

Bits 31:28 **VERSION_NUMBER**: Version.

Bits 27:12 **PART_NUMBER**: part number.

Bits 11:1 **MANUF_ID**: Manufacturer ID.

Bit 0 **RESERVED**

8.2.3 Fast-Mode Plus pin capability control register (I2C_FMP_CTRL)

This register allows activating the Fast-mode Plus driving capability on I2C open-drain pads.

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	I2C2_PA14_FMP	2C2_PA13_FMP	I2C2_PA7_FMP	I2C2_PA6_FMP	I2C1_PB11_FMP	I2C1_PB10_FMP	I2C1_PB7_FMP	I2C1_PB6_FMP	I2C1_PA1_FMP	I2C1_PA0_FMP
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 9 **I2C2_PA14_FMP**: I2C2 Fast-Mode Plus driving capability for I2C2_SDA on PA14 I/O.
- 0: PA14 pin operated in standard mode.
- 1: FM+ mode is enabled on PA14 pin, and speed control is bypassed.

Bit 8 **I2C2_PA13_FMP**: I2C2 Fast-Mode Plus driving capability for I2C2_SCL on PA13 I/O.
- 0: PA13 pin operated in standard mode.
- 1: FM+ mode is enabled on PA13 pin, and speed control is bypassed.

Bit 7 **I2C2_PA7_FMP**: I2C2 Fast-Mode Plus driving capability for I2C2_SDA on PA7 I/O.
- 0: PA7 pin operated in standard mode.
- 1: FM+ mode is enabled on PA7 pin, and speed control is bypassed.

Bit 6 **I2C2_PA6_FMP**: I2C2 Fast-Mode Plus driving capability for I2C2_SCL on PA6 I/O.
- 0: PA6 pin operated in standard mode.
- 1: FM+ mode is enabled on PA6 pin, and speed control is bypassed.

Bit 5 **I2C1_PB11_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SCL on PB11 I/O.
- 0: PB11 pin operated in standard mode.
- 1: FM+ mode is enabled on PB11 pin, and speed control is bypassed.

Bit 4 **I2C1_PB10_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SDA on PB10 I/O.
- 0: PB10 pin operated in standard mode.
- 1: FM+ mode is enabled on PB10 pin, and speed control is bypassed.

Bit 3 **I2C1_PB7_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SDA on PB7 I/O.
- 0: PB7 pin operated in standard mode.
- 1: FM+ mode is enabled on PB7 pin, and speed control is bypassed.

Bit 2 **I2C1_PB6_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SCL on PB6 I/O.
- 0: PB6 pin operated in standard mode.
- 1: FM+ mode is enabled on PB6 pin, and speed control is bypassed.

Bit 1 **I2C1_PA1_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SDA on PA1 I/O.
- 0: PA1 pin operated in standard mode.
- 1: FM+ mode is enabled on PA1 pin, and speed control is bypassed.

Bit 0 **I2C1_PA0_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SCL on PA0 I/O.
- 0: PA0 pin operated in standard mode.
- 1: FM+ mode is enabled on PA0 pin, and speed control is bypassed.

8.2.4 I/O Interrupt detection type register (IO_DTR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_DT	PB14_DT	PB13_DT	PB12_DT	PB11_DT	PB10_DT	PB9_DT	PB8_DT	PB7_DT	PB6_DT	PB5_DT	PB4_DT	PB3_DT	PB2_DT	PB1_DT	PB0_DT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA15_DT	PA14_DT	PA13_DT	PA12_DT	PA11_DT	PA10_DT	PA9_DT	PA8_DT	PA7_DT	PA6_DT	PA5_DT	PA4_DT	PA3_DT	PA2_DT	PA1_DT	PA0_DT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **PBx_DT** (x=15 to 0): Interrupt Detection Type for port B I/Os.
- 0: edge detection.
- 1: level detection.

Bits 15:0 **PAx_DT** (x=15 to 0): Interrupt Detection Type for Port A I/Os.
- 0: edge detection.
- 1: level detection.

8.2.5 I/O interrupt edge register (IO_IBER)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_IBE	PB14_IBE	PB13_IBE	PB12_IBE	PB11_IBE	PB10_IBE	PB9_IBE	PB8_IBE	PB7_IBE	PB6_IBE	PB5_IBE	PB4_IBE	PB3_IBE	PB2_IBE	PB1_IBE	PB0_IBE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA15_IBE	PA14_IBE	PA13_IBE	PA12_IBE	PA11_IBE	PA10_IBE	PA9_IBE	PA8_IBE	PA7_IBE	PA6_IBE	PA5_IBE	PA4_IBE	PA3_IBE	PA2_IBE	PA1_IBE	PA0_IBE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **PBx_IBE** (x=15 to 0): Interrupt edge selection for port B I/Os.
- 0: single edge detection.
- 1: both edges detection.

Bits 15:0 **PAx_IBE** (x=15 to 0): Interrupt edge selection for Port A I/Os.
- 0: single edge detection.
- 1: both edges detection.

8.2.6 I/O interrupt polarity event register (IO_IEVR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_IEV	PB14_IEV	PB13_IEV	PB12_IEV	PB11_IEV	PB10_IEV	PB9_IEV	PB8_IEV	PB7_IEV	PB6_IEV	PB5_IEV	PB4_IEV	PB3_IEV	PB2_IEV	PB1_IEV	PB0_IEV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA15_IEV	PA14_IEV	PA13_IEV	PA12_IEV	PA11_IEV	PA10_IEV	PA9_IEV	PA8_IEV	PA7_IEV	PA6_IEV	PA5_IEV	PA4_IEV	PA3_IEV	PA2_IEV	PA1_IEV	PA0_IEV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **PBx_IEV** (x=15 to 0): Interrupt polarity event for port B I/Os.
- 0: falling edge / low level.
- 1: rising edge / high level.

Bits 15:0 **PAx_IEV** (x=15 to 0): Interrupt polarity event for Port A I/Os.
- 0: falling edge / low level.
- 1: rising edge / high level.

8.2.7 I/O interrupt enable register (IO_IER)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_IE	PB14_IE	PB13_IE	PB12_IE	PB11_IE	PB10_IE	PB9_IE	PB8_IE	PB7_IE	PB6_IE	PB5_IE	PB4_IE	PB3_IE	PB2_IE	PB1_IE	PB0_IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PA15_IE	PA14_IE	PA13_IE	PA12_IE	PA11_IE	PA10_IE	PA9_IE	PA8_IE	PA7_IE	PA6_IE	PA5_IE	PA4_IE	PA3_IE	PA2_IE	PA1_IE	PA0_IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **PBx_IE** (x=15 to 0): Interrupt enable for port B I/Os.
- 0: interrupt is disabled.
- 1: interrupt is enabled.

Bits 15:0 **PAx_IE** (x=15 to 0): Interrupt enable for Port A I/Os.
- 0: interrupt is disabled.
- 1: interrupt is enabled.

8.2.8 I/O interrupt status and clear register (IO_ISCR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_ISC	PB14_ISC	PB13_ISC	PB12_ISC	PB11_ISC	PB10_ISC	PB9_ISC	PB8_ISC	PB7_ISC	PB6_ISC	PB5_ISC	PB4_ISC	PB3_ISC	PB2_ISC	PB1_ISC	PB0_ISC
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PA15_ISC	PA14_ISC	PA13_ISC	PA12_ISC	PA11_ISC	PA10_ISC	PA9_ISC	PA8_ISC	PA7_ISC	PA6_ISC	PA5_ISC	PA4_ISC	PA3_ISC	PA2_ISC	PA1_ISC	PA0_ISC
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 **PBx_ISC** (x=15 to 0): Interrupt status (before mask) for port B I/Os.
- 0: no pending interrupt.
- 1: event occurred on corresponding I/O / interrupt occurred (if enabled).

Cleared by writing 1 in the bit.

Bits 15:0 **PAx_ISC** (x=15 to 0): Interrupt status (before mask) for port A I/Os.
- 0: no pending interrupt.
- 1: event occurred on corresponding I/O / interrupt occurred (if enabled).
- Cleared by writing 1 in the bit.

8.2.9 Power controller interrupt enable register (PWRC_IER)

This register allows control of the enable or mask on the interrupt sources of the Power Controller (PWRC) block.

Address offset: 0x20

Reset value: 0x0000 0000

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

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

Bit 2 **WKUP_IE**: Power Controller Wakeup event interrupt enable.
- 0: Interrupt on wakeup event seen by the PWRC is disabled.
- 1: Interrupt on wakeup event seen by the PWRC is enabled.

Bit 1 **PVD_IE**: Programmable Voltage Detector interrupt enable.
- 0: PVD interrupt is disabled.
- 1: PVD interrupt is enabled.

Bit 0 **BORH_IE**: BORH interrupt enable.
- 0: BORH interrupt is disabled.
- 1: BORH interrupt is enabled.

8.2.10 Power controller interrupt status and clear register (PWRC_ISCR)

This register allows checking the status and clear the interrupt sources of the Power Controller (PWRC) block.

Address offset: 0x24

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUP_ISC	PVD_ISC	BORH_ISC
													rc_w1	rc_w1	rc_w1

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

Bit 2 **WKUP_ISC**: Indicates the Power Controller receives a Wakeup event.
- 0: no pending interrupt.
- 1: Wakeup event on PWRC occurred / interrupt occurred (if enabled).

Cleared by writing 1 in the bit.
This flag is read at 1 if a wakeup event arrives so close to the low power mode entry requests that the PWRC aborts before shutting down the system.

Bit 1 **PVD_ISC**: Programmable Voltage Detector status.
- 0: no pending interrupt.
- 1: voltage went under programmed threshold / interrupt occurred (if enabled).

Cleared by writing 1 in the bit.
See *Section 5.3.3: Power voltage detection (PVD)* for details.

Bit 0 **BORH_ISC**: BORH interrupt enable.
- 0: no pending interrupt.
- 1: voltage went under BORH threshold / interrupt occurred (if enabled).

Cleared by writing 1 in the bit.
See *Section 5.3.2: BORH* for details.

8.2.11 I/O analog switch control register (GPIO_SWA_CTRL)

This register allows selecting the analog source to connect on analog pads embedding two features.

Address offset: 0x28

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATB1_nPVD
															rw

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

Bit 0 **ATB1_nPVD**: select the analog feature on PB14 between ATB1 and PVD when the PB14 I/O is programmed in analog mode (in the associated GPIO_MODER register):

– 0: PVD external voltage feature is selected (default).

– 1: ATB1 feature is selected.

8.2.12 Internal asynchronous interrupt detection type register (INTAI_DTR)

This register allows to select the type of detection (level or edge) of internal asynchronous events.

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6 **LC_ACTIVITY_DT**: detection type on LC_ACTIVITY signal:
- 0: detection on edge (default).
- 1: detection on level

Bit 5 **RFIP_BUSY_STATUS_DT**: detection type on RFIP_BUSY_STATUS signal:
- 0: detection on edge (default).
- 1: detection on level

Bit 4 **COMP_DT**: detection type on COMP_OUT (after COMP_POL selection) signal (STM32WL33xx only):
- 0: detection on edge (default).
- 1: detection on level

Bits 3:2 Reserved, must be kept at reset value.

Bits 1 **RX_DT**: detection type on RX_SEQUENCE signal:
- 0: detection on edge (default).
- 1: detection on level

Bits 0 **TX_DT**: detection type on TX_SEQUENCE signal:
- 0: detection on edge (default).
- 1: detection on level

8.2.13 Internal asynchronous interrupt edge register (INTAI_IBER)

This register allows to select the detected edge of internal asynchronous events. Options are single edge or both edges when edge detection type is active. The configuration of this register hasn't any effect if the corresponding detection type is set to 'detection on level' in the INTAI_DTR register.

Address offset: 0x30

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LC_ACTIVITY_IBE	RFIP_BUSY_STAUS_IBE	COMP_IBE	Res.	Res.	RX_IBE	TX_IBE
									rw	rw	rw			rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LC_ACTIVITY_IBE**: interrupt edge register on LC_ACTIVITY signal:
– 0: detection on single edge (default).
– 1: detection on both edges

Bit 5 **RFIP_BUSY_STATUS_IBE**: interrupt edge register on RFIP_BUSY_STATUS signal:
– 0: detection on single edge (default).
– 1: detection on both edges

Bit 4 **COMP_IBE**: interrupt edge register on COMP_OUT signal (STM32WL33xx only):
– 0: detection on single edge (default).
– 1: detection on both edges

Bits 3:2 Reserved, must be kept at reset value.

Bits 1 **RX_IBE**: interrupt edge register on RX_SEQUENCE signal:
– 0: detection on single edge (default).
1: detection on both edges

Bits 0 **TX_IBE**: interrupt edge register on TX_SEQUENCE signal:
– 0: detection on single edge (default).
1: detection on both edges

8.2.14 Internal asynchronous interrupt polarity event register (INTAI_IEVR)

This register defines the polarity of the internal asynchronous signals detection.

Address offset: 0x34

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LC_ACTIVITY_IEV	RFIP_BUSY_STATUS_IEV	COMP_IEV	Res.	Res.	RX_IEV	TX_IEV
									rw	rw	rw			rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LC_ACTIVITY_IEV**: interrupt polarity event on LC_ACTIVITY signal:
– 0: detection on falling edge / low level (default).
– 1: detection on rising edge / high level

Bit 5 **RFIP_BUSY_STATUS_IEV**: interrupt polarity event on RFIP_BUSY_STATUS signal:
– 0: detection on falling edge / low level (default).
– 1: detection on rising edge / high level

Bit 4 **COMP_IEV**: interrupt polarity event on COMP_OUT signal (STM32WL33xx only):
– 0: detection on falling edge / low level (default).
– 1: detection on rising edge / high level

Bits 3:2 Reserved, must be kept at reset value.

Bits 1 **RX_IEV**: interrupt polarity event on RX_SEQUENCE signal:
– 0: detection on falling edge / low level (default).
– 1: detection on rising edge / high level

Bits 0 **TX_IEV**: interrupt polarity event on TX_SEQUENCE signal:
– 0: detection on falling edge / low level (default).
– 1: detection on rising edge / high level

8.2.15 Internal asynchronous interrupt enable register (INTAI_IER)

This register enable/disable interrupt generation on internal asynchronous signals detection.

Address offset: 0x38

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LC_ACTIVITY_IE	RFIP_BUSY_STATUS_IE	COMP_IE	Res.	Res.	RX_IE	TX_IE
									rw	rw	rw			rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LC_ACTIVITY_IE**: interrupt enable on LC_ACTIVITY signal:
- 0: LC_ACTIVITY interrupt is disabled (default).
- 1: LC_ACTIVITY interrupt is enabled

Bit 5 **RFIP_BUSY_STATUS_IE**: interrupt enable on RFIP_BUSY_STATUS signal:
- 0: RFIP_BUSY_STATUS interrupt is disabled (default).
- 1: RFIP_BUSY_STATUS interrupt is enabled

Bit 4 **COMP_IE**: interrupt enable on COMP_OUT signal:
- 0: COMP_OUT interrupt is disabled (default).
- 1: COMP_OUT interrupt is enabled

Bits 3:2 Reserved, must be kept at reset value.

Bits 1 **RX_IE**: interrupt enable on RX_SEQUENCE signal:
- 0: RX_SEQUENCE interrupt is disabled (default).
- 1: RX_SEQUENCE interrupt is enabled

Bits 0 **TX_IE**: interrupt enable on TX_SEQUENCE signal:
- 0: TX_SEQUENCE interrupt is disabled (default).
- 1: TX_SEQUENCE interrupt is enabled

8.2.16 Internal asynchronous interrupt detection status and clear register (INTAI_ISCR)

This register allows checking and clear the status of the internal asynchronous interrupts.

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LC_ACTIVITY_ISEDGE	LC_ACTIVITY_ISC	RFIP_BUSY_STATUS_ISC	COMP_ISC	RX_ISEDGE	TX_ISEDGE	RX_ISC	TX_ISC
								r	rc_w1	rc_w1	rc_w1	r	r	rc_w1	rc_w1

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

Bit 7 **LC_ACTIVITY_ISEDGE**: interrupt edge status on LC_ACTIVITY signal:
– 0: falling edge on LC_ACTIVITY detected.
– 1: rising edge on LC_ACTIVITY detected.

Bit 6 **LC_ACTIVITY_ISC**: interrupt status on LC_ACTIVITY (can be a rising or a falling edge depending on INTAI_IEVR and INTAI_IBER):
– 0: no activity on LC_ACTIVITY detected.
– 1: activity on LC_ACTIVITY occurred.

Bit 5 **RFIP_BUSY_STATUS_ISC**: interrupt status on RFIP_BUSY_STATUS (can be a rising or a falling edge depending on INTAI_IEVR and INTAI_IBER):
– 0: no activity on RFIP_BUSY_STATUS detected.
– 1: activity on RFIP_BUSY_STATUS occurred.

Bit 4 **COMP_ISC**: interrupt status on COMP_OUT (can be a rising or a falling edge depending on INTAI_IEVR and INTAI_IBER):
– 0: no activity on COMP_OUT detected.
– 1: activity on COMP_OUT occurred.

Bit 3 **RX_ISEDGE**: interrupt edge status on RX_SEQUENCE signal:
– 0: falling edge on RX_SEQUENCE detected.
– 1: rising edge on RX_SEQUENCE detected.

Bit 2 **TX_ISEDGE**: interrupt edge status on TX_SEQUENCE signal:
– 0: falling edge on TX_SEQUENCE detected.
– 1: rising edge on TX_SEQUENCE detected.

Bit 1 **RX_ISC**: interrupt status on RX_SEQUENCE (can be a rising or a falling edge depending on INTAI_IEVR and INTAI_IBER):
– 0: no activity on RX_SEQUENCE detected.
– 1: activity on RX_SEQUENCE occurred.

Bit 0 **TX_ISC**: interrupt status on TX_SEQUENCE (can be a rising or a falling edge depending on INTAI_IEVR and INTAI_IBER):
– 0: no activity on TX_SEQUENCE detected.
– 1: activity on TX_SEQUENCE occurred.

8.2.17 SYSCFG status register 1 (SYSCFG_SR1)

Address offset: 0x40

Reset value: 0x0000 0000

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

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **RFIP_BUSY_STATUS**: MR_SUBG BUSY status:
Software should check that MR_SUBG IP is not busy (or relay on the related interrupt) before to initiate any system clock frequency switch to operate the switching in a safe way.
– 0: MR_SUBG is not busy.
– 1: MR_SUBG is busy

Bits 4:0 Reserved, must be kept at reset value.

8.3 System controller register map

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the System controller base address location in the STM32WL30xx/31xx/33xx.

Table 20. SYSCFG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DIE_ID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRODUCT				VERSION				REVISION			
	Reset value																					0	0	0	1	0	0	1	0	0	0	0	0
0x04	JTAG_ID	VERSION_NUMBER				PART_NUMBER																MANUF_ID											Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	
0x08	I2C_FMP_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C2_PA14_FMP	I2C2_PA13_FMP	I2C2_PA7_FMP	I2C2_PA6_FMP	I2C1_PB11_FMP	I2C1_PB10_FMP	I2C1_PB7_FMP	I2C1_PB6_FMP	I2C1_PA1_FMP	I2C1_PA0_FMP
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x0C	IO_DTR	PB15_DT	PB14_DT	PB13_DT	PB12_DT	PB11_DT	PB10_DT	PB9_DT	PB8_DT	PB7_DT	PB6_DT	PB5_DT	PB4_DT	PB3_DT	PB2_DT	PB1_DT	PB0_DT	PA15_DT	PA14_DT	PA13_DT	PA12_DT	PA11_DT	PA10_DT	PA9_DT	PA8_DT	PA7_DT	PA6_DT	PA5_DT	PA4_DT	PA3_DT	PA2_DT	PA1_DT	PA0_DT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	IO_IBER	PB15_IBE	PB14_IBE	PB13_IBE	PB12_IBE	PB13_IBE	PB12_IBE	PB10_IBE	PB9IBE	PB7_IBE	PB6_IBE	PB5_IBE	PB4_IBE	PB3_IBE	PB2_IBE	PB1_IBE	PB0_IBE	PA14_IBE	PA14_IBE	PA13_IBE	PA12_IBE	PA11_IBE	PA10_IBE	PA9_IBE	PA8_IBE	PA7_IBE	PA6_IBE	PA5_IBE	PA4_IBE	PA3_IBE	PA2_IBE	PA1_IBE	PA0_IBE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	IO_IEVR	PB15_IEV	PB14_IEV	PB13_IEV	PB12_IEV	PB11_IEV	PB10_IEV	PB9_IEV	PB8_IEV	PB7_IEV	PB6_IEV	PB5_IEV	PB4_IEV	PB3_IEV	PB2_IEV	PB1_IEV	PB0_IEV	PA15_IEV	PA14_IEV	PA13_IEV	PA12_IEV	PA11_IEV	PA10_IEV	PA9_IEV	PA8_IEV	PA7_IEV	PA6_IEV	PA5_IEV	PA4_IEV	PA3_IEV	PA2_IEV	PA1_IEV	PA0_IEV
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	IO_IER	PB15_IE	PB14_IE	PB13_IE	PB12_IE	PB11_IE	PB10_IE	PB9_IE	PB8_IE	PB7_IE	PB6_IE	PB5_IE	PB4_IE	PB3_IE	PB2_IE	PB1_IE	PB0_IE	PA15_IE	PA14_IE	PA13_IE	PA12_IE	PA11_IE	PA10_IE	PA9_IE	PA8_IE	PA7_IE	PA6_IE	PA5_IE	PA4_IE	PA3_IE	PA2_IE	PA1_IE	PA0_IE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	IO_ISCR	PB15_ISC	PB14_ISC	PB13_ISC	PB12_ISC	PB11_ISC	PB10_ISC	PB9_ISC	PB8_ISC	PB7_ISC	PB6_ISC	PB5_ISC	PB4_ISC	PB3_ISC	PB2_ISC	PB1_ISC	PB0_ISC	PA15_ISC	PA14_ISC	PA13_ISC	PA12_ISC	PA11_ISC	PA10_ISC	PA9_ISC	PA8_ISC	PA7_ISC	PA6_ISC	PA5_ISC	PA4_ISC	PA3_ISC	PA2_ISC	PA1_ISC	PA0_ISC
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20. SYSCFG register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	PWRC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUP_IE	PVD_IE	BORH_IE
	Reset value																														0	0	0
0x24	PWRC_ISCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUP_ISC	PVD_ISC	BORH_ISC
	Reset value																														0	0	0
0x28	Reserved																																
0x2C	INTAI_DTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LC_ACTIVITY_DT	RFIP_BUSY_STATUS_DT	COMP_DT	Res.	Res.	RX_DT	TX_DT
	Reset value																										0	0	0			0	0
0x30	INTAI_IBER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LC_ACTIVITY_IBE	RFIP_BUSY_STATUS_IBE	COMP_IBE	Res.	Res.	RX_IBE	TX_IBE
	Reset value																										0	0	0			0	0
0x34	INTAI_IEVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LC_ACTIVITY_IEV	RFIP_BUSY_STATUS_IEV	COMP_IEV	Res.	Res.	RX_IEV	TX_IEV
	Reset value																										0	0	0			0	0
0x38	INTAI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LC_ACTIVITY_IE	RFIP_BUSY_STATUS_IE	COMP_IE	Res.	Res.	RX_IE	TX_IE
	Reset value																										0	0	0			0	0

Table 20. SYSCFG register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3C	INTAI_ISCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LC_ACTIVITY_ISEDGE	LC_ACTIVITY_ISC	RFIP_BUSY_STATUS_ISC	COMP_ISC	RX_ISEDGE	TX_ISEDGE	RX_ISC	TX_ISC
	Reset value																									0	0	0	0	0	0	0	0
0x40	SYSCFG_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFIP_BUSY_STATUS	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					

9 Embedded flash memory (FLASH)

The flash controller implements the erase and program flash memory operation. The flash controller also implements the read and write protection.

9.1 Flash main features

Flash memory features:

- Flash memory with single-bank architecture:
  - up to 256 Kbytes for STM32WL33xx
  - up to 128 Kbytes for the STM32WL30xx/31xx/33xx
- Memory organization:
  - 1 bank
  - main memory up to 256 Kbytes on STM32WL33xx
  - main memory up to 128 kbytes on STM32WL30xx/31xx
  - page size: 2 Kbytes
- 32-bit wide data read.
- 32-bit wide data write.
- Page erase (2 Kbytes) and mass erase.

Flash controller features:

- Flash memory read operations
- Flash memory write operations: single data write or 4x32-bits burst write (to reduce programming time by 4)
- Flash memory erase operations
- Page write protect mechanism by 4 segments of variable size:
  - from 1 to 63 pages for STM32WL30xx/31xx devices
  - from 1 to 127 pages for STM32WL33xx devices
- Option byte loader hardware mechanism dedicated toST analog trimming bits.

Caution: When the BORH feature is enabled in the PWRC (PWRC_CR1.ENBORH = 1), the flash controller stops programming activity on the flash memory as soon as the power supply voltage is below the BORH threshold (visible through the flag PWRC_ISCR.BORHISC in the system controller).

9.2 Description

The flash memory is organized as follows, depending on the device:

- The STM32WL30xx/31xx/33xx main memory block contains up to 64 pages of 2 Kbytes. Each page is made of 8 rows of 256 bytes (64 words).
- The STM32WL3xx main memory block contains 128 pages of 2 Kbytes. Each page is made of 8 rows of 256 bytes (64 words).

Erasing the whole flash memory results in all ones in every bit cell of the flash.

In parallel, the flash controller manages 1 Kbyte OTP (One-Time Programmable) for user data. The OTP data cannot be erased. The OTP data area can no more be written only as soon as the last word (address 0x1000_1BFC) is different from 0xFFFF_FFFF and a system reset occurred.

The flash memory is mapped on the AHB-Lite bus with the range described below:

Table 21. Flash memory section address

Section	Flash AHB start address	Flash AHB end address
Main flash	0x1004_0000	0x1007_FFFF (1)
User OTP	0x1000_1800	0x1000_1BFF

1. Depends on flash size. See *Table 24: Flash size information on page 207*.

9.3 Flash controller register descriptions

9.3.1 Command register (COMMAND)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

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

Bit 7:0 **COMMAND**: Command opcode to launch any operation on flash memory. See for command list and detail.

Table 22. Command list available for customer

Command	Flash sector	Description	Command Opcode
ERASE	main memory	Erase page defined by register ADDRESS.	0x11
WRITE (1)	main memory	Program one location (defined by registers DATA and ADDRESS).	0x33
MASSREAD	main memory	Read all locations and compare with DATA register value or produce LFSR signature.	0x55
SLEEP	Whole memory	Put flash in sleep mode. ***Warning: once this command is launched, no access (read) nor action (any command except WAKE) on the flash component are possible until the WAKE command is requested (and associated CMDDONE status is returned)***	0xAA
WAKEUP	Whole memory	Get flash out of sleep mode.	0xBB
BURSTWRITE	main memory	Program 4 locations (ADDRESS → ADDRESS+3) with data in DATA0-DATA3 registers. ***Warning: a burst always starts (=DATA0 is always written) on a 16-bytes aligned address even if ADDRESS is not 16-bytes aligned (register bit field ADDRESS[1:0] always considered as 0 at flash controller level).***	0xCC
OTPWRITE	User OTP	One time writable 1 Kbytes for customer. No erase not second programming is possible.	0xEE

1. Each address can be programmed only 2 times without erase operation in between.

Status bits behavior versus commands:

- Writing to the COMMAND register starts the action that is performed on the flash memory.
- The CMDSTART flag goes and stays high until it is cleared.
- When the command has finished the CMDDONE flag goes high.
- When a MASS READ command was given and when CMDDONE is high, the READOK flag can be checked or the LFSRVAL register can be read (contains the signature of the mass read).

The sequences to use the different commands are described in *9.7: Programmers model on page 214*.

9.3.2 Configuration register (CONFIG)

Address offset: 0x04

Reset value: 0x0000 0018

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

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **WAIT_STATES**: Number of wait states to be inserted on flash read (AHB accesses).
The flash memory embedded in STM32WL30xx/31xx/33xx devices requires 1 wait_state when system clock frequency is 64 MHz.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DIS_GROUP_WRITE**:
– 0: burst write operations are allowed/enabled.
– 1: burst write operations are blocked and results in a single write.
Note: if this bit is set during an on-going burst write operation, the flash controller stops the write operation at the end of the current word writing even if some words are still to be written.

Bit 1 **REMAP**: bit to redirect boot area on SRAM0. See *Table 5: Address remapping depending on the REMAP bit* for details.

Bit 0 Reserved, must be kept at reset value.

The flash can be read in one system clock cycle (the best for power consumption) when the system clock is 32 MHz maximum.

9.3.3 Interrupt status register (IRQSTAT)

The interrupt status register shows the masked version of the interrupt raw register.

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READOK_MIS	ILLCMD_MIS	CMDERR_MIS	CMDSTART_MIS	CMDDONE_MIS
											rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bit 4 **READOK_MIS**: Mass read OK masked interrupt status.
This bit is set at the end of a MASSREAD operation if all the words read in the memory match the DATA0 register value.
Cleared by writing 1.

Bit 3 **ILLCMD_MIS**: Illegal command masked interrupt status.
This bit is set when a bad opcode command is written in the COMMAND register.
Cleared by writing 1.

Bit 2 **CMDERR_MIS**: command error masked interrupt status.
This bit is set if a command opcode is written in COMMAND register while the flash is busy.
Cleared by writing 1.

Bit 1 **CMDSTART_MIS**: Command started masked interrupt status.
This bit is set once the requested command execution has started.
Cleared by writing 1.

Bit 0 **CMDDONE_MIS**: Command done masked interrupt status.
This it is set once the requested command execution is completed.
Cleared by writing 1.

The CMDDONE and CMDSTART bits are updated a few clock cycles after the requested command has been started by writing to the COMMAND register.

Note: Clearing a bit by writing in IRQSTAT (respectively IRQRAW) register also cleared the same bit in IRQRAW (respectively IRQSTAT) register as they are referring to a common condition/event.

9.3.4 Interrupt mask register (IRQMASK)

The mask bit in IRQMASK masks the condition in the status register IRQSTAT and prevent the generation of the interrupt.

Address offset: 0x0C

Reset value: 0x0000 003F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READOKM	ILLCMDM	CMDERRM	CMDSTARTM	CMDDONEM
											rw	rw	rw	rw	rw

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

Bit 4 **READOKM**: Mass read OK mask.
- 0: enable interrupt on “Mass read OK” event.
- 1: disable interrupt on “Mass read OK” event

Bit 3 **ILLCMDM**: Illegal command mask.
- 0: enable interrupt on “illegal command” event.
- 1: disable interrupt on “illegal command” event

Bit 2 **CMDERRM**: command error mask.
- 0: enable interrupt on “command error” event.
- 1: disable interrupt on “command error” event

Bit 1 **CMDSTARTM**: Command started mask.
- 0: enable interrupt on “command started” event.
- 1: disable interrupt on “command started” event

Bit 0 **CMDDONEM**: Command done mask.
- 0: enable interrupt on “command done” event.
- 1: disable interrupt on “command done” event

9.3.5 Raw status register (IRQRAW)

The raw status register shows the unmasked condition of interrupt events.

Address offset: 0x10

Reset value: 0x0000 0001

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

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

Bit 4 **READOK_RIS**: Mass read OK raw/unmasked interrupt status.
This bit is set at the end of a MASSREAD operation if all the words read in the memory match the DATA0 register value.
Cleared by writing 1.

Bit 3 **ILLCMD_RIS**: Illegal command raw/unmasked interrupt status.
This bit is set when a bad opcode command is written in the COMMAND register.
Cleared by writing 1.

Bit 2 **CMDERR_RIS**: command error raw/unmasked interrupt status.
This bit is set if a command opcode is written in COMMAND register while the flash is busy.
Cleared by writing 1.

Bit 1 **CMDSTART_RIS**: Command started raw/unmasked interrupt status.
This bit is set once the requested command execution has started.
Cleared by writing 1.

Bit 0 **CMDDONE_RIS**: Command done raw/unmasked interrupt status.
This it is set once the requested command execution is completed.
Cleared by writing 1.

The CMDDONE and CMDSTART bits are updated a few clock cycles after the requested command has been started by writing to the COMMAND register.

Note: Clearing a bit by writing in IRQSTAT (respectively IRQRAW) register also cleared the same bit in IRQRAW (respectively IRQSTAT) register as they are referring to a common condition/event.

9.3.6 Size register (SIZE)

Address offset: 0x14

Reset value: 0x000- ---- (depends on device)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG_SIZE[1:0]		SWD_DISABLE	FLASH_SECURE	Res.	RAM_SIZE	Res.
									r	r	-	-		r	

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:21 **PKG_SIZE**: indicates device package size:
- 10: 32 pin package (QFN32).
- 11: 48 pin package (QFN48).

Bit 20 **SWD_DISABLE**: indicates the SWD is disabled on the device:
- 0: SWD feature is available on the device, a debugger can be used.
- 1: SWD feature is disabled on the device, a debugger can no more be used.

Bit 19 **FLASH_SECURE**: indicates the main flash memory is locked by a customer key.
- 0: the main flash memory is not protected.
- 1: the main flash memory is protected through a customer key.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **RAM_SIZE**: indicates the size of RAM available in the device:
- 0: 16 Kbytes of RAM available (SRAM0 bank)
- 1: 32 Kbytes of RAM available (SRAM0 and SRAM1 (STM32WL33xx only) banks)

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **FLASH_SIZE**: indicates the last usable address of the flash using memory component address format. See *Table 24* for relation between address at flash component level and AHB address mapping.
- 0x3FFF: 64 Kbytes of main flash are available on this device.
- 0x7FFF: 128 Kbytes of main flash are available on this device.
- 0xBFFF: 192 Kbytes of main flash are available on this device.
- 0xFFFF: 256 Kbytes of main flash are available on this device.

Table 24. Flash size information

Main flash size	Highest usable address at flash level[1]	Highest usable address at AHB level
64 Kbytes	0x3FFF	0x1004_FFFC
128 Kbytes	0x7FFF	0x1005_FFFC
192 Kbytes	0xBFFF	0x1006_FFFC
256 Kbytes	0xFFFF	0x1007_FFFC

1. Value seen in FLASH_SIZE bit field.

9.3.7 Address register (ADDRESS)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

Bits 15:6 **XADDR[9:0]**:
– XADDR[9:3]: page number (from 0 to 127)
– XADRR[2:0]: row number (from 0 to 7)

Bits 5:0 **YADDR[5:0]**: word number inside the selected row (from 0 to 63).
Note: when burst write command is used, YADDR[1:0] bit field is always considered as 0 by the flash controller.

Address to provide to the flash is not the AHB device mapping address but the address respecting flash component format.

The main flash is composed of 128 pages containing 8 rows each with 64 words = 256 bytes by row.

To program the ADDRESS register, the formula is the following:

– XADDR[9:0] = AHB address bit[17:8]
– YADDR[5:0] = AHB address bit[7:2]

Example 1: To program a word (32-bit) at AHB address 0x1005_0454:

– XADDR[9:0] = AHB address bit[17:8] = 0x104
– YADDR[5:0] = AHB address bit[7:2] = 0x15
– ADDRESS register = 0x4115

9.3.8 Linear feedback shift register (LFSRVAL)

The LFSRVAL register contains the signature issued by a MASSREAD command.

The LFSRVAL register is initialized with all ones when the MASS READ command is written to the COMMAND register. Then every read value is put through the LFSR.

The final signature can be read in this register once the CMDDONE information is set.

customer.

Address offset: 0x24

Reset value: 0xFFFF FFFF

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

Bits 31:0 **LSFRVAL**: signature after a MASSREAD command, generated through a Linear Feedback Shift Register block.

9.4 Main flash page protection registers 0 to 1

The PAGEPROTx registers allows protecting from accidental write a contiguous set of pages called segment in the following description. A maximum of four segments can be defined.

9.4.1 Page protection register 0 (PAGEPROT0)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEG1[15:0]															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SEG0[15:0]															
rw															

Bits 31:16 **SEG1**: second segment definition.
See SEG0 description for details on SEG1[15:0] content.

Bits 15:0 **SEG0**: First segment definition.
A segment SEGx is built as follows:
- SEGx[15]: reserved
- SEGx[14:8] = OFFSET: page number to start the write protection (value between 0 and 0x7F)
- SEGx[7]: reserved
- SEGx[6:0] = SIZE: number of 2 Kbyte pages to protect including the starting page (provided in SEGx[14:8]).

Note:
- SIZE=0 means no segment defined so if all segments have SIZE=0, then no write protection is applied on the whole flash memory.
- The segments can overlap, the protection on a page is guaranteed if at least one segment covers this page.
- If OFFSET + SIZE > 127d so exceed the maximum size of the flash memory, the end of the segment is positioned on the maximum allowed address (page 127).

9.4.2 Page protection register 1 (PAGEPROT1)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEG3[15:0]															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SEG2[15:0]															
rw															

Bits 31:16 **SEG3**: fourth segment definition.
See PAGEPROT0.SEG0 description for details on SEG3[15:0] content.

Bits 15:0 **SEG2**: third segment definition.
See PAGEPROT0.SEG0 description for details on SEG2[15:0] content.

9.5 Data registers 0 to 3

The DATA0 register needs to be written with:

- The desired value written to the flash location (for single write or mass write).
- The desired compare value for a (mass) read operation, the flag READOK indicates if there was a match or not. For mass read, all read values must match for READOK.

The DATA1-DATA3 registers need to be written only for burst write.

Data 0 register (DATA0)

Address offset: 0x40

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA0[31:16]															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA0[15:0]															
rw															

Bits 31:0 **DATA0**: this register has several usage:
– Data to write in flash in single write mode
– First data to be written in flash on a burst write
– Compared value for a MASSREAD command (useful only if flash is fully written with the same word)
Note: In this last case, the flag READOK indicates if there was a match or not at the end of the mass read.

9.5.1 Data 1 register (DATA1))

Address offset: 0x44

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA1[31:16]															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA1[15:0]															
rw															

Bits 31:0 **DATA1**: data that is written at ADDRESS+1 during a BURSTWRITE command.
Note: this register is used only on burst write.

9.5.2 Data 2 register (DATA2)

Address offset: 0x48

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA2[31:16]															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA2[15:0]															
rw															

Bits 31:0 **DATA2**: data that is written at ADDRESS+2 during a BURSTWRITE command.
Note: this register is used only on burst write.

9.5.3 Data 3 register (DATA3)

Address offset: 0x4C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[31:16]															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA3[15:0]															
rw															

Bits 31:0 **DATA3**: data that is written at ADDRESS+3 during a BURSTWRITE command.
Note: this register is used only on burst write.

9.6 Flash controller register map

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the flash controller base address location in the STM32WL30xx/31xx/33xx.

Table 25. Flash controller register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**COMMAND**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMMAND[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x04	**CONFIG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEP_SM	WAIT_STATE[1:0]		Res.	DIS_GROUP_WRITE	REMAP	LONGACCESS
	Reset value																										0	0	1		0	0	1
0x08	**IRQSTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READOK_MIS	ILLCMD_MIS	CMDERR_MIS	CMDSTART_MIS	CMDDONE_MIS
	Reset value																												0				
0x0C	**IRQMASK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READOKM	ILLCMDM	CMDERRM	CMDSTARTM	CMDDONEM
	Reset value																																
0x10	**IRQRAW**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READOK_RIS	ILLCMD_RIS	CMDERR_RIS	CMDSTART_RIS	CMDDONE_RIS
	Reset value																												0				
0x14	**SIZE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG_SIZE[1:0]		SWD_DISABLE	FLASH_SECURE	Res.	RAM_SIZE	Res.	FLASH_SIZE[15:0]															
	Reset value										-	-	-	-	-	-		-	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	**ADDRESS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XADDR[9:0]										YADDR[5:0]					
	Reset value																	0		0		0		0		0		0		0		0	
0x24	**LFSRVAL**	LSFRVAL[31:0]																															
	Reset value																																
0x34	PAGEPROT0	SEG1[15:0]																SEG0[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25. Flash controller register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38	PAGEPROT1	SEG3[15:0]																SEG2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	Reserved																																
0x40	**DATA0**	DATA0[31:0]																															
	Reset value																																
0x44	**DATA1**	DATA1[31:0]																															
	Reset value																																
0x48	**DATA2**	DATA2[31:0]																															
	Reset value																																
0x4C	**DATA3**	DATA3[31:0]																															
	Reset value																																

9.7 Programmers model

The flash memory size depends on the product:

- The STM32WL30xx/31xx/33xx embeds up to 128 Kbytes (32768 x 32-bit) of internal flash memory.
- The STM32WL30xx/31xx/33xx embeds up to 256 Kbytes (65536 x 32-bit) of internal flash memory

A flash interface implements instruction access and data access based on the AHB protocol. It implements the logic necessary to carry out the flash memory operations (Program/Erase) controlled through the flash registers.

9.7.1 General information

Writing to flash memory only allows the clearing of bits from ‘1’ to ‘0’. This means any write from ‘0’ to ‘1’ implies erasing before performing a write.

- The STM32WL30xx/31xx/33xx flash memory is composed of 64 pages containing 8 rows of 64 words (64 x 8 x 64 = 32768 words). Each word is 32-bit = 4 bytes long which means 256 Kbytes of flash memory.
- The STM32WL33xx flash memory is composed of 128 pages containing 8 rows of 64 words (128 x 8 x 64 = 65536 words). Each word is 32-bit = 4 bytes long which means 256 Kbytes of flash memory.

The address inside the flash controller ADDRESS register is built as follows:

ADDRESS[13:0] = XADR[9:0] & YADR[5:0] with

- XADR[9:3] = page address
- XADR[2:0] = row address
- YADR[5:0] = word address (one word = four bytes)

Note: One specific address can be written only twice between two erase actions even if each writing only clears bits.

9.7.2 Write page protection example

Example to write protect against accidental programming knowing the flash starts at address 0x1004_0000 and contains 128 pages of 2 Kbytes (pages 0 to 127).

- the address ranges 0x1004C000-0x1004FFFF

Starting page: 0xC000 / 0x800 = 0x18

SEG0.OFFSET = 0x18

Number of pages: (0x10000 - 0xC000) / 0x800 = 0x8

SEG0.SIZE = 0x08

- and the address ranges 0x1005E000-0x1005FFFF.

Starting page: 0x1E000 / 0x800 = 0x3C

SEG1.OFFSET = 0x38

Number of pages: (0x20000 - 0x1E000) / 0x800 = 0x4

SEG0.SIZE = 0x04

Conclusion: Program the PAGEPROT0 = 0x3C041808.

9.7.3 Read function examples

There are two possible read accesses:

- Read one single word: simple read as if SRAM memory: read the desired flash address and get read data on the bus.
- MASSREAD command: read the full flash memory and compare with expected content.

There are two ways of using MASSREAD:

- Full flash contains a fixed 32-bit pattern: indicate the expected pattern (value to be compared with each read value inside flash) in the flash controller DATA register and check the READOK flag in the flash controller interrupt register once the command is completed.
- Else: request a MASSREAD command without specifying any expected read value and check the LFSRVAL register once the command is completed. This LFSRVAL register contains a signature of the memory read.

MASSREAD sequence:

– Write in the flash controller DATA register the expected value (if MASSREAD is used in combination with the READOK flag).
– Write the MASSREAD command (0x55) in the flash controller COMMAND register.
– Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
– Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
– Then, wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command is completed.
– Check the READOK flag (expected high) in the IRQSTAT register or the LFSRVAL register value to ensure flash memory content is the expected result.
– Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register.

9.7.4 Erase function examples

The flash controller allows erasure of one page.

ERASE sequence (erase one page):

– Write the page address to be erased by writing in the flash controller ADDRESS register the following value:
  ADDRESS[15:9] = XADR[9:3] = page address to erase
  ADDRESS[8:0] = 9’b0 (row and word addresses at zero).
– Write the ERASE command (0x11) in the flash controller COMMAND register.
– Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating command is taken into account and under execution.
– Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
– Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command is completed.
– Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register.
– After this command, the erased page contains only bits set to ‘1’.

9.7.5 Enabling protection example

The device offers three level of protection to prevent application cloning and or altering of application code. The different levels are described in the *Table 26*. It is important to notice that disabling of SWD access is an irreversible operation.

Table 26. System memory protection

Protection	UserKey[Data0]	UserKey[Data1]	Disable protection	Description
None	0xFFFFFFFF	0xFFFFFFFF	Not applicable	Debugger can access and modify flash memory and RAM content. This is the default configuration. All the bootloader commands are available.
Readout	0xAAAAAAAA	0xAAAAAAAA	Perform mass erase	Debugger cannot read or modify both flash memory and RAM content. The bootloader commands READ_MEMORY, WRITE_MEMORY and GO are disabled and MASS ERASE is available.
SWD	0xABACABAD	0xABACABAD	This selection is irreversible	Debugger connection is not possible. There is no possibility to access to the device via SWD and all the bootloader commands are disabled
Not specified	Any other value	Any other value	Not applicable	This configuration is forbidden and can lead to unrecoverable damage of the device

In order to activate the desired level of protection, the following KEYWRITE sequence should be used:

1. Write DATA0 (LSB key) and DATA1 (MSB key) registers with the value to program
2. Write 0xC7EF584D to DATA2 and 0xB3A21096 to DATA3
3. Write the KEYWRITE command (0xFF) in the flash controller COMMAND register
4. Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
5. Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
6. Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode), indicating that the command is completed.
7. Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register. The key are activated (and flash and RAM banks are protected) after a reset.

The secure bootloader capability offers a further security level for preventing unwanted control from external users.

Note: When the secure bootloader capability is enabled the flash and RAM contents are readable. The GO command of the bootloader is disabled and all the other bootloader commands are available. Only signed firmware can be executed.

9.7.6 Write page protection example

Example to write protect against accidental programming knowing the flash starts at address 0x1004_0000 and contains 256 pages of 2 Kbytes (pages 0 to 255).

For the address range 0x1004C000 to 0x1004FFFF

- Starting page: 0xC000 / 0x800 = 0x18
  SEG0[15:8]= 0x18 (OFFSET = 0x18)
- Number of pages: (0x10000 - 0xC000) / 0x800 = 0x8
  SEG0[7:0]= 0x08 (SIZE = 0x08)

And for the address ranges 0x1005E000 to 0x1005FFFF.

- Starting page: 0x1E000 / 0x800 = 0x3C
  SEG1[15:8] = 0x3C (OFFSET = 0x3C)
- Number of pages: (0x20000 - 0x1E000) / 0x800 = 0x4
  SEG1[7:0] = 0x04 (SIZE = 0x04)

Conclusion: Program the PAGEPROT0 = 0x3C041808.

9.7.7 Write function examples

The flash controller allows writing one word (WRITE), up to 4 words (BURSWRITE) or the full main flash memory (with a single fixed word).

Note: As a write can only program to ‘0’ on bits already set to ‘1’, it is necessary to erase the page and request that the bits be set to ‘1’ (instead of ‘0’) in order to re-write to ‘0’.

WRITE sequence:

- Indicate the location to write by filling the flash controller ADDRESS register with the targeted address (page, row and word number)
- Write the value to program in the flash controller DATA register.
- Write the WRITE command (0x33) in the flash controller COMMAND register.
- Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
- Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
- Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command is completed.
- Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register.

BUSRTWRITE sequence:

– Indicate the location to write by filling the flash controller ADDRESS register with the targeted address of the first data to write (page, row and word number). DATA0 is written ad ADDRESS, DATA1 at ADDRESS+1, etc.
– Write the values to program in the flash controller DATA0-3 registers. To write less than four words, write 0xFFFFFFFF in the unused DATA1-3 registers.
– Write the BURSTWRITE command (0xCC) in the flash controller COMMAND register.
– Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
– Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
– Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command is completed.
– Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register.

9.7.8 Other command usage descriptions

OTPWRITE sequence:

– Write DATA0 register with the value to program (no burst write feature is available as only few bytes to be written once only)
– Write ADDRESS register according to the following rule:
ADDRESS[15:9] = don’t care (page number frozen by hardware on this command)
ADDRESS[8:6] = 0, 1, 2 or 3
ADDRESS[5:0] = full area possible (from 0x00 to 0x3F)
– Write the OTPWRITE command (0xEE) in the flash controller COMMAND register
– Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
– Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
– Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode), indicating that the command is completed.
– Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register.

Note: The OTP area can be filled until the last word (address 0x1000_1BFC) is written with a value different from 0xFFFF_FFFF and a system reset occurs.

10 DMA controller (DMA)

10.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory as well as memory to memory. Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.

The DMA has an arbiter for handling the priority between DMA requests.

10.2 DMA main features

- Eight independently configurable channels (requests)
- Each of the eight channels is connected to dedicated hardware DMA requests, software trigger is also supported on each channel. This configuration is done by software.
- Priorities between requests from channels of the DMA are software programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (request 1 has priority over request 2, etc.)
- Independent source and destination transfer size (byte, half word, word), emulating packing and unpacking. Source/destination addresses must be aligned on the data size.
- Support for circular buffer management
- 3 event flags (DMA Half Transfer, DMA Transfer complete and DMA Transfer Error) logically ORed together in a single interrupt request for each channel
- Memory-to-memory transfer (SRAM0/SRAM1).
- Peripheral-to-memory and memory-to-peripheral, and peripheral-to-peripheral transfers
- Access to SRAMs, APB0 and APB1 peripherals as source and destination
- Programmable number of data to be transferred: up to 65536

10.3 DMA functional description

The DMA controller performs direct memory transfer by sharing the system bus with the other masters of the device. The DMA request may stop the CPU access to the system bus for some bus cycles, when the CPU and DMA are targeting the same destination (memory or peripheral). The bus matrix implements round-robin scheduling, thus ensuring at least half of the system bus bandwidth (both to memory and peripheral) for the CPU.

10.3.1 DMA transactions

After an event, the peripheral sends a request signal to the DMA Controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA Controller accesses the peripheral, an Acknowledge is sent to the peripheral by the DMA Controller. The peripheral releases its request as soon as it gets the Acknowledge from the DMA Controller. Once the request is deasserted by the peripheral, the DMA Controller release the Acknowledge. If there are more requests, the peripheral can initiate the next transaction.

In summary, each DMA transfer consists of three operations:

- The loading of data from the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The storage of the data loaded to the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The post-decrementing of the DMA_CNDTRx register, which contains the number of transactions that have still to be performed.

10.3.2 Arbiter

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences.

The priorities are managed in two stages:

- Software: each channel priority can be configured in the DMA_CCRx register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: if 2 requests have the same software priority level, the channel with the lowest number gets priority versus the channel with the highest number. For example, channel 2 gets priority over channel 4.

10.3.3 DMA channels

Each channel can handle DMA transfer between a peripheral register located at a fixed address and a memory address. The amount of data to be transferred (up to 65535) is programmable. The register which contains the amount of data items to be transferred is decremented after each transaction.

Programmable data sizes

Transfer data sizes of the peripheral and memory are fully programmable through the PSIZE and MSIZE bits in the DMA_CCRx register.

Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented after each transaction depending on the PINC and MINC bits in the DMA_CCRx register. If incremented mode is enabled, the address of the next transfer is the address of the previous one incremented by 1, 2 or 4 depending on the chosen data size. The first transfer address is the one programmed in the DMA_CPARx/DMA_CMARx registers. During transfer operations, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel is configured in noncircular mode, no DMA request is served after the last transfer (that is once the number of data items to be transferred has reached zero). In order to reload a new number of data items to be transferred into the DMA_CNDTRx register, the DMA channel must be disabled.

Note: If a DMA channel is disabled, the DMA registers are not reset. The DMA channel registers (DMA_CCRx, DMA_CPARx and DMA_CMARx) retain the initial values programmed during the channel configuration phase.

In circular mode, after the last transfer, the DMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address vaules from the DMA_CPARx/DMA_CMARx registers.

Channel configuration procedure

The following sequence should be followed to configure a DMA channelx (where x is the channel number).

1. Set the peripheral register address in the DMA_CPARx register. The data is moved from/ to this address to/ from the memory after the peripheral event.
2. Set the memory address in the DMA_CMARx register. The data is written to or read from this memory after the peripheral event.
3. Configure the total number of data to be transferred in the DMA_CNDTRx register. After each peripheral event, this value is decremented.
4. Configure the channel priority using the PL[1:0] bits in the DMA_CCRx register
5. Configure data transfer direction, circular mode, peripheral & memory incremented mode, peripheral & memory data size, and interrupt after half and/or full transfer in the DMA_CCRx register
6. Activate the channel by setting the ENABLE bit in the DMA_CCRx register.

As soon as the channel is enabled, it can serve any DMA request from the peripheral connected on the channel.

Once half of the bytes are transferred, the half-transfer flag (HTIF) is set and an interrupt is generated if the Half-Transfer Interrupt Enable bit (HTIE) is set. At the end of the transfer, the Transfer Complete Flag (TCIF) is set and an interrupt is generated if the Transfer Complete Interrupt Enable bit (TCIE) is set.

Circular mode

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_CCRx register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This mode is called Memory to Memory mode.

If the MEM2MEM bit in the DMA_CCRx register is set, then the channel initiates transfers as soon as it is enabled by software by setting the Enable bit (EN) in the DMA_CCRx register. The transfer stops once the DMA_CNDTRx register reaches zero. Memory to Memory mode may not be used at the same time as Circular mode.

10.3.4 Programmable data width, data alignment and endians

When PSIZE and MSIZE are not equal, the DMA performs some data alignments as described in Table 27.

Table 27. Programmable data width and endian behavior (when PINC=MINC=1 and NDT[(1)]=4)

Port width Src => Dest	Source content addr / data	Transfer operation	Dest content addr / data
8 => 8	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	Read B0[7:0] @0x0 then Write B0[7:0] @0x0 Read B1[7:0] @0x1 then Write B1[7:0] @0x1 Read B2[7:0] @0x2 then Write B2[7:0] @0x2 Read B3[7:0] @0x3 then Write B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8 => 16	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	Read B0[7:0] @0x0 then Write 00B0[15:0] @0x0 Read B1[7:0] @0x1 then Write 00B1[15:0] @0x2 Read B2[7:0] @0x2 then Write 00B2[15:0] @0x4 Read B3[7:0] @0x3 then Write 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8 => 32	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	Read B0[7:0] @0x0 then Write 000000B0[31:0] @0x0 Read B1[7:0] @0x1 then Write 000000B1[31:0] @0x4 Read B2[7:0] @0x2 then Write 000000B2[31:0] @0x8 Read B3[7:0] @0x3 then Write 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16 => 8	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4/ @0x6 / B7B6	Read B1B0[15:0] @0x0 then Write B0[7:0] @0x0 Read B3B2[15:0] @0x2 then Write B2[7:0] @0x1 Read B5B4[15:0] @0x4 then Write B4[7:0] @0x2 Read B7B6[15:0] @0x6 then Write B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16 => 16	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4/ @0x6 / B7B6	Read B1B0[15:0] @0x0 then Write B1B0[15:0] @0x0 Read B3B2[15:0] @0x2 then Write B3B2[15:0] @0x2 Read B5B4[15:0] @0x4 then Write B5B4[15:0] @0x4 Read B7B6[15:0] @0x6 then Write B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16 => 32	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4/ @0x6 / B7B6	Read B1B0[15:0] @0x0 then Write 0000B1B0[31:0] @0x0 Read B3B2[15:0] @0x2 then Write 0000B3B2[31:0] @0x4 Read B5B4[15:0] @0x4 then Write 0000B5B4[31:0] @0x8 Read B7B6[15:0] @0x6 then Write 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32 => 8	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC/BFBEBDBC	Read B3B2B1B0[31:0] @0x0 then Write B0[7:0] @0x0 Read B7B6B5B4[31:0] @0x4 then Write B4[7:0] @0x1 Read BBBAB9B8[31:0] @0x8 then Write B8[7:0] @0x2 Read BFBEBDBC[31:0] @0xC then Write BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC

Table 27. Programmable data width and endian behavior (when PINC=MINC=1 and NDT[1]=4)

Port width	Source content	Transfer operation	Dest content
Src => Dest	**addr / data**		**addr / data**
32 => 16	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8/ BBBAB9B8 @0xC/BFBEBDBC	Read B3B2B1B0[31:0] @0x0 then Write B1B0[15:0] @0x0 Read B7B6B5B4[31:0] @0x4 then Write B5B4[15:0] @0x2 Read BBBAB9B8[31:0] @0x8 then Write B9B8[15:0] @0x4 Read BFBEBDBC[31:0] @0xC then Write BDBC[15:0] @0x6	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32 => 32	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 /BBBAB9B8 @0xC/BFBEBDBC	Read B3B2B1B0[31:0] @0x0 then Write B3B2B1B0[31:0] @0x0 Read B7B6B5B4[31:0] @0x4 then Write B7B6B5B4[31:0] @0x4 Read BBBAB9B8[31:0] @0x8 then Write BBBAB9B8[31:0] @0x8 Read BFBEBDBC[31:0] @0xC then Write BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 /BBBAB9B8 @0xC/BFBEBDBC

1. NDT = Number of data items to transfer.

Addressing an AHB peripheral that does not support byte or halfword write operations

When the DMA initiates an AHB byte or halfword write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus. So when the used AHB slave peripheral does not support byte or halfword write operations (when HSIZE is not used by the peripheral) and does not generate any error, the DMA writes the 32 HWDATA bits as shown in the two examples below:

- To write the halfword “0xABCD”, the DMA sets the HWDATA bus to “0xABCDABCD” with HSIZE = HalfWord
- To write the byte “0xAB”, the DMA sets the HWDATA bus to “0xABABABAB” with HSIZE = Byte

Assuming that the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take the HSIZE data into account, it transforms any AHB byte or halfword operation into a 32-bit APB operation in the following manner:

- an AHB byte write operation of the data “0xB0” to 0x0 (or to 0x1, 0x2 or 0x3) is converted to an APB word write operation of the data “0xB0B0B0B0” to 0x0
- an AHB halfword write operation of the data “0xB1B0” to 0x0 (or to 0x2) is converted to an APB word write operation of the data “0xB1B0B1B0” to 0x0

For instance, if you want to write the APB backup registers (16-bit registers aligned to a 32-bit address boundary), you must configure the memory source size (MSIZE) to “16-bit” and the peripheral destination size (PSIZE) to “32-bit”.

10.3.5 Error management

A DMA transfer error can be generated by reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or a write access, the faulty channel is automatically disabled through a hardware clear of its EN bit in the corresponding

Channel configuration register (DMA_CCRx). The channel's transfer error interrupt flag (TEIF) in the DMA_IFR register is set and an interrupt is generated if the transfer error interrupt enable bit (TEIE) in the DMA_CCRx register is set.

10.3.6 Interrupts

An interrupt can be produced on a half-transfer, transfer complete or transfer error for each DMA channel. Separate interrupt enable bits are available for flexibility.

Table 28. DMA interrupt requests

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

10.3.7 DMA request mapping

A DMAMUX is present in the STM32WL30xx/31xx/33xx. This selects which requester is connected to which DMA channel. See *Table 31: DMAMUX: assignment of multiplexer inputs to resource* for details.

10.4 DMA registers

Refer to *Section 1.3: Acronyms* for a list of abbreviations used in register descriptions.

The peripheral registers must be accessed by words (32-bit) only.

10.4.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEIF8	HTIF8	TCIF8	GIF8	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31, 27, 23, 19, 15, 11, 7, 3 **TEIFx:** Channel x transfer error flag (x = 1..8)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No transfer error (TE) on channel x

1: A transfer error (TE) occurred on channel x

Bits 30, 26, 22, 18, 14, 10, 6, 2 **HTIFx:** Channel x half transfer flag (x = 1..8)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No half transfer (HT) event on channel x

1: A half transfer (HT) event occurred on channel x

Bits 29, 25, 21, 17, 13, 9, 5, 1 **TCIFx:** Channel x transfer complete flag (x = 1..8)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No transfer complete (TC) event on channel x

1: A transfer complete (TC) event occurred on channel x

Bits 28, 24, 20, 16, 12, 8, 4, 0 **GIFx:** Channel x global interrupt flag (x = 1..8)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No TE, HT or TC event on channel x

1: A TE, HT or TC event occurred on channel x

10.4.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTEIF 8	*CHTIF 8*	*CTCIF 8*	*CGIF8*	*CTEIF 7*	*CHTIF 7*	*CTCIF7*	*CGIF7*	*CTEIF6*	*CHTIF6*	*CTCIF6*	*CGIF6*	*CTEIF5*	*CHTIF5*	*CTCIF5*	*CGIF5*
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF 4	CHTIF 4	CTCIF 4	CGIF4	CTEIF 3	CHTIF 3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 31, 27, 23, 19, 15, 11, 7, 3 **CTEIFx:** Channel x transfer error clear (x = 1..8)
This bit is set and cleared by software.
0: No effect
1: Clears the corresponding TEIF flag in the DMA_ISR register

Bits 30, 26, 22, 18, 14, 10, 6, 2 **CHTIFx:** Channel x half transfer clear (x = 1..8)
This bit is set and cleared by software.
0: No effect
1: Clears the corresponding HTIF flag in the DMA_ISR register

Bits 29, 25, 21, 17, 13, 9, 5, 1 **CTCIFx:** Channel x transfer complete clear (x = 1..8)
This bit is set and cleared by software.
0: No effect
1: Clears the corresponding TCIF flag in the DMA_ISR register

Bits 28, 24, 20, 16, 12, 8, 4, 0 **CGIFx:** Channel x global interrupt clear (x = 1..8)
This bit is set and cleared by software.
0: No effect
1: Clears the GIF, TEIF, HTIF and TCIF flags in the DMA_ISR register

10.4.3 DMA channel x configuration register (DMA_CCRx) (x = 1..8, where x = channel number)

Address offset: 0x008 + 0d20 × (channel number - 1)

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **MEM2MEM:** Memory to memory mode

This bit is set and cleared by software.

0: Memory to memory mode disabled
1: Memory to memory mode enabled

Bits 13:12 **PL[1:0]:** Channel priority level

These bits are set and cleared by software.

00: Low
01: Medium
10: High
11: Very high

Bits 11:10 **MSIZE[1:0]:** Memory size

These bits are set and cleared by software.

00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bits 9:8 **PSIZE[1:0]:** Peripheral size

These bits are set and cleared by software.

00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bit 7 **MINC:** Memory increment mode

This bit is set and cleared by software.

0: Memory increment mode disabled
1: Memory increment mode enabled

Bit 6 **PINC:** Peripheral increment mode

This bit is set and cleared by software.

0: Peripheral increment mode disabled
1: Peripheral increment mode enabled

Bit 5 **CIRC:** Circular mode
This bit is set and cleared by software.
0: Circular mode disabled
1: Circular mode enabled

Bit 4 **DIR:** Data transfer direction
This bit is set and cleared by software.
0: Read from peripheral
1: Read from memory

Bit 3 **TEIE:** Transfer error interrupt enable
This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 2 **HTIE:** Half transfer interrupt enable
This bit is set and cleared by software.
0: HT interrupt disabled
1: HT interrupt enabled

Bit 1 **TCIE:** Transfer complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 0 **EN:** Channel enable
This bit is set and cleared by software.
0: Channel disabled
1: Channel enabled

10.4.4 DMA channel x number of data register (DMA_CNDTRx) (x = 1..8, where x = channel number)

Address offset: 0x00C + 0d20 × (channel number - 1)

Reset value: 0x0000 0000

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

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

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

Bits 15:0 **NDT[15:0]:** Number of data to transfer

Number of data to be transferred (0 up to 65535). This register can only be written when the channel is disabled. Once the channel is enabled, this register is read-only, indicating the remaining bytes to be transmitted. This register decrements after each DMA transfer.

Once the transfer is completed, this register can either stay at zero or be reloaded automatically by the value previously programmed if the channel is configured in auto-reload mode.

If this register is zero, no transaction can be served whether the channel is enabled or not.

10.4.5 DMA channel x peripheral address register (DMA_CPARx) (x = 1..8, where x = channel number)

Address offset: 0x010 + 0d20 × (channel number - 1)

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 31:0 **PA[31:0]:** Peripheral address

Base address of the peripheral data register from/to which the data is read/written.

When PSIZE is 01 (16-bit), the PA[0] bit is ignored. Access is automatically aligned to a half-word address.

When PSIZE is 10 (32-bit), PA[1:0] are ignored. Access is automatically aligned to a word address.

10.4.6 DMA channel x memory address register (DMA_CMARx) (x = 1..8, where x = channel number)

Address offset: 0x014 + 0d20 × (channel number - 1)

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 31:0 **MA[31:0]:** Memory address

Base address of the memory area from/to which the data is read/written.

When MSIZE is 01 (16-bit), the MA[0] bit is ignored. Access is automatically aligned to a half-word address.

When MSIZE is 10 (32-bit), MA[1:0] are ignored. Access is automatically aligned to a word address.

10.5 DMA register map

The following table gives the DMA register map and the reset values.

Table 29. DMA register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMA_ISR	TEIF8	HTIF8	TCIF8	GIF8	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DMA_IFCR	*CTEIF8*	*CHTIF8*	*CTCIF8*	*CGIF8*	*CTEIF7*	*CHTIF7*	*CTCIF7*	*CGIF7*	*CTEIF6*	*CHTIF6*	*CTCIF6*	*CGIF6*	*CTEIF5*	*CHTIF5*	*CTCIF5*	*CGIF5*	*CTEIF4*	*CHTIF4*	*CTCIF4*	*CGIF4*	*CTEIF3*	*CHTIF3*	*CTCIF3*	*CGIF3*	*CTEIF2*	*CHTIF2*	*CTCIF2*	*CGIF2*	*CTEIF1*	*CHTIF1*	*CTCIF1*	*CGIF1*
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	DMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	DMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CMAR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	DMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_CMAR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x044	DMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	DMA_CPAR4	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	DMA_CMAR4	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved																																
0x058	DMA_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	DMA_CNDTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DMA_CPAR5	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	DMA_CMAR5	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved																																
0x06C	DMA_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	DMA_CNDTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	DMA_CPAR6	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	DMA_CMAR6	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved																																
0x080	DMA_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x084	**DMA_CNDTR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	**DMA_CPAR7**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	**DMA_CMAR7**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	Reserved																																
0x094	DMA_CCR8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	**DMA_CNDTR8**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C	**DMA_CPAR8**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	**DMA_CMAR8**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4 to 0x31C	Reserved																																

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

11 DMA request multiplexer (DMAMUX)

11.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until it is served by the DMA controller which generates a DMA acknowledge signal and the corresponding DMA request signal is de-asserted.

For simplicity, the functional description of the DMA request/acknowledge protocol and its associated control signals are abstracted in this document and globally named as DMA request lines. The DMA controller response signals are not shown in figures nor described in the text.

The DMAMUX request multiplexer allows routing a DMA request line between the peripherals and the DMA controller of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line.

11.2 DMAMUX main features

- 8-channel programmable DMA request line multiplexer output.
- Per DMA request line multiplexer channel output:
  - 26 input DMA request linesfrom peripherals.
  - One DMA request line output.

11.3 DMAMUX implementation

11.3.1 DMAMUX instantiation

DMAMUX is instantiated with the following hardware configuration parameters.

Table 30. DMAMUX instantiation

Feature	DMAMUX
Number of DMAMUX output request channels	8
Number of DMAMUX request generator channels	26
Number of DMAMUX request trigger inputs	1
Number of DMAMUX synchronization inputs	1
Number of DMAMUX peripheral request inputs	1

11.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

Table 31. DMAMUX: assignment of multiplexer inputs to resource

DMA request MUX input	Resource	DMA request MUX input	Resource
1	Reserved	14	LPUART_RX
2	SPI3_RX	15	LPUART_TX
3	SPI3_TX	16	ADC_CH0[1] (DS channel)
4	SPI1_RX[1]	17	TIM2_TRG
5	SPI1_TX[1]	18	TIM2_CH1
6	AES_OUT	19	TIM2_CH2
7	AES_IN	20	TIM2_CH3
8	I2C1_RX[1]	21	TIM2_CH4
9	I2C1_TX[1]	22	TIM2_UP
10	I2C2_RX[2]	23	TIM16_CH1
11	I2C2_TX[2]	24	TIM16_UP
12	USART_RX	25	TIM16_TRG
13	USART_TX	26	DAC

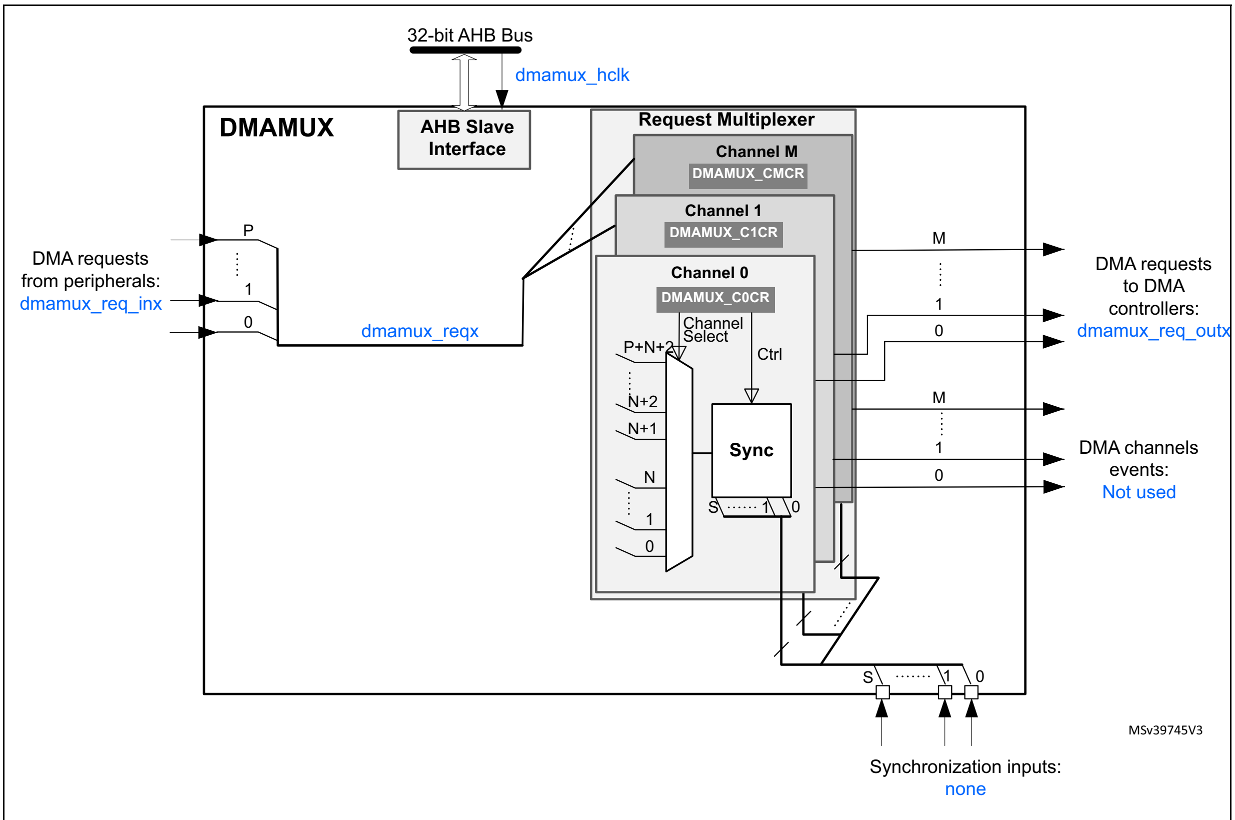
1. ADC, I2C1, SPI1 are available only on STM32WL31xx/33xx devices.
2. I2C2 is available only on STM32WL33xx devices.

11.4 DMAMUX functional description

11.4.1 DMAMUX block diagram

Figure 21 shows the DMAMUX block diagram.

Figure 21. DMAMUX block diagram



The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (dmamux_reqx) from peripherals (dmamux_req_inx) from .
- DMAMUX requests outputs to channels of DMA controllers (dmamux_req_outx).

11.4.2 DMAMUX channels

A DMAMUX channel is a DMAMUX request multiplexer channel which may include, depending on the selected input of the request multiplexer.

A DMAMUX request multiplexer channel is connected and dedicated to one single DMA controller(s) channel.

Channel configuration procedure

The following sequence should be followed to configure both a DMAMUX x channel and the related DMA channel y:

11.4.3 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named as DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced from the peripherals .

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the 8-bit DMAREQ_ID field in the DMAMUX_CxCR register.

Note: The null value in the field DMAREQ_ID corresponds to no DMA request line selected.

A same non-null DMA_REQ_ID value shall not be programmed to different x and y DMAMUX request multiplexer channels (via DMAMUX1_CxCR and DMAMUX CyCR).

It is not allowed to configure a same non-null DMAREQ_ID to two different channels of the DMAMUX request line multiplexer.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

11.5 DMAMUX registers

Refer to the table about register boundary addresses for the DMAMUX base address.

The registers can only be accessed by words (32-bits).

Note: ADC, I2C1, SPI1 are available only on STM32WL31xx/33xx devices.
I2C2 is available only on STM32WL33xx devices.
Any reference to these IPs on DMAMUX registers is valid within this perimeter.

11.5.1 DMAMUX request line multiplexer Channel x Configuration Register (DMAMUX_CxCR)

Address offset: 0x04 * x (x = 0 to 7)
Reset value: 0x0000 0000

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

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

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

Bits 4:0 **DMAREQ_ID[4:0]**: DMA REQuest IDentification
Selects the input DMA request. C.f. the DMAMUX table about assignments of multiplexer inputs to resources.

11.6 DMAMUX register map

The following table summarizes the DMAMUX registers and reset values. Refer to the register boundary address table for the DMAMUX register base address.

Table 32. DMAMUX register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMAMUX_C0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x004	DMAMUX_C1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x008	DMAMUX_C2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x00C	DMAMUX_C3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x010	DMAMUX_C4CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x014	DMAMUX_C5CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x018	DMAMUX_C6CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0

Table 32. DMAMUX register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	DMAMUX_C7CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x020 - 0x3E8	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

12 Analog digital converter (ADC)

The STM32WL31xx and STM32WL33xx embed a 12-bit ADC. The ADC is not available on STM32WL30xx devices.

The ADC consists in a 12-bit successive approximation analog-to-digital converter (SAR) with 2 x 8 multiplexed channels allowing measurements of up to eight external sources and up to three internal sources.

12.1 Features

- Conversion frequency is up to 1 Msps.
- Three input voltage ranges are supported (0 → 1.2V, 0 → 2.4 V, 0 → 3.6 V)
- Up to eight analog single ended channels or four analog differential inputs or a mix of both.
- Temperature sensor conversion
- Battery level conversion up to 3.6 V.
- Continuous or single acquisition.
- ADC Mode conversion only available, programmable in continuous or single mode.
- ADC Down Sampler for multi-purpose applications to improve analog performance while off-loading the CPU (ratio adjustable from 1 to 128).
- A watchdog feature to inform when data is outside thresholds.
- DMA capability.
- Interrupt sources with flags.

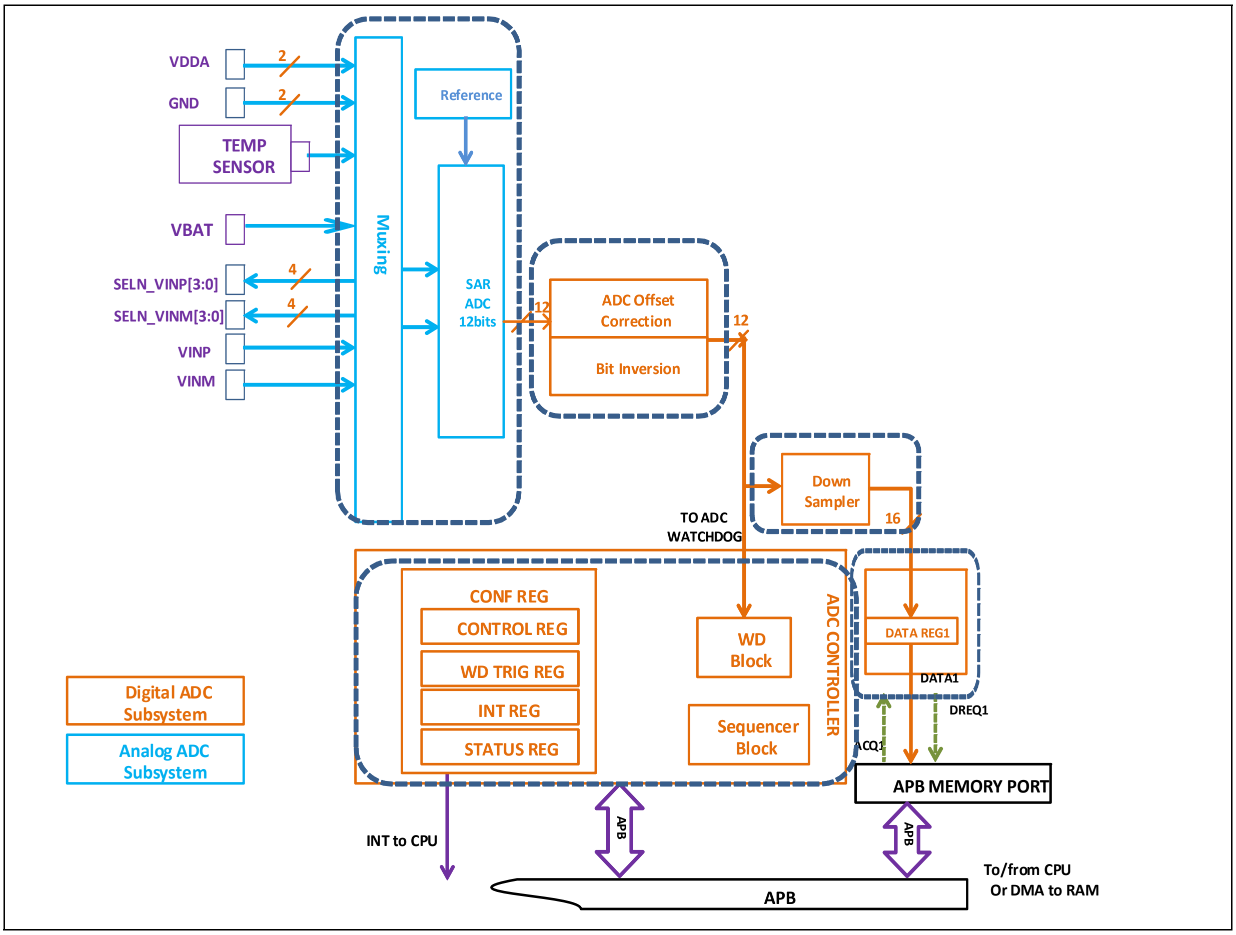
12.2 ADC presentation

The *Figure 22* shows the top level diagram of the ADC.

The analog ADC is designed to support sixteen possible inputs:

- External signals through ADC_VINPx and ADC_VINMx, where x=0,1,2 or 3
  - Up to 4 differential inputs
  - Up to 8 single-ended inputs one single ended input to interface with a temperature sensor in a range up to 1.2 V
- One single ended input connected to VBAT for battery level detector in a range up to 3.6 V
- Two single ended inputs connected to GND for calibration
- Two single ended inputs connected to 1.2 V for calibration
- One single ended input connected to Temperature Sensor

Figure 22. ADC top level diagram



The input of the data path can come from the analog ADC through the possible inputs mentioned previously.

The conversion data path can go through a down sampler (for static or low frequency input signals).

Caution: Do not change the configuration registers related to the function in use. Any change done by the user on the different bits are applied immediately, with an immediate effect on the on-going process (conversion, decimator filter or downsampler). This action can lead to unexpected results.

Caution: For VBAT < 2.7V, the IO Booster needs to be activated to maintain linearity.

12.2.1 Temperature sensor subsystem (STM32WL31xx/33xx only)

The temperature sensor can be used to measure the junction temperature (Tj) of the device. The temperature sensor is internally connected to the ADC input channels which are used to convert the sensor output voltage to a digital value.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation. The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production. During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. In this way the temperature can be calculated with this formula:

$$\text{Temperature in celsius} = \frac{\text{Cmeas} - \text{C30} + \text{TCK}}{10}$$

Where:

TCK is the chuck temperature in 0.1°C (e.g. 30°C = 300) readable @0x10001E5C.

C30 is the temperature sensor calibration value acquired at 30°C readable @0x10001E60.

Cmeas is the actual temperature sensor output value converted by ADC.

Note: ADC gain and offset calibrations are left as default values, 0xFFF.

12.2.2 ADC input modes conversion

The ADC is designed to deliver a digital value corresponding to the ratio between the voltage applied on the converted channel and the reference voltage, VDDA. Note that

VDDA is also the ADC's power supply. The formula for ADC digital converted data after calibration and offset is the following:

- Single ended input mode

  Code = Integer(Slope * VIN) [clamped at 4095] where Slope for single ended input mode has the following value:

  3.6 V mode: Slope = 4096/3.6 [calibrated gain = 1/3]

  2.4 V mode: Slope = 4096/2.4 [calibrated gain = 1/2]

  1.2 V mode: Slope = 4096/1.25 [calibrated gain =0.96, gain clamped at 1]

- Differential input mode

  Code = Integer(Slope * (VINP - VINN)) + 2048 [clamped at 4095] where Slope for differential input mode has the following value:

  3.6 V mode: Slope = 2048/3.6 [calibrated gain = 1/3]

  2.4 V mode: Slope = 2048/2.4 [calibrated gain = 1/2]

  1.2 V mode: Slope = 2048/1.25 [calibrated gain =0.96, gain clamped at 1]

Steady state input impedance

As the input nature of the ADC is a switched-capacitor, its steady state input impedance is defined as the impedance seen in DC. It depends only on the analog sampling frequency, Fs, and the input capacitor, Cin: Zin = 1/(Cin*Fs).

Input signal sampling transient response

As represented on *Figure 24*, the analog signal path consists of a series resistance (Rext) between source and pin, the internal switch resistor (Rin) and the internal sampling capacitor (Cin). The charging of the capacitor is controlled by Rin. When there is Rext in series, the effective value of charging of Cin is governed by Rin+Rext. So the charging time constant iscome (Rin+Rext)*Cin and the necessary time to reach a given accuracy is longer. The ADC sampling time Tsw is a function of ADC frequency which is 1/Ts as shown in *Figure 23*.

Figure 23. ADC sampling time Tsw and sampling period Ts

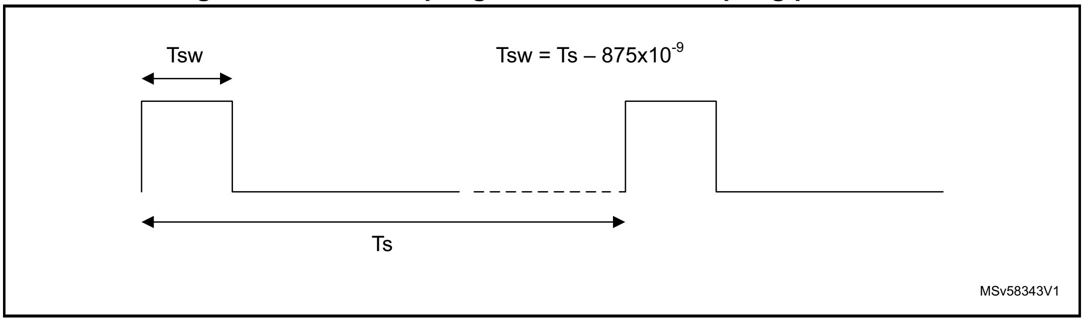
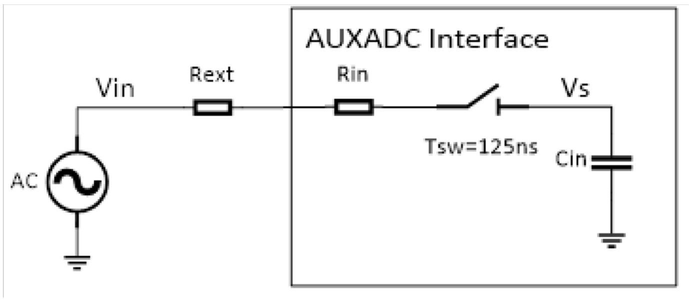


Figure 24. Effect of analog source resistance



Knowing that: Rin = 550 ohms, Cin = 4 pF, and imposing a maximum sampling error of 1/2 LSB, we can determine the maximum input resistance as below:

ε= (Vs-Vin)/Vin = $-e^{-t/RC}$

ln|ε| ≤ t/(Rext+Rin)*Cin

(Rext+Rin)*Cin ≤ t/ ln|ε|

(Rext+Rin)*Cin ≤ 125e-9/ ln(122e-6)

(Rext+Rin)*Cin ≤ 14 ns

Rext ≤ 14 ns/4pF - 550Ω

Rext ≤ 2950Ω

where: |ε| ≤ 1/213

|ε| ≤ 122e-6

12.2.3 Down sampler (DS)

This Down Sampler is a simple averaging filter which can divide the ADC frequency by 1 to 128 by power of 2.

The goal is to handle multiple ADC samples and average then into a single data with increased data width ranging from 12-bit to 16-bit.

The Down Sampler increases the data precision but reduces the output data rate.

Note: A constraint on the ratio between APB system clock (F_{PCLK}) and the output data rate (DRout) must be respected:

- In ADC mode, the ratio to respect is
  F_{PCLK} / DRout ≥ 4.

Example: F_{PCLK} must be at least 2 MHz to have a DRout = 500 kHz.

If the DMA is not used to get the data output by the decimation filter path, the CPU needs to be clocked at a frequency ratio high enough (taking into account bus matrix latency) to avoid missing samples.

12.3 Interrupts

There are 5 maskable interrupts generated by the ADC block. These interrupts are combined to produce one single interrupt output which is the only interrupt line from the ADC to the CPU.

Table 33. ADC interrupt requests

Interrupt event	Event flag	Interrupt / flag clearing method	Interrupt enable control bit
ADC end of conversion	EOC_IRQ	Write 1 on EOC_IRQ bit	EOC_IRQ_ENA
Down Sampler end of conversion	EODS_IRQ	Write 1 on EODS_IRQ bit	EODS_IRQ_ENA
End of conversion sequence	EOS_IRQ	Write 1 on EOS_IRQ bit	EOS_IRQ_ENA
Analog Watchdog event	AWD_IRQ	Write 1 on AWD_IRQ bit	AWD_IRQ_ENA
Down Sampler overrun	OVR_DS_IRQ	Write 1 on OVR_DS_IRQ bit	OVR_DS_IRQ_ENA

12.4 DMA interface

The ADC has one DMA channel interface to get Down Sampler data output value.

The DMA feature is enabled by software through CONF register by DMA_DS_ENA bit.

When DMA feature is disabled, the data can be read by the CPU through the corresponding APB register.

12.5 ADC mode

ADC mode is the only conversion mode available for STM32WL30xx/31xx/33xx devices. As a consequence no concurrent mode is also available.

The input signal can come from:

- 8 single external channels (4 positive + 4 negative, or 4 differential when coupled).
- VBAT: negative single ended input to 3,6 V
- Temperature Sensor: positive single ended input to 1,2 V

The conversion can be continuous or single mode.

12.5.1 ADC mode overview

Presentation

The ADC mode has the following characteristics:

- The input in the ADC mode can be the eight external channels and the two internal sources (VBAT and Temperature sensor).
- The data path goes from the ADC to the down sampler.
- The converted data is output in the DS_DATAOUT register.
- The output data rates are in the range 117 sps to 1 Msps.
- The 12-bit converted data can be extended up to 16-bit data thanks to the down sampler. However, in this case, the output data rate is decreased.
- A regular sequence of conversion can be executed in single of continuous mode.
  - A regular sequence consists in chaining ADC conversions on any ADC input channel and in any order.
  - A regular sequence can chain up to 16 conversions.
  - The source of the input for each conversion of the sequence is selected through SEQx bit field in SEQ_1 and SEQ_2 registers.
  - This regular sequence can be run once or repeated continuously by setting the CONT bit in CONF register.

ADC mode usage

This paragraph describes the process to use the ADC mode:

- Power on the ADC if not yet done by setting the ADC_ON_OFF bit in the CTRL register.
- Program the targeted data rate through SAMPLE_RATE and DS_CONF registers.
- Program the input voltage selections through SWITCH register.
- Program the COMP_1 to COMP_4 and the COMP_SEL registers.
- Program the ADC mode through the OP_MODE bit field in the CONF register.
- Program the targeted regular sequence (up to 16 chained conversions) through SEQ_1 and SEQ_2 registers.
- Specify the length of the sequence in SEQ_LEN bit field in CONF (from 0 for one conversion to 0xF for sixteen conversions).

Note: To have more than one conversion, ensure the bit SEQUENCE is well at 1 in CONF register.

- Program the CONT bit and the SEQ_LEN bit field in the CONF register, considering SEQUENCE bit is always set) depending on the wished sequence:
  - CONT = 0 and SEQ_LEN = 0 to have a single conversion on a single channel.
  - CONT = 0 and SEQ_LEN > 0 to have a single run of a sequence chaining several conversions on different channels/sources.
  - CONT = 1 and SEQ_LEN = 0 to have a continuous conversion of a single channel/source.
  - CONT = 1 and SEQ_LEN > 0 to have a continuous run of sequence chaining several conversions on different channels/sources.
- Launch the programmed regular sequence by setting the START_CONV bit in CTRL register.
- Each time a data is available at the output of the Sown Sampler, the data is stored in the DS_DATAOUT register and the EODS flag is set (as analog mode goes through the Down Sampler).
- To get the converted values:
  - Either the DMA is enabled on DS data path (through DMA_DS_ENA bit in CONF register) and DMA copies the converted data in RAM at the end of each data conversion
  - Or the software has enabled the EODS_IRQ interrupt and is able to get the data from DS_DATAOUT register before a new converted data is generated.

Note: If the CPU does not manage to get the converted data before a new converted data is generated, the OVR_DS_IRQ flag is raised to inform a data has been lost.

The software can program the hardware behavior in case of overrun through the OVR_DS_CFG bit in CONF register:

if 0, the previous data is kept, the new one is lost.

if 1, the previous data is lost, the new one is kept.

- Each time the regular sequence is completed, the EOS_IRQ flag is raised (and may generate an interrupt if enabled).
- If the sequence is a single sequence (SEQ_LEN=0), the ADC stops at the end of the sequence and does not restart until START_CONV bit is not set again.
- The data conversion goes on until the software stops it by setting the STOP_OP_MODE bit in the CTRL register: in this case, the conversion stops immediately and on-going conversion data is not issued.

12.6 ADC registers description

12.6.1 Version register (VERSION_ID)

Address offset: 0x00

Reset value: 0x0000 0021

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

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

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

Bit 7:0 **VERSION_ID[7:0]**: version of the embedded IP.

12.6.2 ADC configuration register (CONF)

Address offset: 0x04

Reset value: 0x0002 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAMPLE_RATE_MSB			Res.	ADC_CONT_1V2	BIT_INVERT_DIFF	BIT_INVERT_SN	Res.
								rw				rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVR_DS_CFG	Res.	DMA_DS_ENA	SAMPLE_RATE[1:0]		Res.	Res.	Res.	Res.	SMPS_SYNCHRO_ENA	SEQ_LEN[3:0]				SEQUENCE	CONT
rw		rw	rw						rw	rw				rw	rw

Bits 31: 24 Reserved

Bits 23:21 **SAMPLE_RATE_MSB**: sample rate most significant bit
This field is an extension of SAMPLE_RATE definition in bits 12,11 of CONF register. It impacts the conversion rate of ADC (F_ADC). See SAMPLE_RATE bits for the full description

Bit 20 Reserved, must be kept at reset value.

Bit 19 **ADC_CONT_1V2**: select the input sampling method:
- 0: Sampling time is 125 ns regardless of the sampling period.
- 1: Sampling time is a function of the sampling period.

Bit 18 **BIT_INVERT_DIFF**: invert bit to bit the ADC data output (1's complement) when a differential input is connected to the ADC:
- 0: no inversion (default)
- 1: enable the inversion

Bit 17 **BIT_INVERT_SN**: invert bit to bit the ADC data output (1’s complement) when a single negative input is connected to the ADC:
- 0: no inversion (default)
- 1: enable the inversion

Bit 16 Reserved, must be kept at reset value.

Bit 15 **OVR_DS_CFG**: Down Sampler overrun configuration:
- 0: the previous data is kept, the new one is lost
- 1: the previous data is lost, the new one is kept

Bit 14 Reserved, must be kept at reset value.

Bit 13 **DMA_DS_EN**: enable the DMA mode for the Down Sampler data path:
- 0: DMA mode is disabled
- 1: DMA mode is enabled

Bits 12:11 **SAMPLE_RATE[1:0]**: conversion rate of ADC (F_ADC):
F_ADC = F_ADC_CLK/(16 + 16*SAMPLE_RATE_MSB + 4*SAMPLE_RATE)
where F_ADC_CLK is the analog ADC clock frequency. By default F_ADC_CLK is 16 MHz frequency.

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **SMPS_SYNCHRO_ENA**: synchronize the ADC start conversion with a pulse generated by the SMPS:
- 0: SMPS synchronization is disabled for all ADC clock frequencies
- 1: SMPS synchronization is enabled

Note: SMPS_SYNCHRO_ENA must be 0 when PWRC_CR5.NOSMPS = 1.

Bits 5:2 **SEQ_LEN[3:0]**: number of conversions in a regular sequence:
- 0000: 1 conversion, starting from SEQ0
- 0001: 2 conversions, starting from SEQ0
- ...
- 1111: 16 conversions, starting from SEQ0

Bit 1 **SEQUENCE**: enable the sequence mode (active by default):
- 0: sequence mode is disabled, only SEQ0 is selected
- 1: sequence mode is enabled, conversions from SEQ0 to SEQx with x=SEQ_LEN

Note: clearing this bit is equivalent to SEQUENCE=1 and SEQ_LEN=0000. Ideally, this bit can be kept high as redundant with keeping high and setting SEQ_LEN=0000.

Bit 0 **CONT**: regular sequence runs continuously when ADC mode is enabled:
- 0: enable the single conversion: when the sequence is over, the conversion stops
- 1: enable the continuous conversion: when the sequence is over, the sequence starts again until the software sets the CTRL.STOP_OP_MODE bit.

12.6.3 ADC control register (CTRL)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **STOP_OP_MODE** (1): stop the on-going OP_MODE (ADC mode):
- 0: no effect
- 1: stop on-going ADC mode

Note: this bit is set by software and cleared by hardware.

Bit 1 **START_CONV** (1): generate a start pulse to initiate an ADC conversion:
- 0: no effect
- 1: start the ADC conversion

Note: this bit is set by software and cleared by hardware.

Bit 0 **ADC_ON_OFF**:
- 0: power off the ADC
- 1: power on the ADC

1. When setting the STOP_MODE_OP, the user has to wait around 10 us before to start a new ADC conversion by setting the START_CONV bit.

12.6.4 ADC input voltage switch selection register (SWITCH)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SE_VIN_7[1:0]		SE_VIN_6[1:0]		SE_VIN_5[1:0]		SE_VIN_4[1:0]		SE_VIN_3[1:0]		SE_VIN_2[1:0]		SE_VIN_1[1:0]		SE_VIN_0[1:0]	
																rw		rw		rw		rw		rw		rw		rw		rw	

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

Bits 15:14 **SE_VIN_7[1:0]**: input voltage for VINP[3]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 13:12 **SE_VIN_6[1:0]**: input voltage for VINP[2]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 11:10 **SE_VIN_5[1:0]**: input voltage for VINP[1]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 9:8 **SE_VIN_4[1:0]**: input voltage for VINP[0]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 7:6 **SE_VIN_3[1:0]**: input voltage for VINM[3] / VINP[3]-VINM[3]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 5:4 **SE_VIN_2[1:0]**: input voltage for VINM[2] / VINP[2]-VINM[2]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 3:2 **SE_VIN_1[1:0]**: input voltage for VINM[1] / VINP[1]-VINM[1]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 1:0 **SE_VIN_0[1:0]**: input voltage for VINM[0] / VINP[0]-VINM[0]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

12.6.5 Down sampler configuration register (DS_CONF)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:3 **DS_WIDTH[2:0]**: program the Down Sampler width of data output (DSDTATA)
- 000: DS_DATA output on 12-bit (default)
- 001: DS_DATA output on 13-bit
- 010: DS_DATA output on 14-bit
- 011: DS_DATA output on 15-bit
- 100: DS_DATA output on 16-bit
- 1xx: reserved

Bits 2:0 **DS_RATIO[2:0]**: program the Down Sampler ratio (N factor)
- 000: ratio = 1, no down sampling (default)
- 001: ratio = 2
- 010: ratio = 4
- 011: ratio = 8
- 100: ratio = 16
- 101: ratio = 32
- 110: ratio = 64
- 111: ratio = 128

12.6.6 ADC sequence programming 1 register (SEQ_1)

Address offset: 0x20

Reset value: 0x0000 0000

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
SEQ7[3:0]	SEQ6[3:0]	SEQ5[3:0]	SEQ4[3:0]	SEQ3[3:0]	SEQ2[3:0]	SEQ1[3:0]	SEQ0[3:0]
rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **SEQ7[3:0]**: channel number code for 8th conversion of the sequence.
See SEQ0 for code detail.

Bits 27:24 **SEQ6[3:0]**: channel number code for 7th conversion of the sequence.
See SEQ0 for code detail.

Bits 23:20 **SEQ5[3:0]**: channel number code for 6th conversion of the sequence.
See SEQ0 for code detail.

Bits 19:16 **SEQ4[3:0]**: channel number code for 5th conversion of the sequence.
See SEQ0 for code detail.

Bits 15:12 **SEQ3[3:0]**: channel number code for 4th conversion of the sequence.
See SEQ0 for code detail.

Bits 11:8 **SEQ2[3:0]**: channel number code for 3rd conversion of the sequence.
See SEQ0 for code detail.

Bits 7:4 **SEQ1[3:0]**: channel number code for second conversion of the sequence.
See SEQ0 for code detail.

Bits 3:0 **SEQ0[3:0]**: channel number code for first conversion of the sequence
- 0000: VINM[0] to ADC single negative input
- 0001: VINM[1] to ADC single negative input
- 0010: VINM[2] to ADC single negative input
- 0011: VINM[3] to ADC single negative input
- 0100: VINP[0] to ADC single positive input
- 0101: VINP[1] to ADC single positive input
- 0110: VINP[2] to ADC single positive input
- 0111: VINP[3] to ADC single positive input
- 1000: VINP[0]-VINM[0] to ADC differential input
- 1001: VINP[1]-VINM[1] to ADC differential input
- 1010: VINP[2]-VINM[2] to ADC differential input
- 1011: VINP[3]-VINM[3] to ADC differential input
- 1100: VBAT - Battery level detector
- 1101: Temperature sensor
- 111x: reserved

12.6.7 ADC sequence programming 2 register (SEQ_2)

Address offset: 0x24

Reset value: 0x0000 0000

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
SEQ15[3:0]	SEQ14[3:0]	SEQ13[3:0]	SEQ12[3:0]	SEQ11[3:0]	SEQ10[3:0]	SEQ9[3:0]	SEQ8[3:0]
rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **SEQ15[3:0]**: channel number code for 16th conversion of the sequence.
See SEQ8 for code detail.

Bits 27:24 **SEQ14[3:0]**: channel number code for 15th conversion of the sequence.
See SEQ8 for code detail.

Bits 23:20 **SEQ13[3:0]**: channel number code for 14th conversion of the sequence.
See SEQ8 for code detail.

Bits 19:16 **SEQ12[3:0]**: channel number code for 13th conversion of the sequence.
See SEQ8 for code detail.

Bits 15:12 **SEQ11[3:0]**: channel number code for 12th conversion of the sequence.
See SEQ8 for code detail.

Bits 11:8 **SEQ10[3:0]**: channel number code for 11th conversion of the sequence.
See SEQ8 for code detail.

Bits 7:4 **SEQ9[3:0]**: channel number code for 10th conversion of the sequence.
See SEQ8 for code detail.

Bits 3:0 **SEQ8[3:0]**: channel number code for 9th conversion of the sequence
- 0000: VINM[0] to ADC single negative input
- 0001: VINM[1] to ADC single negative input
- 0010: VINM[2] to ADC single negative input
- 0011: VINM[3] to ADC single negative input
- 0100: VINP[0] to ADC single positive input
- 0101: VINP[1] to ADC single positive input
- 0110: VINP[2] to ADC single positive input
- 0111: VINP[3] to ADC single positive input
- 1000: VINP[0]-VINM[0] to ADC differential input
- 1001: VINP[1]-VINM[1] to ADC differential input
- 1010: VINP[2]-VINM[2] to ADC differential input
- 1011: VINP[3]-VINM[3] to ADC differential input
- 1100: VBAT - Battery level detector
- 1101: temperature sensor
- 111x: reserved

12.6.8 ADC gain and offset correction 1 register (COMP_1)

Address offset: 0x28

Reset value: 0x0000 0555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSFFSET1[7:0]								GAIN1[11:0]											
												rw								rw											

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

Bits 19:12 **OFFSET1[7:0]**: first calibration point: signed offset compensation[7:0]

Bits 11:0 **GAIN1[11:0]**: first calibration point: gain AUXADC_GAIN_1V2[11:0]

12.6.9 ADC gain and offset correction 2 register (COMP_2)

Address offset: 0x2C

Reset value: 0x0000 0555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSFFSET2[7:0]								GAIN2[11:0]											
												rw								rw											

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

Bits 19:12 **OFFSET2[7:0]**: first calibration point: signed offset compensation[7:0]

Bits 11:0 **GAIN2[11:0]**: second calibration point: gain AUXADC_GAIN_1V2[11:0]

12.6.10 ADC gain and offset correction 3 register (COMP_3)

Address offset: 0x30

Reset value: 0x0000 0555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSFFSET3[7:0]								GAIN3[11:0]											
												rw								rw											

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

Bits 19:12 **OFFSET3[7:0]**: first calibration point: signed offset compensation[7:0]

Bits 11:0 **GAIN3[11:0]**: third calibration point: gain AUXADC_GAIN_1V2[11:0]

12.6.11 ADC gain and offset correction 4 register (COMP_4)

Address offset: 0x34

Reset value: 0x0000 0555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSFFSET4[7:0]								GAIN4[11:0]											
													rw							rw											

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

Bits 19:12 **OFFSET4[7:0]**: first calibration point: signed offset compensation[7:0]

Bits 11:0 **GAIN4[11:0]**: fourth calibration point: gain AUXADC_GAIN_1V2[11:0]

12.6.12 ADC gain and offset selection register (COMP_SEL)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET_GAIN8[1:0]		OFFSET_GAIN7[1:0]		OFFSET_GAIN6[1:0]		OFFSET_GAIN5[1:0]		OFFSET_GAIN4[1:0]		OFFSET_GAIN3[1:0]		OFFSET_GAIN2[1:0]		OFFSET_GAIN1[1:0]		OFFSET_GAIN0[1:0]	
														rw		rw		rw		rw		rw		rw		rw		rw		rw	

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **OFFSET_GAIN8[1:0]**: gain / offset used in ADC differential mode with Vinput range = 3.6 V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 15:14 **OFFSET_GAIN7[1:0]**: gain / offset used in ADC single positive mode with Vinput range = 3.6 V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 13:12 **OFFSET_GAIN6[1:0]**: gain / offset used in ADC single negative mode with Vinput range = 3.6 V. This field also selects the gain/offset for VBAT inp:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 11:10 **OFFSET_GAIN5[1:0]**: gain / offset used in ADC differential mode with Vinput range = 2.4 V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 9:8 **OFFSET_GAIN4[1:0]**: gain / offset used in ADC single positive mode with Vinput range = 2.4 V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 7:6 **OFFSET_GAIN3[1:0]**: gain / offset used in ADC single negative mode with Vinput range = 2.4 V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 5:4 **OFFSET_GAIN2[1:0]**: gain / offset used in ADC differential mode with Vinput range = 1.2 V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 3:2 **OFFSET_GAIN1[1:0]**: gain / offset used in ADC single positive mode with Vinput range = 1.2 V. This field also selects the gain/offset for Temperature Sensor input:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 1:0 **OFFSET_GAIN0[1:0]**: gain / offset used in ADC single negative mode with Vinput range = 1.2 V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

12.6.13 ADC watchdog threshold register (WD_TH)

Address offset: 0x3C

Reset value: 0x0FFF 0000

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

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

Bits 27:16 **WD_HT[11:0]**: analog watchdog high level threshold.

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

Bits 11:0 **WD_LT**[11:0]: analog watchdog low level threshold.

12.6.14 ADC watchdog configuration register (WD_CONF)

Address offset: 0x40

Reset value: 0x0000 0000

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

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

Bits 15:0 **AWD_CHX[15:0]**: analog watchdog channel selection to define which input channel(s) need to be guarded by the watchdog.

- Bit0: VINM[0] to ADC negative input
- Bit1: VINM[1] to ADC negative input
- Bit2: VINM[2] to ADC negative input
- Bit3: VINM[3] to ADC negative input
- Bit4: GND to ADC negative input
- Bit5: VBAT to ADC negative input
- Bit6: GND to ADC negative input
- Bit7: VDDA to ADC negative input
- Bit8: VINP[0] to ADC positive input
- Bit9: VINP[1] to ADC positive input
- Bit10: VINP[2] to ADC positive input
- Bit11: VINP[3] to ADC positive input
- Bit12: VBAT to ADC positive input
- Bit13: TEMP to ADC positive input
- Bit14: GND to ADC positive input
- Bit15: VDDA to ADC positive input

12.6.15 Down Sampler data out register (DS_DATAOUT)

Address offset: 0x44

Reset value: 0x0000 0000

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

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

Bits 15:0 **DS_DATA[15:0]**: contain the converted data at the output of the Down Sampler.

12.6.16 ADC Interrupt status register (IRQ_STATUS)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_DS_IRQ	AWD_IRQ	EOS_IRQ	Res.	EODS_IRQ	EOC_IRQ
																										rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **OVR_DS_IRQ**: set to indicate a Down Sampler overrun (at least one data is lost)

When read, provide the status of the interrupt:

– 0: no overrun occurred

– 1: overrun occurred

Writing this bit clears the status of the interrupt:

– 0: no effect

– 1: clear the interrupt

Bit 4 **AWD_IRQ**: set when an analog watchdog event occurs.

When read, provide the status of the interrupt:

– 0: no analog watchdog event occurred

– 1: analog watchdog event has occurred

Writing this bit clears the status of the interrupt:

– 0: no effect

– 1: clear the interrupt

Bit 3 **EOS_IRQ**: set when a sequence of conversion is completed.

When read, provide the status of the interrupt:

– 0: sequence of conversion is not completed

– 1: sequence of conversion is completed

Writing this bit clears the status of the interrupt:

– 0: no effect

– 1: clear the interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EODS_IRQ**: set when the Down Sampler conversion is completed.
When read, provide the status of the interrupt:
- 0: Down Sampler conversion is not completed
- 1: Down Sampler conversion is completed

Writing this bit clears the status of the interrupt:
- 0: no effect
- 1: clear the interrupt

Bit 0 **EOC_IRQ**: set when the ADC conversion is completed.
When read, provide the status of the interrupt:
- 0: ADC conversion is not completed
- 1: ADC conversion is completed

Writing this bit clears the status of the interrupt:
- 0: no effect
- 1: clear the interrupt

12.6.17 ADC Interrupt enable register (IRQ_ENABLE)

Address offset: 0x50

Reset value: 0x0000 0000

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

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **OVR_DS_IRQ_ENA**: Down Sampler overrun interrupt enable:
- 0: Down Sampler interrupt is disabled
- 1: Down Sampler interrupt is enabled

Bit 4 **AWD_IRQ_ENA**: analog watchdog interrupt enable:
- 0: analog watchdog interrupt is disabled
- 1: analog watchdog interrupt is enabled

Bit 3 **EOS_IRQ_ENA**: End of regular sequence interrupt enable:
- 0: EOS interrupt is disabled
- 1: EOS interrupt is enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EODS_IRQ_ENA**: End of conversion interrupt enable for the Down Sampler output:
- 0: EODF interrupt is disabled
- 1: EODF interrupt is enabled

Bit 0 **EOC_IRQ_ENA**: End of ADC conversion interrupt enable:
- 0: EOC interrupt is disabled
- 1: EOC interrupt is enabled

12.7 ADC register map

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the ADC base address location in STM32WL30xx/31xx/33xx devices.

Table 35. ADC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	VERSION_ID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VERSION_ID[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0x04	CONF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAMPLE_RATE_MSB			Res.	ADC_CONT_1V2	BIT_INVERT_DIFF	BIT_INVERT_SN	Res.	OVR_DS_CFG	Res.	DMA_DS_ENA	SAMPLE_RATE[1:0]		Res.	Res.	Res.	Res.	SMPS_SYNCHRO_ENA	SEQ_LEN[3:0]				SEQUENCE	CONT.
	Reset value									0	0	0		0	0	1		0		0	0	0					0	0	0	0	0	1	0
0x08	CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP_OP_MODE	START_CONV	ADC_ON_OFF
	Reset value																														0	0	0
0x14	SWITCH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SE_VIN_7[1:0]		SE_VIN_6[1:0]		SE_VIN_5[1:0]		SE_VIN_4[1:0]		SE_VIN_3[1:0]		SE_VIN_2[1:0]		SE_VIN_1[1:0]		SE_VIN_0[1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DS_CONF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS_WIDTH[2:0]			DS_RATIO[2:0]		
	Reset value																											0	0	0	0	0	0
0x20	SEQ_1	SEQ7[3:0]				SEQ6[3:0]				SEQ5[3:0]				SEQ4[3:0]				SEQ3[3:0]				SEQ2[3:0]				SEQ1[3:0]				SEQ0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	SEQ_2	SEQ15[3:0]				SEQ14[3:0]				SEQ13[3:0]				SEQ12[3:0]				SEQ11[3:0]				SEQ10[3:0]				SEQ9[3:0]				SE80[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	COMP_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET1[7:0]								GAIN1[11:0]											
	Reset value													0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1

Table 35. ADC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	COMP_2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET2[7:0]								GAIN2[11:0]											
	Reset value													0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1
0x30	COMP_3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET3[7:0]								GAIN3[11:0]											
	Reset value													0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1
0x34	COMP_4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET4[7:0]								GAIN4[11:0]											
	Reset value													0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1
0x38	COMP_SEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET_GAIN8[1:0]		OFFSET_GAIN7[1:0]		OFFSET_GAIN6[1:0]		OFFSET_GAIN5[1:0]		OFFSET_GAIN4[1:0]		OFFSET_GAIN3[1:0]		OFFSET_GAIN2[1:0]		OFFSET_GAIN01[1:0]		OFFSET_GAIN0[1:0]	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	WD_TH	Res.	Res.	Res.	Res.	WD_HT[11:0]												Res.	Res.	Res.	Res.	WD_LT[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x40	WD_CONF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD_CHX[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	DS_DATAOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS_DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_DS_IRQ	AWD_IRQ	EOS_IRQ	Res.	EODS_IRQ	EOC_IRQ
	Reset value																											0	0	0		0	0
0x50	IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_DS_IRQ_EN	AWD_IRQ_ENA	EOS_IRQ_ENA	Res.	EODS_IRQ_ENA	EOC_IRQ_ENA
	Reset value																											0	0	0		0	0

13 Comparator (COMP)

13.1 Introduction

The STM32WL33xx embeds one ultra-low power comparator COMP. The comparator and DAC are not available on STM32WL30xx and STM32WL31xx devices.

The comparator can be used for a variety of f unctions including.

- Wake-up from low-power mode triggered by an analog signal,
- Analog signal conditioning.
- Cycle-by-cycle current control loop when combined with a PWM output from a timer.

13.2 COMP main features

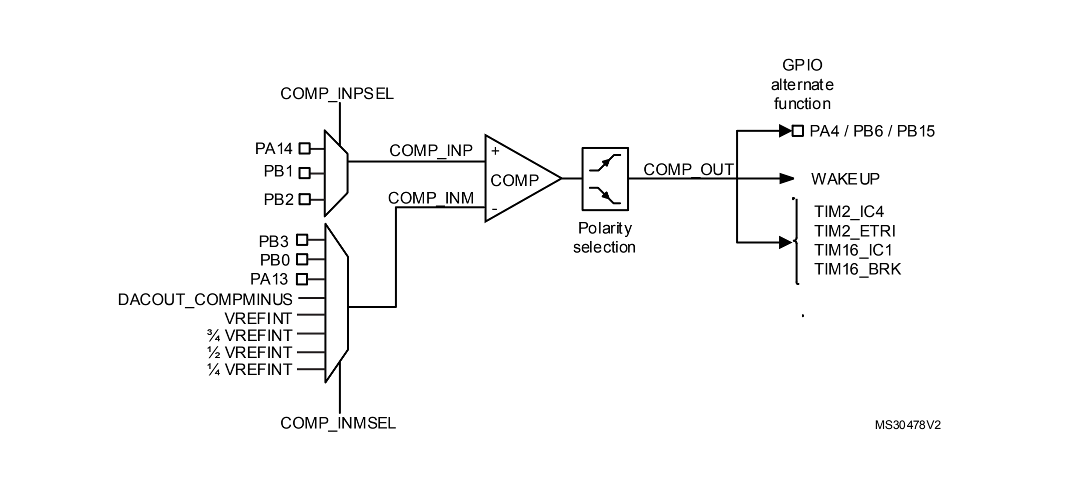
- The comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
  - DAC output
- Programmable hysteresis
- Programmable speed / consumption
- The outputs can be redirected to an I/O or to timer inputs for triggering:
  - Break events for fast PWM shutdown
  - Cycle-by-cycle current control, using OCREF_CLR_INT through ETR inputs
  - capture events
- Comparator output with blanking source
- The comparator has interrupt generation capability with wake-up from Sleep and Deepstop modes (through the PWR controller).

13.3 Comp functional description

13.3.1 COMP block diagram

Figure 25 shows the block diagram of the Comparator.

Figure 25. Comparator block diagram



13.3.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.
The comparator output can be connected to the I/Os using the alternate function channel given in “GPIOs alternate optioins” table.
The output can also be internally redirected to a variety of timer input for the following purpose:

- Emergency shut-down of PWM signals, using BKIN input
- Input capture for timing measures

It is possible to have the comparator output simultaneously redirected internally and externally.

13.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the APB0 clock.

Note: Important: the polarity selection logic and the output redirection to the port works independently from the APB0 clock. This allows the comparator to work even in Deepstop mode.

13.3.4 Comparator LOCK mechanism

The comparator can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (readonly).

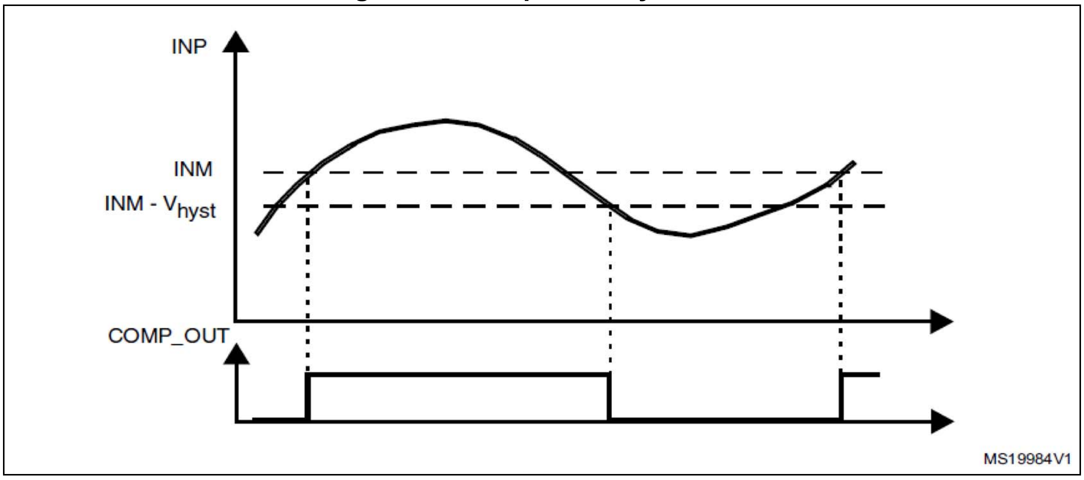
Once the programming is completed, the COMP_LOCK bit can be set to 1. This causes the whole register to become read-only, including the COMP_LOCK bit.

The write protection can only be reset by a MCU reset.

13.3.5 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

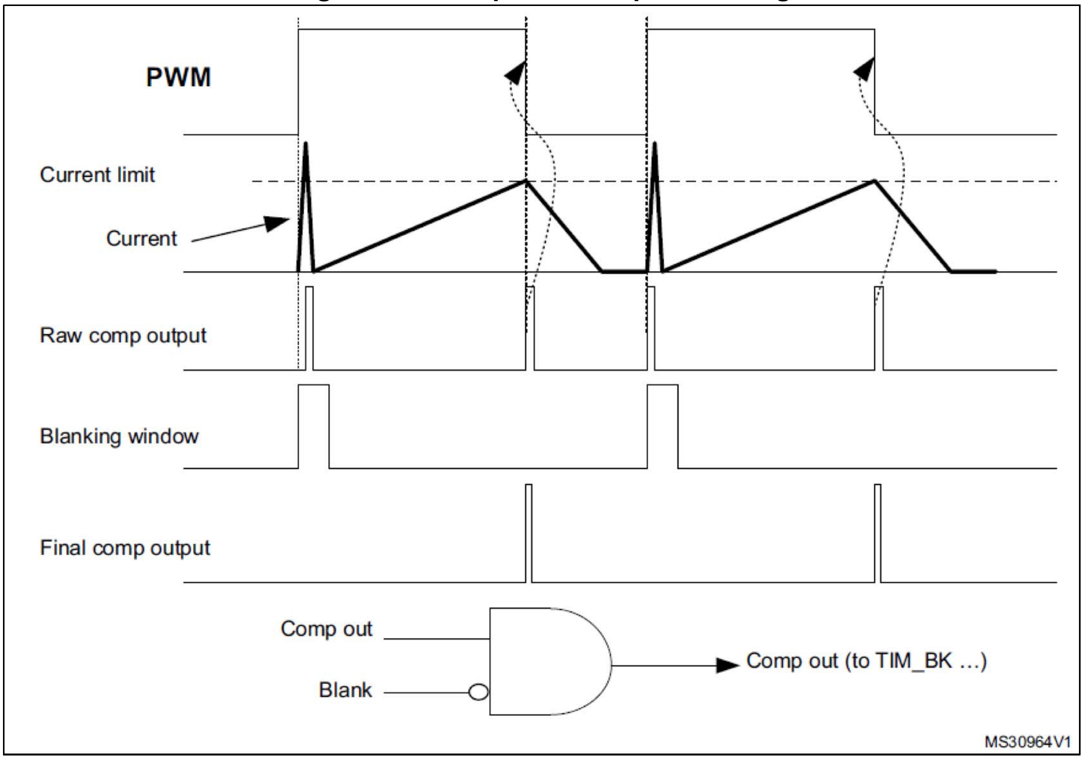
Figure 26. Comparator hysteresis



13.3.6 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes).It consists of a selection of a blanking window which is a timer output compare signal. The selection is done by software (refer to the comparator register description for possible blanking signals). Then, the complementary of the blanking signal is ANDed with the comparator output to provide the wanted comparator output. See the example provided in the figure below.

Figure 27. Comparator output blanking



13.3.7 COMP power and speed modes

COMP power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The bits COMP_PWR_MODE[1:0] in COMP_CSR registers can be programmed as follows:

00: High speed / full power

01 or 10: Medium speed / medium power

11: Low speed / ultra-low-power

13.4 Comp low-power modes

The comparator is available and functional down to Deepstop mode. The scaler is available in Deepstop mode too. In Deepstop mode:

- the comparator input channels (p and n) are connected to the selected IOs
- the analog switches of the selected IOs are kept enable
- the COMP_OUT (including polarity) can generate a system wakeup if the corrisponding line is enabled and configured in the PWRC
- the configuration of the IOs, on which the COMP_OUT is mapped, is retained so the COMP_OUT’s activity is propagated outside
- when the scaler is enabled (SCALE_EN=1) the regulated core voltage (V12) is kept at 1.2 V and is not reduced to 1.0 V

Table 36. Comparator behavior in the low power modes

MODE	Description
Deepstop	No effect on the comparator. Comparator registers status is retained. Comparator interrupt causes the device to wakeup the system and exit the Deepstop mode.
Shutdown	The COMP registers are powered down and must be reinitialized after exiting Shutdown mode

13.5 Comp interrupts and wakeup

The comparator output is internally connected to the System Controller (SYSCFG). The comparator has its own INTAI line and can generate interrupts.
The comparator output is internally connected to the Power Controller (PWRC). The comparator has its own wakeup line. This mechanism is used to exit from DeepStop mode.

To enable the COMP interrupt, it is required to follow this sequence:

1. Configure and enable the INTAI line corresponding to the COMP output event in the SYSCFG and select the rising, falling, or both edges sensitivity
2. Configure and enable the NVIC IRQ channel mapped to the corresponding INTAI line
3. Enable the COMP

To enable the COMP wakeup, it is required to follow this sequence:

1. Configure and enable the wakep line corresponding to the COMP output event in the PWRC and select polarity
2. Enable the COMP

13.6 Comp registers

13.6.1 Comparator control and status register (COMP_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	Res.	BLANKING[2:0]			HYST[1:0]	
rs	r							rw	rw		rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POLARITY	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL[1:0]		INMSEL[2:0]			PWRMODE[1:0]		Res.	EN
rw							rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **LOCK:** COMP_CSR register lock bit

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator control register, COMP1_CSR[31:0].

0: COMP1_CSR[31:0] are read/write

1: COMP1_CSR[31:0] are read-only

Bit 30 **VALUE:** Comparator output status bit

This bit is read-only. It reflects the current comparator output taking into account POLARITY bit effect.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 **SCALEN:** Voltage scaler enable bit

This bit is set and cleared by software. This bit enable the outputs of the VREFINT divider available on the minus input of the Comparator

0: scaler disable

1: scaler enable

Bit 22 **BRGEN:** Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enable the bridge of the scaler.

0: Scaler resistor bridge disable

1: Scaler resistor bridge enable

If SCALEN is set and BRGEN is reset, BG voltage reference is available but not 1/4BGAP, 1/2BGAP, 3/4 BGAP. BGAP value is sent instead of 1/4BGAP, 1/2BGAP, 3/4 BGAP.

If SCALEN and BRGEN are set, 1/4 BGAP 1/2BGAP 3/4BGAP and BGAP voltage references are available.

Bit 21 Reserved, must be kept at reset value.

Bits 20:18 **BLANKING[2:0]:** Comparator blanking source selection bits

These bits select which timer output controls the comparator output blanking.

000: No blanking

001: TIM2 OC4 selected as blanking source

010: TIM16 OC1 selected as blanking source

All other values: reserved

Bits 17:16 **HYST[1:0]:** Comparator hysteresis selection bits

These bits are set and cleared by software (only if LOCK not set). They select the Hysteresis voltage of the comparator .

00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 15 **POLARITY:** Comparator polarity selection bit

This bit is set and cleared by software (only if LOCK not set). It inverts Comparator polarity.

0: Comparator output value not inverted
1: Comparator output value inverted

Bits 14:9 Reserved, must be kept at reset value.

Bits 8:7 **INPSEL[1:0]**: Comparator input plus selection bit

This bit is set and cleared by software (only if LOCK not set).

00: PA14
01: PB1
1x: PB2

Bits 6:4 **INMSEL**: Comparator input minus selection bits

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of comparator.

000: 1/4 V_{REFINT}
001: 1/2 V_{REFINT}
010: 3/4 V_{REFINT}
011: V_{REFINT}
100: DAC OUT
101: PA13
110: PB0
111: PB3

Bits 3:2 **PWRMODE[1:0]:** Power Mode of the comparator

These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the Comparator.

00:High speed
01 or 10:Medium speed
11:Ultra low power

Bit 1 Reserved, must be kept at reset value.

Bit 0 **EN:** Comparator enable bit

This bit is set and cleared by software (only if LOCK not set). It switches on Comparator.

0: Comparator switched OFF
1: Comparator switched ON

13.7 COMP register map

The following table summarizes the comparator registers.

Table 37. COMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	COMP1_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	Res.	SCALEN	BRGEN	Res.	BLANKING			HYST		POLARITY.	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL		INMSEL			PWRMODE		Res.	EN
	Reset value	0	0							0	0		0	0	0	0	0	0							0	0	0	0	0	0	0		0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

14 Digital-to-analog converter (DAC)

14.1 Introduction

The STM32WL33xx embeds a DAC module. The DAC is not available on STM32WL30xx and STM32WL31xx devices.

The DAC module is a 6-bit, voltage output digital-to-analog converter. The DAC may be used in conjunction with the DMA controller. The DAC has three output channels:

- DACOUT_GPIO connected to PA13
- DACOUT_VCMBUF can be connected to different GPIOs (see *Table 8: GPIO alternate options AF0 - AF3* and *Table 9: GPIO alternate options AF4 - AF6*).
- DACOUT_COMPMINUS connected to COMPARATOR COMPMINUS input

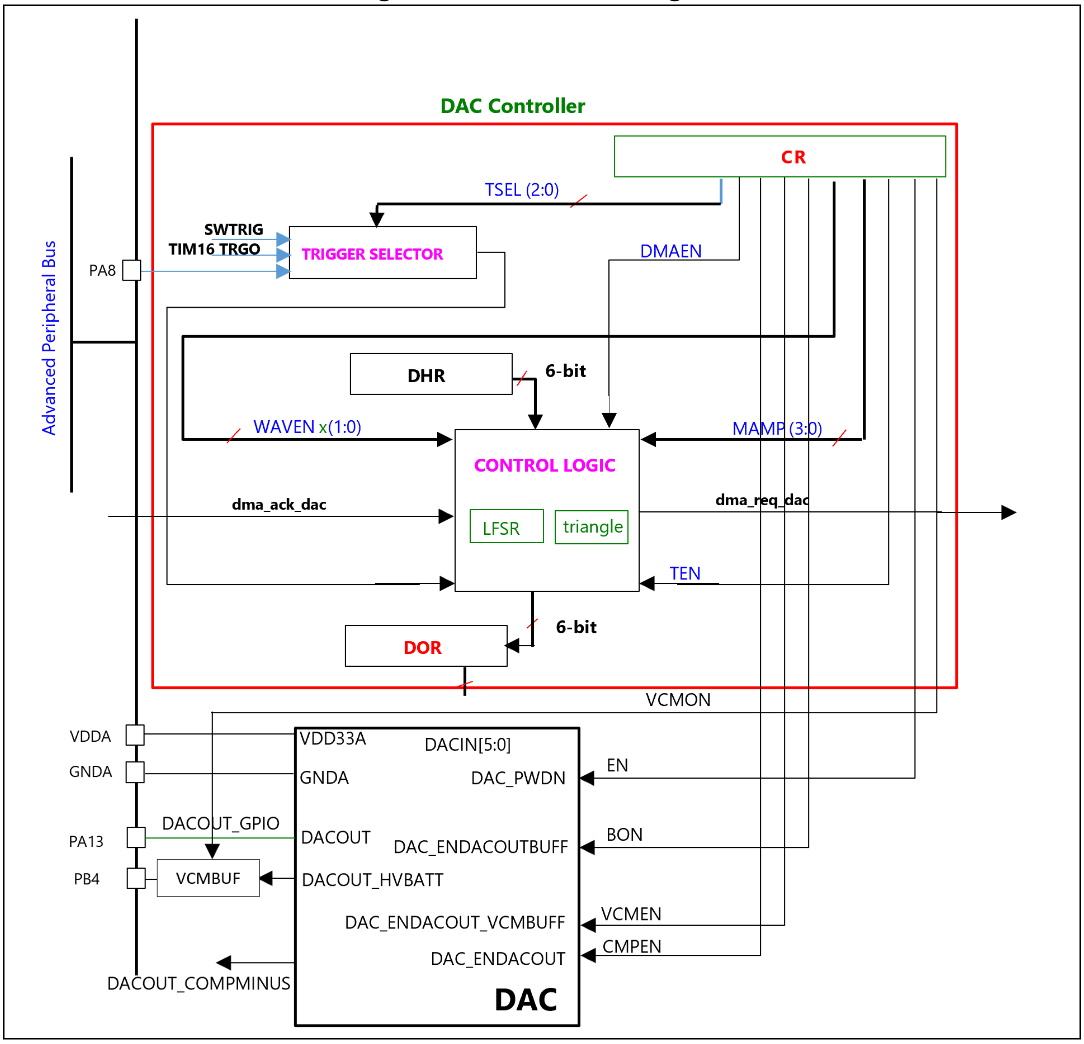
14.2 DAC main features

- Synchronized update capability
- Noise-wave generation
- Triangular-wave generation
- DMA capability
- DMA underrun error detection and interrupt generation
- External triggers for conversion

14.3 DAC block diagram

Figure 28 shows the block diagram of the DAC.

Figure 28. DAC block diagram



14.4 DAC functional description

14.4.1 DAC channel enable

DAC channel can be powered on by setting its corresponding EN bit in the DAC_CR register. The DAC channel is then enabled after a startup time t_{WAKEUP}.

Note: The EN bit enables the analog DAC Channel macrocell only. The DAC Channel digital interface is enabled even if the EN bit is reset.

14.4.2 DAC outputs enable

The DAC integrates one output buffer that can be used to reduce the output impedance, and to drive external loads directly without having to add an external operational amplifier. DAC channel output buffer can be enabled and disabled using the corresponding BON bit in the DAC_CR register.

DAC has other two output channels connected internally respectively to VCMBUFF (which can be switched on by VCMON bit) and to COMPINMINUS that can be enabled using corresponding bits VCMEN and CMPEN.

14.4.3 DAC conversion

The DAC_DOR cannot be written directly and any data transfer to the DAC channel must be performed by loading the DAC_DHR register.

Data stored in the DAC_DHR register are automatically transferred to the DAC_DOR register after one APB0 clock cycle, if no hardware trigger is selected (TEN bit in DAC_CR register is reset). However, when a hardware trigger is selected (TEN bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three APB0 clock cycles later.

When DAC_DOR is loaded with the DAC_DHR contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

14.4.4 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

DACOUT = VDD x (DOR/64)

14.4.5 DAC trigger selection

If the TEN control bit is set, conversion can then be triggered by an external event (TIM16 counter, PA8 from SYSCFG). The TSEL[2:0] control bits determine which out of 3 possible events trigger conversion (see register *DAC control register (DAC_CR)*).

Each time a DAC interface detects a rising edge TIM16 TRGO output, or on the PA8 pin event from SYSCFG, the last data stored into the DAC_DHR register are transferred into the DAC_DOR register. The DAC_DOR register is updated three APB0 cycles after the trigger occurs, while if software trigger occurs it’s updated one APB0 cycle later.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DOR register has been loaded with the DAC_DHR register contents.

Note: TSEL[2:0] bit cannot be changed when the EN bit is set.

When software trigger is selected, the transfer from the DAC_DHR register to the DAC_DOR register takes only one APB0 clock cycle.

14.4.6 DMA request

DAC channel has a DMA capability. DMA channel is used to service DAC channel DMA requests.

A DAC DMA request is generated when an external trigger (but not a software trigger) occurs while the DMAEN bit is set. The value of the DAC_DHR register is then transferred into the DAC_DOR register.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channel underrun flag DMAUDR in the DAC_SR register is set, reporting the error condition. The DAC channel continues to convert old data.

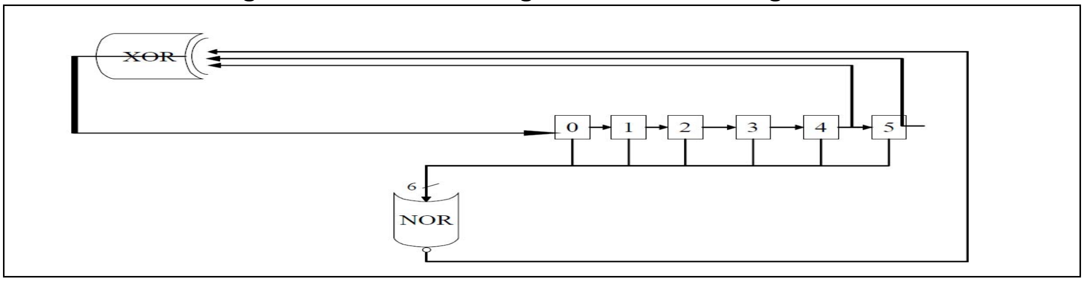
The software should clear the DMAUDR flag by writing “1”, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channel to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For the DAC channel, an interrupt is also generated if its corresponding DMAUDRIE bit in the DAC_CR register is enabled.

14.4.7 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVE[1:0] to “01”. The preloaded value in LFSR is 0x2A. This register is updated, following a specific calculation algorithm, three APB0 clock cycles after each trigger event, unless software trigger is selected, in this case counter is incremented one APB0 clock cycle after each software trigger. A sequence of up to 2^n-1 numbers (where n = number of DAC bit) can be generated before the sequence repeats. The noise produced by this generator has a flat spectral distribution and can be considered white noise. However, instead of having a Gaussian output characteristics, it is uniformly distributed.

Figure 29. DAC LFSR register calculation algorithm



The LFSR value, that may be masked partially or totally by means of the MAMP[3:0] bits in the DAC_CR register, is added up to the DAC_DHR contents without overflow and this value is then stored into the DAC_DOR register.

If LFSR is 0x0000, a '1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVE[1:0] bits.

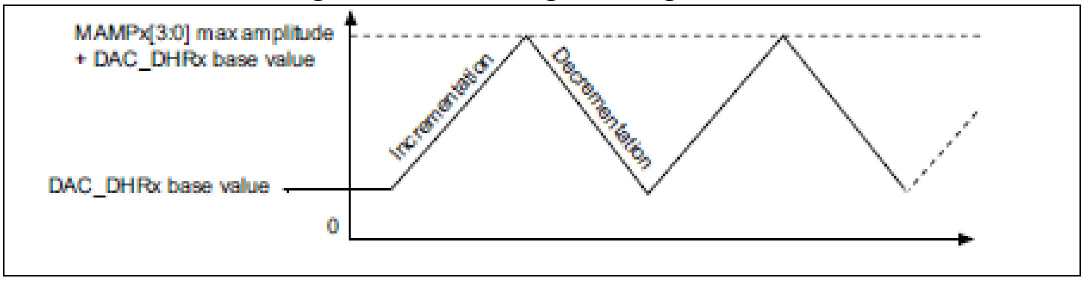
Note: The DAC trigger must be enabled for noise generation by setting the TEN bit in the DAC_CR register.

14.4.8 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVE[1:0] to "1X". The amplitude is configured through the MAMP[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three APB0 clock cycles after each trigger event, unless software trigger is selected, in this case counter is incremented one APB0 clock cycle after each software trigger. The value of this counter is then added to the DAC_DHR register without overflow and the sum is stored into the DAC_DOR register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMP[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVE[1:0] bits.

Figure 30. DAC triangle wave generation



Note: The DAC trigger must be enabled for triangle-wave generation by setting the TEN bit in the DAC_CR register.

The MAMP[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

14.5 DAC registers

14.5.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

Note: VCMBUFF can be used only in the LCSC (STM32WL33xx only) application (see Section 15: LC sensor controller (LCSC)) or by programming LCSC_ANATST_CFG register (bits[1:0]=0x3). VCMBUFF output is Vdd/2 whatever the value of DACOUT.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCMON
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VCMEN	CMPEN	DMAUDRIE	DMAEN	MAMP[3:0]				WAVE[1:0]		TSEL[2:0]			TEN	BON	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept cleared.

Bit 16 **VCMON**: VCMBUFF power-up. This bit is set and cleared by software.
0: VCM BUFFER OFF
1: VCM BUFFER ON

Bit 15 **VCMEN**: DAC channel output to VCM BUFFER enable. This bit is set and cleared by software.
0: DAC channel output to VCM BUFFER disabled
1: DAC channel output to VCM BUFFER enabled

Bit 14 **CMPEN**: DAC channel output to COMP INMINUS enable. This bit is set and cleared by software.
0: DAC channel output to COMP INMINUS disabled
1: DAC channel output to COMP INMINUS enabled

Bit 13 **DMAUDRIE**: DAC channel DMA Underrun Interrupt enable This bit is set and cleared by software.
0: DAC channel DMA Underrun Interrupt disabled
1: DAC channel DMA Underrun Interrupt enabled

Bit 12 **DMAEN**: DAC channel DMA enable This bit is set and cleared by software.
0: DAC channel DMA mode disabled
1: DAC channel DMA mode enabled

Bits 11:8 **MAMP[3:0]**: DAC channel mask/amplitude selector These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
≥ 0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

Bits 7:6 **WAVE[1:0]**: DAC channel noise/triangle wave generation enable These bits are set and cleared by software.

00: wave generation disabled

01: Noise wave generation enabled

1x: Triangle wave generation enabled

Note: Only used if bit TEN = 1 (DAC channel trigger enabled).

Bits 5:3 **TSEL[2:0]**: DAC channel trigger selection These bits select the external event used to trigger DAC channel.

000: Timer 16 TRGO event

001: PA8 pin event from SYSCFG

010 to 011: Reserved

111: Software trigger

Only used if bit TEN = 1 (DAC channel trigger enabled).

Bit 2 **TEN**: DAC channel trigger enable This bit is set and cleared by software to enable/disable DAC channel trigger.

0: DAC channel trigger disabled and data written into the DAC_DHR register are transferred one APB0 clock cycle later to the DAC_DOR register

1: DAC channel trigger enabled and data from the DAC_DHR register are transferred three APB0 clock cycles later to the DAC_DOR register

Note: When software trigger is selected, the transfer from the DAC_DHR register to the DAC_DOR register takes only one APB0 clock cycle.

Bit 1 **BON**: DAC channel output buffer enable. This bit is set and cleared by software to enable/disable DAC channel output buffer.

0: DAC channel output buffer disabled

1: DAC channel output buffer enabled

Bit 0 **EN**: DAC channel enable This bit is set and cleared by software to enable/disable DAC channel.

0: DAC channel disabled

1: DAC channel enabled

14.5.2 DAC software trigger register (DAC_SWTRIGR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:1 Reserved, must be kept cleared.

Bit 0 **SWTRIG**: DAC channel software trigger This bit is set by software to enable/disable the software trigger.

0: Software trigger disabled

1: Software trigger enabled

Note: This bit is cleared by hardware (one APB0 clock cycle later) once the DAC_DHR register value has been loaded into the DAC_DOR register.

14.5.3 DAC channel data holding register (DAC_DHR)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:6 Reserved, must be kept cleared.

Bits 5:0 **DACDHR[5:0]: DAC channel 6-bit data These bits are written by software which specifies 6-bit data for DAC channel.**

14.5.4 DAC channel data output register (DAC_DOR)

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 31:6 Reserved, must be kept cleared.

Bits 5:0 **DACDOR[5:0]**: DAC channel data output These bits are read-only, they contain data output for DAC channel.

14.5.5 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	DMAUDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r_wc1													

Bits 31:14 Reserved, must be kept cleared.

Bit 13 **DMAUDR**: DAC channel DMA underrun flag This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel

1: DMA underrun error condition occurred for DAC channel (the currently selected trigger is driving DAC channel conversion at a frequency higher than the DMA service capability rate)

Bits 12:0 Reserved, must be kept at cleared.

14.6 DAC register map

The following table summarizes the comparator registers.

Table 38. DAC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCMON	VCMEN	CMPEN	DMAUDRIE	DMAEN	MAMP[3:0]				WAVE[1:0]		TSEL[2:0]			TEN	BON	EN
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DAC_SWTRIGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG
	Reset value																																0
0x10	DAC_DHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACDHR[5:0]					
	Reset value																											0	0	0	0	0	0
0x2C	DAC_DOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACDOR[5:0]					
	Reset value																											0	0	0	0	0	0
0x34	DAC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAUDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																			0													

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

15 LC sensor controller (LCSC)

15.1 Introduction

The STM32WL33xx embeds an LC Sensor controller (LCSC). The LCSC is not available on STM32WL30xx and STM32WL31xx devices.

The LC sensor controller (LCSC) controls DAC, COMP, VCMBUFF power-on/power-off and dedicated GPIOs (PB1, PA14, PB2) for the measurement of LC networks damping times.

In this kind of application, the flow of a fluid in a pipe forces the rotation of a wheel, whose number of revolutions permits quantification of the amount of fluid consumed. Three external LC networks, positioned on top of the wheel, allow counting of the number of revolutions. These networks are to be connected to identified GPIOs. Two of them (PB1 and PA14) allow the position of the rotating wheel to be determined, whereas the other (PB2) is necessary to detect tampering.

15.2 LCSC main features

- Management of the sequence of measurement of the 3 available LC networks, knowing that the third LC, for tamper, is not intended to be measured at each sequence. PB1, PA14 and PB2 GPIO connected to LC nectwork are driven by LCSC (additional functions), as soon as the LCSC is enabled, independently of *General-purpose I/O (GPIO)* registers configuration.
- Management of the power-on/power off of COMP, DAC, VCMBUFF, as soon as the LCSC is enabled, independently on how are configured EN bit in *Comparator control and status register (COMP_CSR)* and EN and VCMON bits in *DAC control register (DAC_CR)*.
- Count the number of *Comparator (COMP)* output edges for each LC to determine the wheel position or if there is any tamper.
- Count the number of wheel revolutions (a full revolution means that the wheel has started rotating from a defined initial position up to returning to this position).
- Wakeup from low power modes (SLEEP or Deepstop) (see *Section 5.8: Programmer's model*) and notify the system through an interrupt (see *Section 2.3.2: Interrupts*) when the total number of wheel revolutions reaches a given threshold fixed by the application user.

15.3 LCSC functional description

15.3.1 LCSC block diagram

Figure 31 shows the block diagram of the LCSC, while *Figure 32* shows connections of LCSC to DAC, COMP, VCMBUFF and PB1, PA14, PB2 and PB4 GPIO.

Figure 31. LCSC block diagram

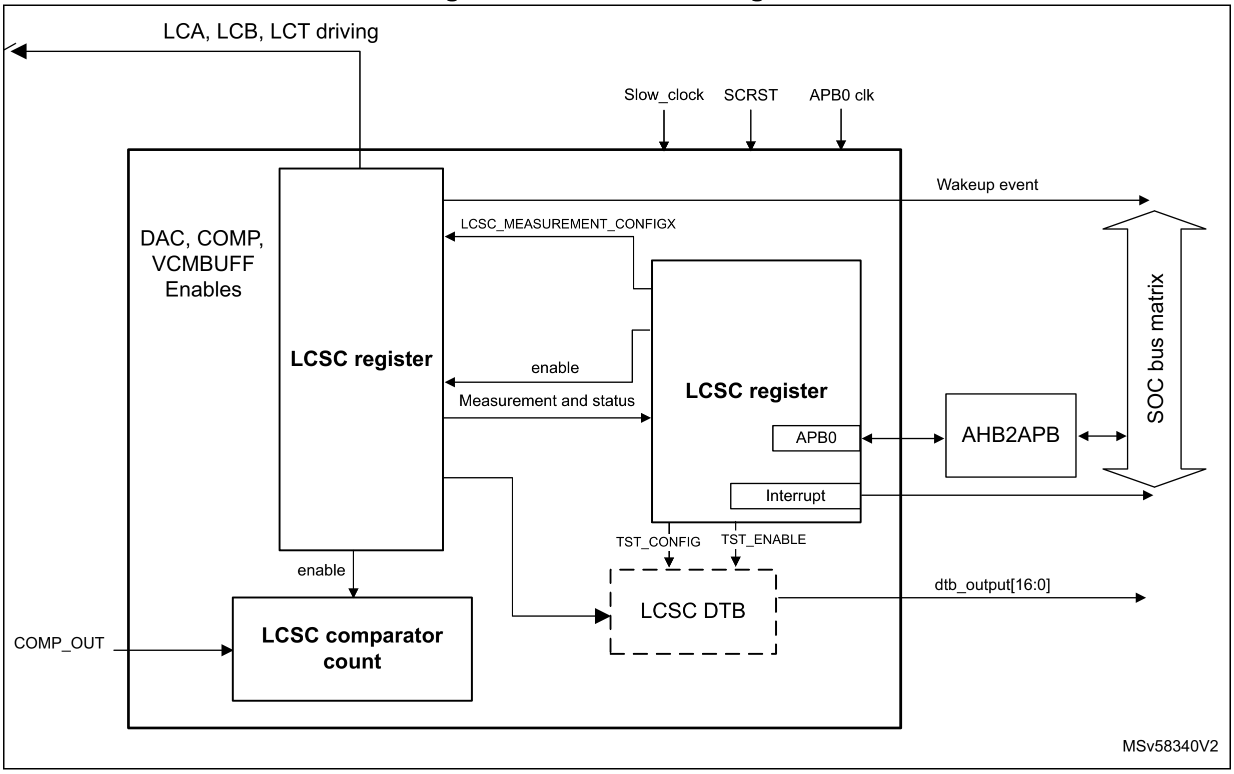
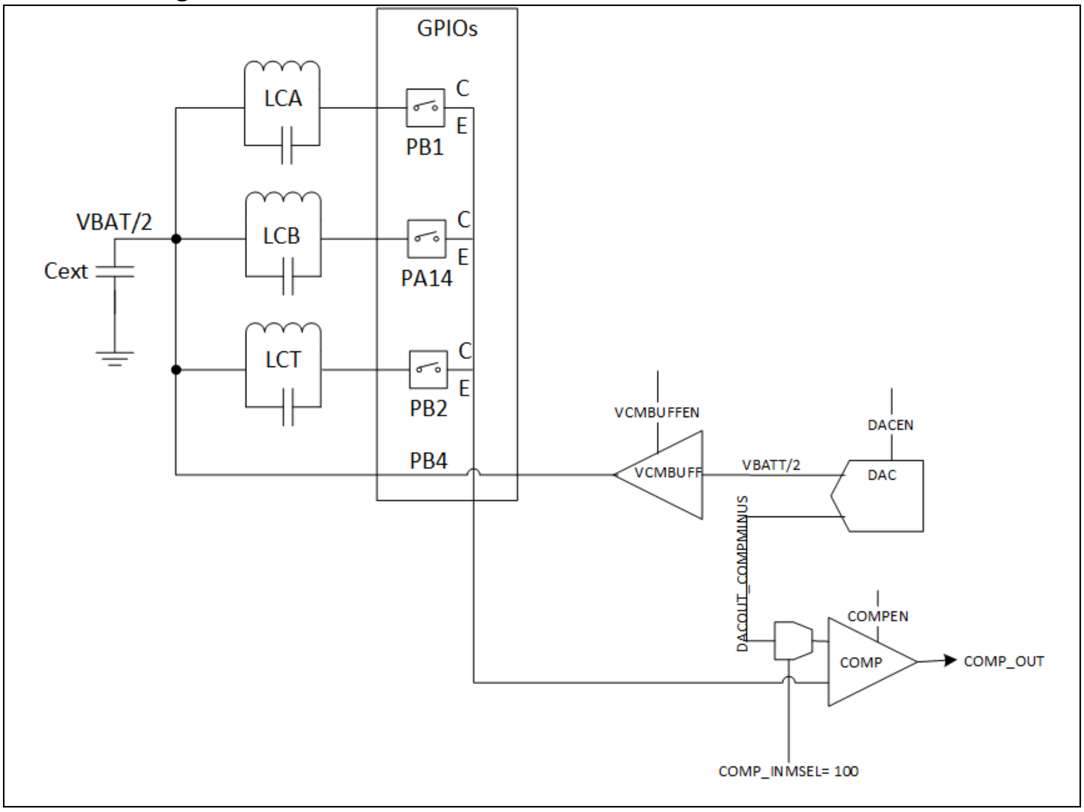


Figure 32. LCSC connection to DAC, COMP, VCMBUFF, GPIOs



15.3.2 Overview

The purpose of the LCSC is to control the COMP, DAC, VCMBUFF and PB1, PA14, PB2 in order to implement the LCSC feature.

Before detailing the behavior of the LCSC, it’s useful to describe the feature and define the different keywords in order to guide the reader through the specifications.

First of all, the LCSC feature is dedicated to the fluid metering (often simplified in water metering). Indeed, this feature is deployed in order to measure the fluid consumption, for instance the water consumption in a house.

The fluid consumption measurement is based on two mechanisms:

- A rotating wheel, which is driven by the fluid movement inside a pipe
- A combination of LC networks allowing to detect the position of the wheel, and finally to allow to count the number of full rotations of the wheel, which is directly linked to the water consumption.

LC network is the name given to the combination of an Inductance (L) and a Capacitor (C), that are integrated passive electronic components. These two components are placed in parallel to form a LC network and in this case the LC network may be considered as an oscillator (see *Figure 32*).

This notion of oscillator is very important, it is this behavior that is used by the LCSC feature to measure the water consumption. Indeed, the idea is to measure the LC oscillation damping time and to use this damping time to determine what the wheel position is.

There are two important things to note to understand the concept:

- The rotating wheel is half covered with a metallic surface (see *Figure 33*)
- The LC oscillation damping time is **faster** if the LC network is placed near a metallic surface.

Back to the LC networks to understand how to use the above information. The idea is to use two LC networks to detect the wheel position, and a third LC network that are described later. In the whole documentation we consider:

- The first LC network as LCA
- The second LC network as LCB
- The third LC network as LCT

We first focus on the LCA and LCB networks, they are positioned external to the SoC and driven by LCSC.

We can divide the rotating wheel half-covered with a metallic surface in 4 quarters.

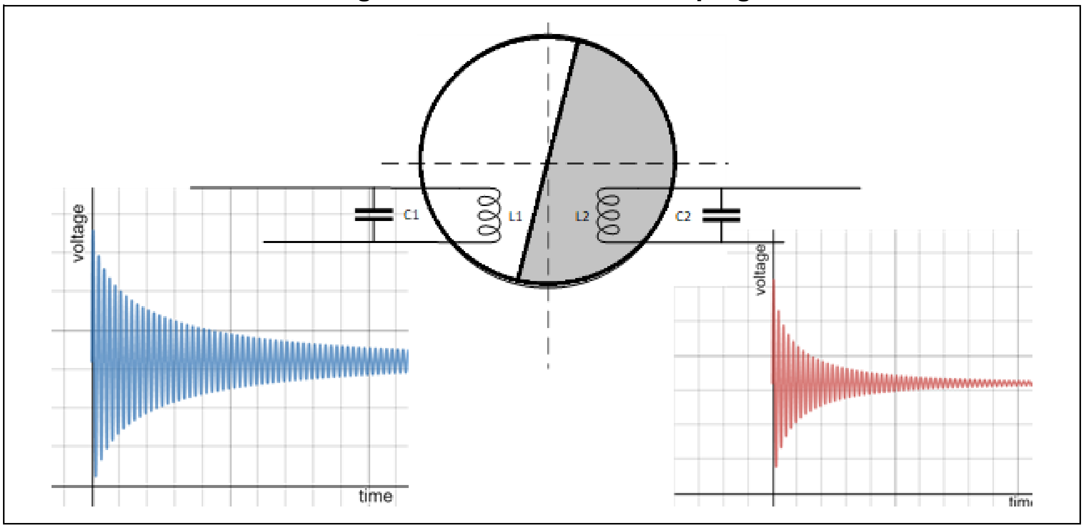
- LCA is located near a first quarter
- LCB is located near a second quarter.

Because the wheel is rotating, the LCs are not always see the same surface, in fact we can even describe 4 different positions:

- LCA and LCB facing a metallic surface
- LCA and LCB facing a none-metallic surface
- LCA facing a metallic surface but LCB not
- LCA not facing a metallic surface but LCB yes.

The *Figure 33* shows the wheel, divided in quarters, with the two LC networks (LCA on the lower left and LCB on the lower right) with the LC oscillation damping for each:

Figure 33. LC oscillation damping

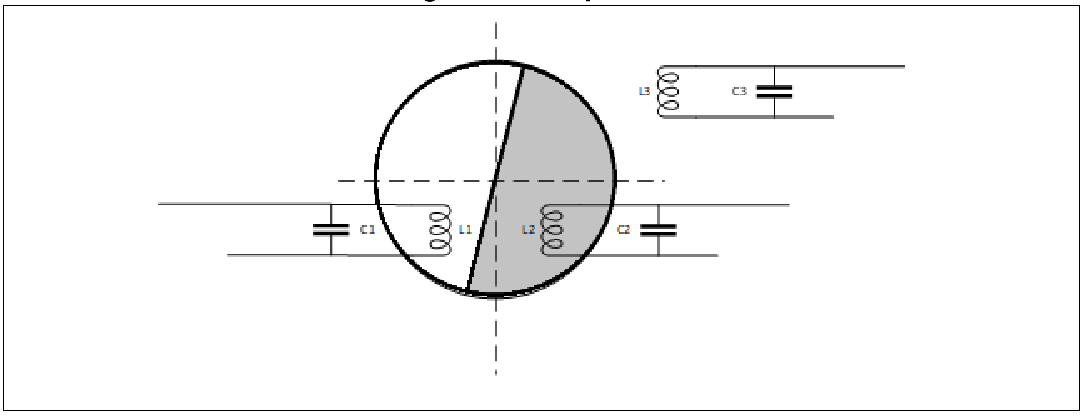


In this wheel position example we can see that LCA is not facing a metallic surface, then the blue oscillation is damping “normally”, whereas the LCB is facing a metallic surface, then the red oscillation is damping much faster.

Finally, the LCSC uses these two LC networks, to repeat LC oscillation damping time measurement, to determine during the whole LCSC process, what is the position of the wheel, which should be rotating if a fluid is moving inside the pipe. Depending on the wheel position, LCSC counts the number of revolutions (full rotations) and finally alert the user if a defined number of revolutions has been counted.

LCT means LC Tamper (see *Figure 34*), because this LC network is totally dedicated to the Tamper Detection, allowing to prevent any kind of fraudulence. Indeed, we could imagine that someone tries to “hack” the system, by adding a metallic surface above the water-meter for example. This would change the measurement results by forcing the two LC to see a metallic surface, and thus to consider that the wheel is no more rotating, because at each measure we would have a faster oscillation damping time, synonym of facing the two metallic-covered quarters.The third LC is not placed near a quarter such as LCA and LCB, but next to the wheel. This means that this LCT should always be measuring an oscillation damping time corresponding to a LC which is not facing a metallic surface. But if a tamper is detected, then this LCT sees an oscillation damping time shorter than expected, synonym of tamper detection, which would lead to alert the user of a fraudulent usage.

Figure 34. Tamper LCT



15.3.3 LCSC reset and clocks

The LCSC clock provided by the clock controller is synchronous with the APB0 clock and can be enabled through LCSCEN bit inside *APB0 macrocell clock enable register (RCC_APB0ENR)*.

Since LCSC must work also in Deepstop mode, another clock is provided to LCSC which is the slow clock selectable through CLKSLOWSEL bits inside *Clock configuration register (RCC_CFGR)*.

The Reset of the LCSC is controlled by LCSCRST bit inside *APB0 macrocell reset register (RCC_APB0RSTR)* and it’s status can be read through LCSCRSTF inside *Clock Interrupt flag register (RCC_CIFR)*, this flag can also generate a PWRC interrupt in case bit LCSCRSTIE is set inside *This register controls the enable and variable rate multiplier for SMPS clock.*.

15.3.4 LCSC programming

To make the LCSC work, some steps need to be configured by software before the block is enabled via the LCSC_EN bit in the *LCSC enable register (LCSC_ENR)*:

1. Configure PB4 as output pin using GPIO registers through GPIOB port mode register (GPIOB_MODER) and output data zero on PB4 pin through GPIOB port output data

register (GPIOB_ODR), this allows to discharge residual voltage on external capacitor, Cext.

2. Configure PB4 as an analog pin using GPIO registers through the *GPIOB port mode register (GPIOB_MODER)*.
3. Enable VCMBUFF output (VDD33/2, also termed VBATT/2) on PB4 pin setting VCMEN bit in the *DAC control register (DAC_CR)*.
4. Connect DACOUT_COMPINMINUS to COMP_INMINUS by setting the CMPEN bit in the *DAC control register (DAC_CR)*.
5. Configure the DAC DHR[6:0] register to chose DAC analog output level of DACOUT_COMPINMINUS using the *DAC channel data holding register (DAC_DHR)*.
6. Configure bitfield DACDHR[5:0] in the *DAC channel data holding register (DAC_DHR)* to chose the DAC analog output level of DACOUT_COMPINMINUS.
7. Configure PWRMODE, HYS to and INMSEL bits (see *Section 13.6.1: Comparator control and status register (COMP_CSR)* to configure COMP and to connect DACOUT_COMPMINUS to the COMP (see *Figure 32: LCSC connection to DAC, COMP, VCMBUFF, GPIOs*).

Caution: If APC bit is set, the user has to disable the PUB1, PUB2, PUA14, PUB4 and PDB1, PDB2, PDA14, PDB4 bits by software inside the , *I/O Port A pull-down control register (PWRC_PDCRA)*, *I/O Port B pull-up control register (PWRC_PUCRB)* and *I/O Port B pull-down control register (PWRC_PDCRB)* for the feature to work properly. Otherwise, if APC is reset, the pull-up, pull-down of PB1, PB2, PA14 and PB4 are automatically disabled.

8. Program LCSC configuration registers (see *LCSC register map*).
9. Enable LCSC through the LCSC_EN bit. These 9 first steps are managed by software.

Caution: Once the LCSC is enabled PB1, PA14, PB2 are configured as additional functions and they are driven by LCSC independently of GPIO configuration registers settings.

10. LCSC enables the DAC, COMP, VCMBUFF allowing to switch-on them. (In this case LCSC takes priority on DACEN, VCMON bits of the DAC and EN bit of the COMP).
11. LCSC waits for a Startup time ($\mathbf{T_{StartInit}}$), (see *Figure 35*) this time allows the DAC, COMP and VCMBUFF to be switched-on correctly (3 slow clock cycles) and to ensure that VCMBUFF voltage has reached VDD3/2 ($\mathbf{T_{StartVCM}}$).

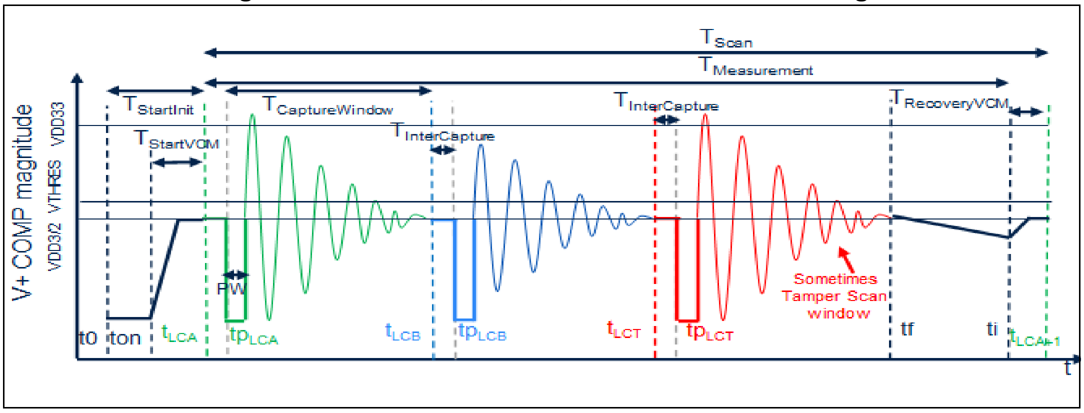
Note: The startup time, such as the other durations to wait in this description, is expressed as a number of slow clock cycles, not like an absolute time.

12. LCSC enables the LCSC comparator count sub-block (see *Figure 31*).
13. LCSC selects the first LC to be measured, LCA.
14. LCSC waits for two slow clock cycles.
15. LCSC generates a pulse low on LCA whose width is programmable (see the *LCSC pulse configuration register (LCSC_PULSE_CR)*) and $\mathbf{T_{capture}}$ starts.
16. LCSC receives the outputs of the COMP and increments the LCSC comparator count value for each value received.
17. LCSC compares comparator count sub-block with LCAB_DAMP_THRES previously configured by the software in the *LCSC control register 1 (LCSC_CR1)*, in order to decide if LCA is near a metallic surface or not.
18. Once the $\mathbf{T_{capture}}$ is elapsed (see *Figure 35*), LCSC reset comparator sub-block counter and selects the second LC to be measured, LCB, whereas the LCA is unselected.
19. LCSC waits for $\mathbf{T_{interCapture}}$ delay (see *Figure 35*).
20. LCSC generates a pulse low on LCB (see see *LCSC pulse configuration register (LCSC_PULSE_CR)*) and $\mathbf{T_{capture}}$ starts.
21. LCSC receives the outputs of the COMP and increments the LCSC comparator count value for each value received.
22. LCSC compares comparator count sub-block with LCAB_DAMP_THRES previously configured by the software in the *LCSC control register 1 (LCSC_CR1)*, in order to decide if LCB is near a metallic surface or not.
23. Once the $\mathbf{T_{capture}}$ is elapsed (see *Figure 35*), the LCB is released, the LCSC comparator count uses the comparisons of 17 and 22 steps to conclude on the position of the wheel. At the same time there are multiple possibilities:
    a) If a tamper detection is required by the software through the TAMP_PSC[7:0] bits, see step 24.
    b) If there is no need to detect any tamper, but the measurement period $\mathbf{T_{measurement}}$ (where $\mathbf{T_{scan}} = \mathbf{T_{measurement}} + \mathbf{T_{recoveryVCM}}$ + 3 slow clock cycles) is already reached, select LCA, wait for a $\mathbf{T_{interCapture}}$ delay and redo the process from step 13.
    c) If there is no need to detect any tamper, and the measurement period $\mathbf{T_{measurement}}$ is not reached, jump to step 30.
24. LCSC selects LCT
25. Wait for a $\mathbf{T_{interCapture}}$ delay

26. LCSC generates a pulse low on LCT whose width is programmable (see the *LCSC pulse configuration register (LCSC_PULSE_CR)*).
27. LCSC receives the outputs of the COMP and increments the LCSC comparator count value for each value received.
28. LCSC compares comparator count sub-block with LCT_DAMP_THRES previously configured by the software in the *LCSC control register 1 (LCSC_CR1)*, in order to decide if LCT is near a metallic surface or not.
29. Once the $\mathbf{T_{capture}}$ is elapsed, the LCT is released and the LCSC comparator count shall use the comparison of step 28 to conclude on the detection of a tamper or not. There are two possibilities:
    a) The measurement period $\mathbf{T_{measurement}}$ is already reached, wait for a $\mathbf{T_{interCapture}}$ delay and LCSC redo the process from step 13.
    b) The measurement period $\mathbf{T_{measurement}}$ is not reached yet, continue to step 30.
30. Ensure that there is not any LC selected (all the LC selection signals at 0)
31. Disable DAC, VCMBUFF, COMP
32. Wait for reaching the end of the measurement period $\mathbf{T_{measurement}}$, plus $\mathbf{T_{recoveryVCM}}$ + 3 slow clock cycles, and redo the process from step 10.

Note: In all cases, if LCSC_EN is released, the process executes step 30 to properly shutdown the DAC, COMP and VCMBUFF.

Figure 35. LCSC PB1, PA14, PB2,PB4 control timing



1. $T_{intercapure}$ is equal to LCSC_CR0.TICAP + 2 slow clock cycles
2. $T_{recoveryVCM}$ is equal to LCSC_CR1.TREC_VCM + 3 slow clock cycles

15.4 LCSC comparator count statistics and monitoring

The number of oscillations of a LC network is dependent of the environmental conditions and may change over time (Temperature, Voltage, Aging, Process, …), then the damping count threshold should be tuned during the runtime of the application and the LCSC. In order to help the software to adapt the value of the LCAB_DAMP_THRES during runtime, the LCSC allows the software to achieve active monitoring of the comparator count min and max values (MIN_LCAB_CNT and MAX_LCAB_CNT), or it can monitor by itself passively comparing the min and max count values to the given bounds (MIN_LCAB_CNT_BOUND, MAX_LCAB_CNT_BOUND). The maximum number of oscillations counted during the LCA and LCB measurements can be reported in the *LCSC statistics register (LCSC_STAT)* register through MAX_LCAB_CNT. The minimum number of oscillation counted during the

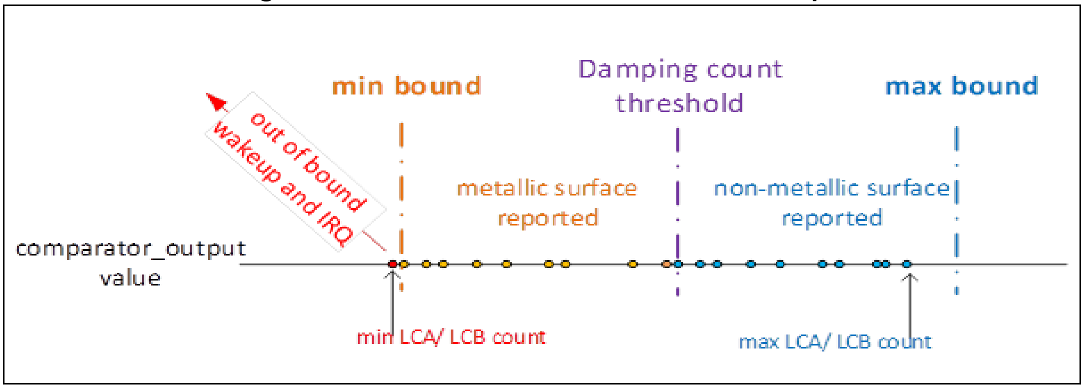
LCA and LCB measurements can be reported in the *LCSC statistics register (LCSC_STAT)* register through MIN_LCAB_CNT. In order to save power, the both MAX_LCAB_CNT and MIN_LCAB_CNT are reported only if bit CNT_OFB_WKP_IE has been previously set by software. The MAX_LCAB_CNT is reset when the software writes 1 in the CNT_OFB_F bit (even when the value is already 1). The MIN_LCAB_CNT is set to 0xFF when the software writes 1 in the CNT_OFB_F bit (even when the value is already 1).

Note: the CNT_OFB_F bit can be written at anytime to 1 to reset the min and max values, whatever its current value.

In addition, the LCSC is monitoring the minimum and maximum values (MAX_LCAB_CNT and MIN_LCAB_CNT) and comparing these values to values given by the software in the MIN_LCAB_CNT_BOUND and MAX_LCAB_CNT_BOUND inside *LCSC statistics register (LCSC_STAT)* register.

When the software has set the CNT_OFB_WKP_IE, the LCSC set the CNT_OFB_F when either the MAX_LCAB_CNT value is strictly greater than the MAX_LCAB_CNT_BOUND value stored in the LCSC_STAT register or the MIN_LCAB_CNT value is strictly lower than the MIN_LCAB_CNT_BOUND value stored in the *LCSC statistics register (LCSC_STAT)* register (see *Figure 36*).

Figure 36. LCAB counters out-of-bound example



Note: The default value of MAX_LCAB_CNT_BOUND at 0xFF and MIN_LCAB_CNT_BOUND at 0x00 makes the CNT_OFB_F flag to never being set and the wakeup and IRQ events never occurring with default configuration.

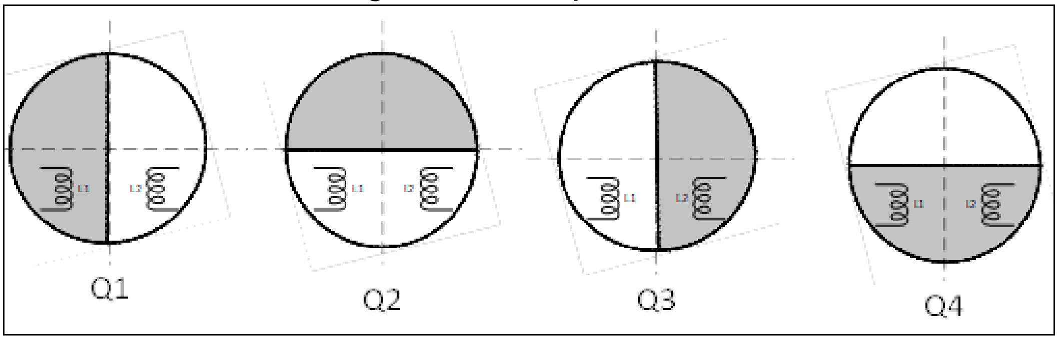
Note: The flag is never set by the LCSC if the CNT_OFB_WKP_IE *is not set as the* MIN_LCAB_CNT *and* MAX_LCAB_CNT *are not computed.*

Note: When an out of bound count is reported to the system the software may decide to update the damping count threshold, the minimum bound or the maximum bound, this is why these register bit fields are considered dynamic.

15.5 LCSC status

The wheel status is reported in the *LCSC status register (LCSC_SR)* register determining in which quarter the wheel is placed (see *Figure 37*).

Figure 37. Wheel quarter status



15.6 LCSC interrupts

Two LCSC interrupts are available:
The first one (LCSC interrupt see *Table 7: Interrupt vectors*) can be generated when:

- the number of wheel clock wise revolution (see the *LCSC wheel status register (LCSC_WHEEL_SR)*) reaches the programmed threshold (see the *LCSC wheel configuration register (LCSC_CONFR)*).
- the number of wheel anti clock wise revolution (see the *LCSC wheel status register (LCSC_WHEEL_SR)*) reaches the programmed threshold (see the *LCSC wheel configuration register (LCSC_CONFR)*).
- a tamper has been detected (see the *LCSC interrupt status register (LCSC_ISR)*).
- out of bound has been reached (see the *LCSC interrupt status register (LCSC_ISR)*).

The second one (LCSC LC_ACTIVITY interrupt) (see *Table 7: Interrupt vectors*) can be generated when:

- an LC oscillation is ongoing (see the *Internal asynchronous interrupt detection status and clear register (INTAI_ISCR)*)

15.7 LCSC low-power modes wakeup

The LCSC is available and functional down to Deepstop mode.

In Deepstop mode, a wakeup event is generated when:

- the number of wheel clock wise revolution (see the *LCSC wheel status register (LCSC_WHEEL_SR)*) reaches the programmed threshold (see *LCSC wheel configuration register (LCSC_CONFR)*).
- the number of wheel anti clock wise revolution (see*LCSC wheel status register (LCSC_WHEEL_SR)*) reaches the programmed threshold (see *LCSC wheel configuration register (LCSC_CONFR)*).
- a tamper has been detected (see *LCSC interrupt status register (LCSC_ISR)*).
- out of bound has been reached (see *LCSC interrupt status register (LCSC_ISR)*).

15.8 LCSC registers

15.8.1 LCSC control register 0 (LCSC_CR0)

Address offset: 0x00

Reset value: 0x000B 005C

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

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **TICAP[2:0]:** Inter Capture Time

The inter-capture duration (also called $\mathbf{T_{interCapture}}$) is the duration between the end of a first TCAP and the following TCAP and is equal to TICAP + 2 slow clock cycles.

This duration is given in number of slow clock cycles

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

Bits 21:16 **TCAP[5:0]:** Capture Time

The capture duration (also called $\mathbf{T_{capture}}$) is the duration of one LC measurement, starting from the excitation pulse on the LC line and ending once the TCAP number of slow clock cycles has been reached.

This duration is given in number of slow clock cycles

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

Bits 13:0 **TMEAS[13:0]**: Measurement Time

The measurement duration (also called $\mathbf{T_{measurement}}$) is the duration of an LC measurement sequence (including LCA, LCB and LCT if necessary, and also the time left before the next power-up of analog blocks (DAC, COMP, VCMBUFF)).

This period is given in number of slow clock cycles.

15.8.2 LCSC control register 1 (LCSC_CR1)

Address offset: 0x04

Reset value: 0x3C01 0C80

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:20 **TSTART_VCM[10:0]:** VCMBUFF Starting Time

Corresponds to the duration $\mathbf{T_{StartVCM}}$ (in number of slow clock cycles) between the enable of the VCMBUFF and the establishment of the Voltage to the value VDD33/2. This time is only to be considered for the first LC measurement after an enabling of the LCSC feature thanks to the LCSC_EN bit.

Note: TSART_VCM must not be set to values lower of 0x02

Bit 19 Reserved, must be kept at reset value.

Bits 18:10 **TREC_VCM[8:0]:** VCMBUFF Recovery Time

Corresponds to the duration $\mathbf{T_{recoveryVCM}}$ (in number of slow clock cycles) between the enable of the VCMBUFF and the establishment of the Voltage to the value VDD33/2. This time is to be considered between each sequence of LC measurement, called TSCAN in the registers. (sequence of LC measurement = LCA, LCB and sometimes LCT).

Note: TREC_VCM must not be set to values lower of 0x02

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:0 **LCAB_DAMP_THRES[7:0]:** Damping threshold for LCA and LCB

The count threshold used to define if the LCA and LCB is near of not of a metallic surface.

15.8.3 LCSC control register 2 (LCSC_CR2)

Address offset: 0x08

Reset value: 0x0000 8000

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

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

Bits 15:8 **LCT_DAMP_THRES[7:0]:** Damping threshold for LCT
The count threshold used to define if the LCT is near of not of a metallic surface.

Bits 7:0 **TAMP_PSC[7:0]**: Tamper measurement interval The tamper detection prescaler comparing to the measurement period: - 0x00: the tamper detection and the LCT measurement are never done - others: the LCT measurement is done once every TAMP_PSC sequences of measurements.

15.8.4 LCSC pulse configuration register (LCSC_PULSE_CR)

Address offset: 0x0C

Reset value: 0x0000 0070

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **LCT_PULSEWIDTH[3:0]**: Low Pulse Width for LCT
Pulse width programming for generated pulse.

Bits 7:4 **PULSETRIM[3:0]:** Low Pulse Trimming
Trimming of the Pulse Generator.

Bits 3:0 **LCAB_PULSEWIDTH[3:0]**: Low Pulse Width for LCA and LCB
Pulse width programming for generated pulse.
Note: LCAB_PULSEWIDTH must not be set to values lower than 0x5.

15.8.5 LCSC enable register (LCSC_ENR)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit 31 **LCSC_EN:** LCSC Enable
Enable (start) or Disable (stop) the LCSC Measurement.

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **CNT_OFB_WKP_IE:** LCAB Counter Out Of Bound wakeup enable
Enable the IRQ and the Wakeup event generation on a LCAB count out of bounds

Bit 2 **TAMP_IE:** Tamper Interrupt and Wakeup Enable
Enable the IRQ and the Wakeup event generation on a tamper detection.

Bit 1 **ACLKWISE_IE:** Anti Clock Wise Interrupt and Wakeup Enable
Enable the IRQ and Wakeup event generation once the ACLKWISE value is equal or greater than the ACLKWISE_THRES bits.

Bit 0 **CLKWISE_IE**: Clock Wise Interrupt and Wakeup Enable
Enable the IRQ and Wakeup event generation once the CLKWISE has reached the CLKWISE_THRES value.

15.8.6 LCSC wheel status register (LCSC_WHEEL_SR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:16 **ACLKWISE[15:0]:** Number of Anti Clock Wise revolutions

Bits 15:0 **CLKWISE[15:0]**: Number of Clock Wise revolutions

15.8.7 LCSC wheel configuration register (LCSC_CONFR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bits 31:16 **ACLKWISE_THRES:** Number of Anti Clock Wise revolutions target

Bits 15:0 **CLKWISE_THRES**: Number of Clock Wise revolutions target

15.8.8 LCSC comparator counter register (LCSC_COMP_CNT)

Address offset: 0x1C

Reset value: 0x0000 0000

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

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

Bits 27:20 **CMP_LCT_CNT[7:0]**: LCT Comparator last damping count

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:10 **CMP_LCB_CNT[7:0]**: LCB Comparator last damping count

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:0 **CMP_LCA_CNT[7:0]**: LCA Comparator last damping count

15.8.9 LCSC status register (LCSC_SR)

Address offset: 0x20

Reset value: 0x0000 0000

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

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

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **LAST_DIR[1:0]**: Lhe last direction detected:
0X: unknown (default value when LCSC is enabled)
10: clockwise
11: counterclockwise

Bits 3:2 **ACLKWISE_STATE[1:0]**:The current state of the LCSC anti clockwise FSM:
00: Q1
01: Q2
10: Q3
11: Q4

Bits 1:0 **CLKWISE_STATE[1:0]**:The current state of the LCSC clockwise FSM:
00: Q1
01: Q2
10: Q3
11: Q4

15.8.10 LCSC statistics register (LCSC_STAT)

Address offset: 0x24

Reset value: 0xFF00 00FF

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

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

Bits 31:24 **MAX_LCAB_CNT_BOUND[7:0]**:The Maximum bound of CMP_LCA_COUNT, CMP_LCB_COUNT used when monitoring the MAX_LCAB_CNT

Bits 23:16 **MIN_LCAB_CNT_BOUND[7:0]**:The Minimum bound of CMP_LCA_COUNT, CMP_LCB_COUNT used when monitoring the MIN_LCAB_CNT

Bits 15:8 **MAX_LCAB_CNT[7:0]**:The Maximum of CMP_LCA_CNT, CMP_LCB_CNT reached during the measurement

Bits 7:0 **MIN_LCAB_CNT[7:0]**:The Minimum of CMP_LCA_CNT, CMP_LCB_CNT reached during the measurement

15.8.11 LCSC version register (LCSC_VER)

Address offset: 0x40

Reset value: 0x0000 1000

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

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

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

Bits 11:8 **PROD[3:0]**: Product

Bits 11:8 **VER[3:0]**: Version

Bits 7:4 **REV[3:0]**: Revision

Bits 3:0 Reserved, must be kept at reset value.

15.8.12 LCSC interrupt status register (LCSC_ISR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT_OFB_F	TAMP_F	ACLKWISE_F	CLKWISE_F
												rw1c	rw1c	rw1c	rw1c

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

Bit 4 **CNT_OFB_F:** Out of Bound Counter Flag
0: Out of Bound not detected
1: Out of Bound detected

Bit 3 **TAMP_F:** Tamper Flag
0: tamper not detected
1: tamper detected

Bit 2 **ACLKWISE_F:** Anti Clock Wise Flag
0: counter ACLKWISE has not reached ACLKWISE_THRES
1: counter CLKWISE has reached CLKWISE_THRES

Bit 1 **CLKWISE_F**: Clock Wise Flag
0: counter CLKWISE has not reached CLKWISE_THRES
1: counter CLKWISE has reached CLKWISE_THRES

15.9 LCSC register map

The following table summarizes the LCSC registers.

The green cells indicates the register is in the VDD12o power domain. This implies those registers are not reset on Deepstop exit.

Table 39. LCSC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LCSC_CR0	Res.	Res.	Res.	Res.	Res.	TICAP			Res.	Res.	TCAP						Res.	Res.	TMEAS													
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0
0x04	LCSC_CR1	Res.	TSTART_VCM											Res.	TREC_VCM									Res.	Res.	LCAB_DAMP_THRES							
	Reset value	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0
0x08	LCSC_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCT_DAMP_THRES								TAMP_PSC							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	LCSC_PULSE_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCT_PULSE_WIDTH				PULSETRIM				LCAB_PULSE_WIDTH			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
0x10	LCSC_ENR	LCSC_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT_OFB_WKP_IE	TAMP_IE	ACLKWISE_IE	CLKWISE_IE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	LCSC_WHEEL_SR	ACLKWISE																CLKWISE															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	LCSC_CONFR	ACLKWISE_THRES																CLKWISE_THRES															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	LCSC_COMP_CTN	Res.	Res.	Res.	Res.	CMP_LCT_CNT								Res.	Res.	CMP_LCB_CNT								Res.	Res.	CMP_LCA_CNT							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	LCSC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LAST_DIR		ACLKWISE_STATE		CLKWISE_STATE	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	LCSC_STAT	MAX_LCAB_CNT_BOUND								MIN_LCAB_CNT_BOUND								MAX_LCAB_CNT								MIN_LCAB_CNT							
	Reset value	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x40	LCSC_VER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROD				VER				REV				Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 39. LCSC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x44	LCSC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT_OFB_F	TAMP_F	ACLKWISE_F	CLKWISE_F
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

16 Liquid crystal display controller (LCD)

16.1 Introduction

The STM32WL33xx embeds an LCD controller. LCD control is not available on STM32WL30xx and STM32WL31xx devices.

The LCD controller is a digital controller/driver for monochrome passive liquid crystal display (LCD) with up to 8 common terminals and up to 16 segment terminals to drive 64 (16x4) or 96 (12x8) LCD picture elements (pixels). The exact number of terminals depends on the device pinout as described in the datasheet.

The LCD is made up of several segments (pixels or complete symbols) which can be turned visible or invisible. Each segment consists of a layer of liquid crystal molecules aligned between two electrodes. When a voltage greater than a threshold voltage is applied across the liquid crystal, the segment becomes visible. The segment voltage must be alternated to avoid an electrophoresis effect in the liquid crystal (which degrades the display). The waveform across a segment must then be generated so as to avoid having a direct current (DC).

16.2 LCD main features

- Highly flexible frame rate control.
- Supports Static, 1/2, 1/3, 1/4 and 1/8 duty.
- Supports Static, 1/2, 1/3 and 1/4 bias.
- Double buffered memory allows data in LCD_RAM registers to be updated at any time by the application firmware without affecting the integrity of the data displayed.
  - LCD data RAM of up to 16 x 32-bit registers which contain pixel information (active/inactive)
- Software selectable LCD output voltage (contrast) from V_{LCDmin} to V_{LCDmax}.
- No need for external analog components:
  - A step-up converter is embedded to generate an internal V_{LCD} voltage higher than V_{DD}
  - Software selection between external and internal V_{LCD} voltage source. In case of an external source, the internal boost circuit is disabled to reduce power consumption
  - A resistive network is embedded to generate intermediate V_{LCD} voltages
  - The structure of the resistive network is configurable by software to adapt the power consumption to match the capacitive charge required by the LCD panel.
  - Integrated voltage output buffers for higher LCD driving capability
- The contrast can be adjusted using two different methods:
  - When using the internal step-up converter, the software can adjust V_{LCD} between V_{LCDmin} and V_{LCDmax}.
  - Programmable dead time (up to 8 phase periods) between frames.
- Full support of Low power modes: the LCD controller can be displayed in DeepStop mode or can be fully disabled to reduce power consumption

- Built in phase inversion for reduced power consumptio n and EMI. (electromagnetic interference)
- Start of frame interrupt to synchronize the software when updating the LCD data RAM.
- Blink capability:
  - Up to 1, 2, 3, 4, 8 or all pixels which can be programmed to blink at a configurable frequency.
  - Software adjustable blink frequency to achieve around 0.5 Hz, 1 Hz, 2 Hz or 4 Hz.
- Used LCD segment and common pins should be configured as GPIO alternate functions and unused segment and common pins can be used for general purpose I/O or for another peripheral alternate function.

Note: 1 When the LCD relies on the internal step-up converter, the VLCD pin should be connected to V_{SS} with a capacitor. Its typical value is 1 µF (see C_{EXT} value in the product datasheets for futher information).

16.3 Glossary

Bias: Number of voltage levels used when driving an LCD. It is defined as 1/(number of voltage levels used to drive an LCD display - 1).

Boost circuit: Contrast controller circuit

Common: Electrical connection terminal connected to several segments (16 segments).

Duty ratio: Number defined as 1/(number of common terminals on a given LCD display).

Frame: One period of the waveform written to a segment.

Frame rate: Number of frames per second, that is, the number of times the LCD segments are energized per second.

LCD: (liquid crystal display) a passive display panel with terminals leading directly to a segment.

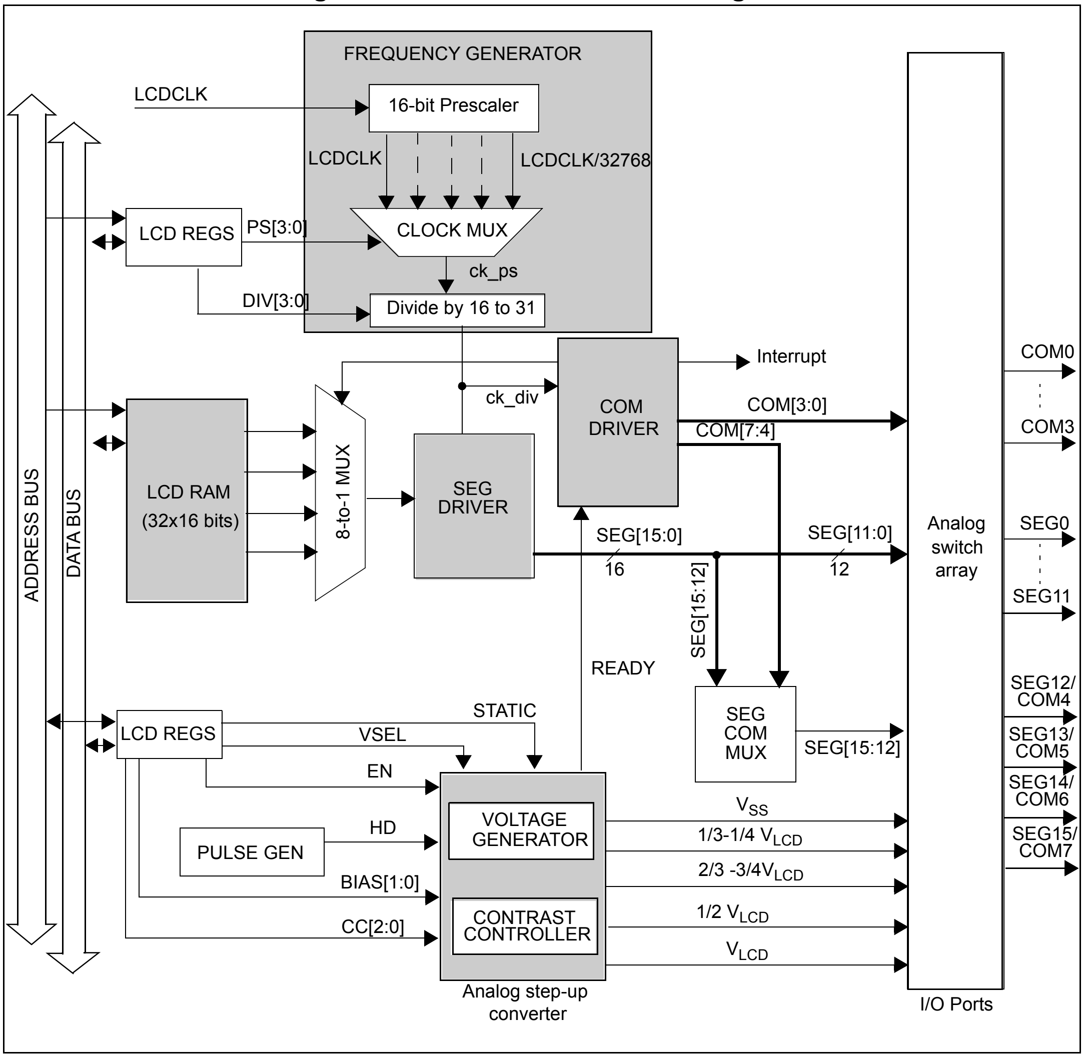
Segment: The smallest viewing element (a single bar or dot that is used to help create a character on an LCD display).

16.4 LCD functional description

16.4.1 General description

The LCD controller has five main blocks (see *Figure 38*):

Figure 38. LCD controller block diagram



Note: LCDCLK is the same as RTCCLK. Refer to the RTC/LCD clock description in the RCC section of this manual.

The frequency generator allows you to achieve various LCD frame rates starting from an LCD input clock frequency (LCDCLK) which is 32 kHz.

3 different clock sources can be used to provide the LCD clock (LCDCLK/RTCCLK):

- 32 kHz Low speed external RC (LSE)
- 32 kHz Low speed internal RC (LSI)
- The CLK_ROOT_DIV clock divided by 512. In this case, the slow clock is not available in Deepstop low power mode and it has not be used for peripherals working in Deepstop low power mode.

16.4.2 Frequency generator

This clock source must be stable in order to obtain accurate LCD timing and hence minimize DC voltage offset across LCD segments. The input clock LCDCLK can be divided by any value from 1 to 2^{15}x 31 (see *Section 16.6.2: LCD frame control register (LCD_FCR) on page 323*). The frequency generator consists of a prescaler (16-bit ripple counter) and a 16 to 31 clock divider. The PS[3:0] bits, in the LCD_FCR register, select LCDCLK divided by $2^{PS[3:0]}$. If a finer resolution rate is required, the DIV[3:0] bits, in the LCD_FCR register, can be used to divide the clock further by 16 to 31. In this way you can roughly scale the frequency, and then fine-tune it by linearly scaling the clock with the counter. The output of the frequency generator block is f_{ck_div} which constitutes the time base for the entire LCD controller. The ck_div frequency is equivalent to the LCD phase frequency, rather than the frame frequency (they are equal only in case of static duty). The frame frequency (f_{frame}) is obtained from f_{ck_div} by dividing it by the number of active common terminals (or by multiplying it for the duty). Thus the relation between the input clock frequency (f_{LCDCLK}) of the frequency generator and its output clock frequency f_{ck_div} is:

$$f_{ckdiv} = \frac{f_{LCDCLK}}{2^{PS} \times (16 + DIV)}$$

$$f_{frame} = f_{ckdiv} \times duty$$

This makes the frequency generator very flexible. An example of frame rate calculation is shown in *Table 40*.

Table 40. Example of frame rate calculation

LCDCLK	PS[3:0]	DIV[3:0]	Ratio	Duty	f_{frame}
32.768 kHz	3	1	136	1/8	30.12 Hz
32.768 kHz	4	1	272	1/4	30.12 Hz
32.768 kHz	4	6	352	1/3	31.03 Hz
32.768 kHz	5	1	544	1/2	30.12 Hz
32.768 kHz	6	1	1088	static	30.12 Hz
32.768 kHz	1	4	40	1/8	102.40 Hz
32.768 kHz	2	4	80	1/4	102.40 Hz
32.768 kHz	2	11	108	1/3	101.14 Hz
32.768 kHz	3	4	160	1/2	102.40 Hz
32.768 kHz	4	4	320	static	102.40 Hz

The frame frequency must be selected to be within a range of around ~30 Hz to ~100 Hz and is a compromise between power consumption and the acceptable refresh rate. In addition, a dedicated blink prescaler selects the blink frequency. This frequency is defined as:

$f_{BLINK} = f_{ck\_div}/2^{(BLINKF + 3)}$,

ith BLINKF[2:0] = 0, 1, 2, ..,7

The blink frequency achieved is in the range of 0.5 Hz, 1 Hz, 2 Hz or 4 Hz.

16.4.3 Common driver

Common signals are generated by the common driver block(see *Figure 38*).

COM signal bias

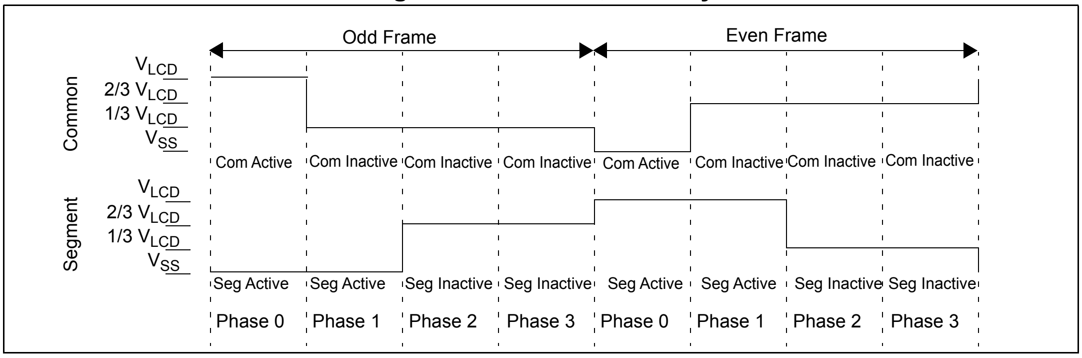
Each COM signal has identical waveforms, but different phases. It has its max amplitude V_{LCD} or V_{SS} only in the corresponding phase of a frame cycle, while during the other phases, the signal amplitude is:

- 1/4 V_{LCD} or 3/4 V_{LCD} in case of 1/4 bias
- 1/3 V_{LCD} or 2/3 V_{LCD} in case of 1/3 bias
- and 1/2 V_{LCD} in case of 1/2 bias.

Selection between 1/2, 1/3 and 1/4 bias mode can be done through the BIAS bits in the LCD_CR register.

A pixel is activated when both of its corresponding common and segment lines are active during the same phase, it means when the voltage difference between common and segment is maximum during this phase. Common signals are phase inverted in order to reduce EMI. As shown in *Figure 39*, with phase inversion, there is a mean voltage of 1/2 V_{LCD} at the end of every odd cycle.

Figure 39. 1/3 bias, 1/4 duty



In case of 1/2 bias (BIAS = 01) the V_{LCD} pin generates an intermediate voltage equal to 1/2 V_{LCD} for odd and even frames (see *Figure 42*).

COM signal duty

Depending on the DUTY[2:0] bits in the LCD_CR register, the COM signals are generated with static duty (see *Figure 41*), 1/2 duty (see *Figure 42*), 1/3 duty (see *Figure 43*), 1/4 duty (see *Figure 44*) or 1/8 duty (see *Figure 45*).

COM[*n*] *n*[0 to 7] is active during phase *n* in the odd frame, so the COM pin is driven to V_{LCD},

During phase *n* of the even frame the COM pin is driven to V_{SS}.

In the case of 1/3 or 1/4) bias:

- COM[*n*] is inactive during phases other than n so the COM pin is driven to 1/3 (1/4) V_{LCD} during odd frames and to 2/3 (3/4) V_{LCD} during even frames

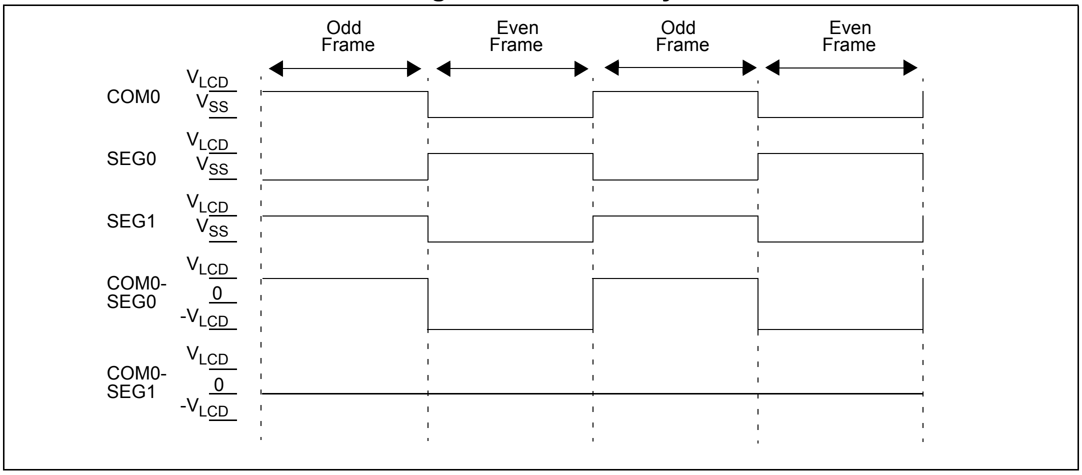
In the case of 1/2 bias:

- If COM[*n*] is inactive during phases other than n, the COM pin is always driven (odd and even frame) to 1/2 V_{LCD}.

When static duty is selected, the segment lines are not multiplexed, which means that each segment output corresponds to one pixel. In this way only up to 4 pixels can be driven. COM[0] is always active while COM[7:1] are not used and are driven to V_{SS}.

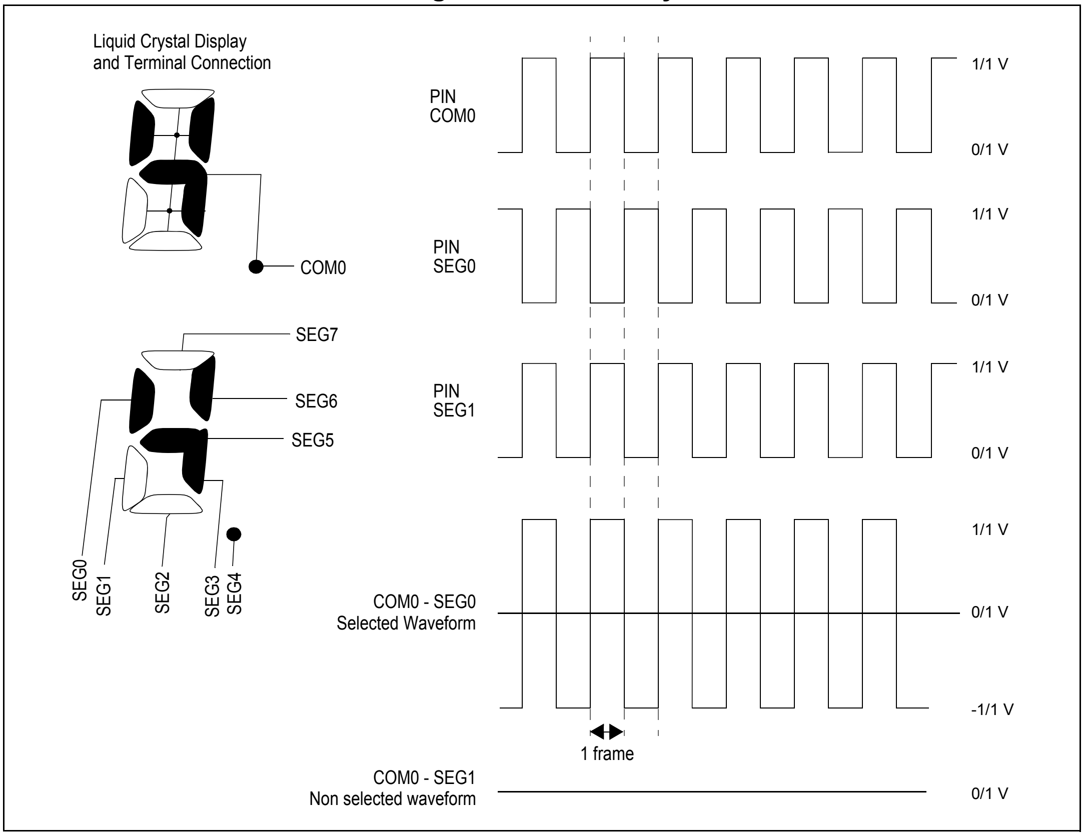
When the LCDEN bit in the LCD_CR register is reset, all common lines are pulled down to V_{SS} and the ENS flag in the LCD_SR register becomes 0. Static duty means that COM[0] is always active and only two voltage levels are used for the segment and common lines: V_{LCD} and V_{SS}. A pixel is active if the corresponding SEG line has a voltage opposite to that of the COM, and inactive when the voltages are equal. In this way the LCD has maximum contrast (see *Figure 40*, *Figure 41*). In the *Figure 40* pixel 0 is active while pixel 1 is inactive.

Figure 40. Static duty



In each frame there is only one phase, this is why f_{frame} is equal to f_{LCD}. If 1/4 duty is selected there are four phases in a frame in which COM[0] is active during phase 0, COM[1] is active during phase 1, COM[2] is active during phase 2, and COM[3] is active during phase 3.

Figure 41. Static duty

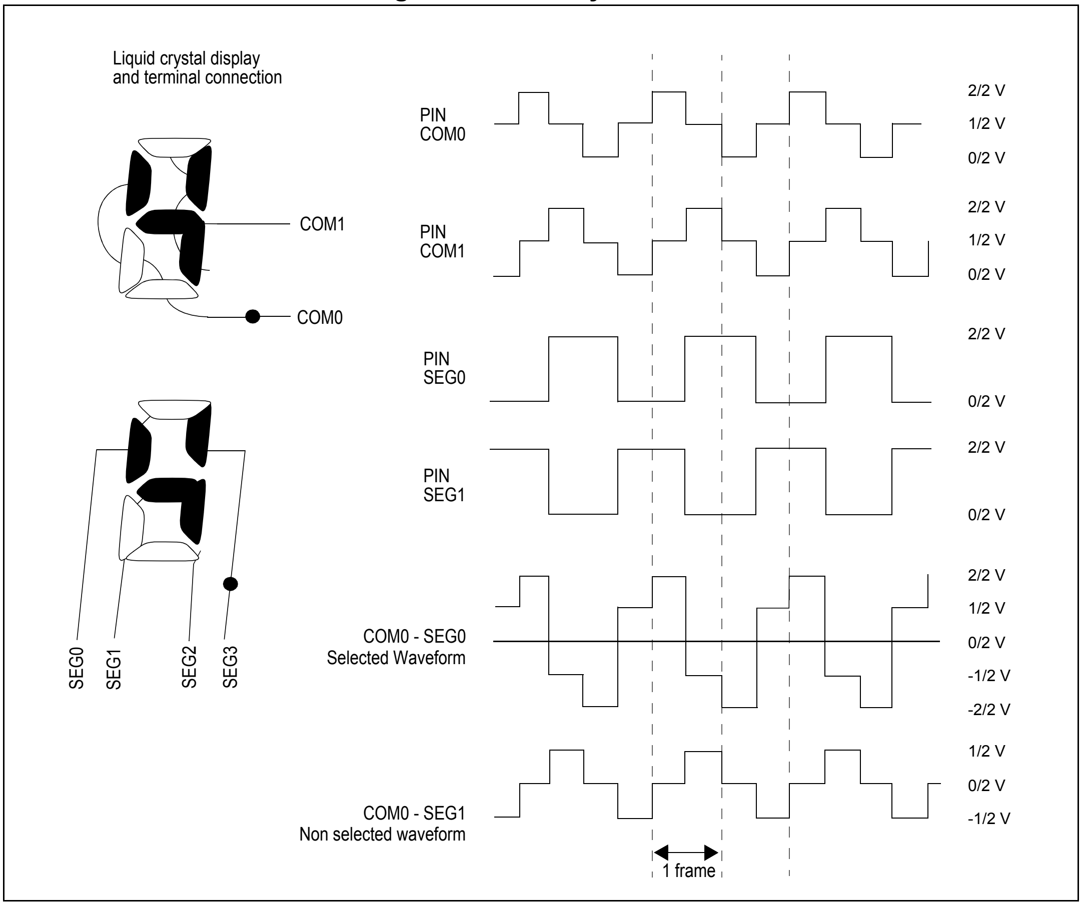


In this mode, the segment terminals are multiplexed and each of them control four pixels. A pixel is activated only when both of its corresponding SEG and COM lines are active in the same phase. In case of 1/4 duty, to deactivate pixel 0 connected to COM[0] the SEG[0] needs to be inactive during the phase 0 when COM[0] is active. To activate pixel16 connected to COM[1] the SEG[0] needs to be active during phase 1 when COM[1] is active (see *Figure 44*). To activate pixels from 0 to 15 connected to COM[0], SEG[0:15] need to be active during phase 0 when COM[0] is active. These considerations can be extended to the other pixels.

8 to 1 mux

When COM[0] is active the common driver block, also drives the 8 to 1 mux shown in *Figure 38* in order to select the content of first two RAM register locations. When *COM[7]* is active, the output of the 8 to 1 mux is the content of the last two RAM locations.

Figure 42. 1/2 duty, 1/2 bias



16.4.4 Segment driver

The segment driver block controls the SEG lines according to the pixel data coming from the 8 to 1 mux driven in each phase by the common driver block.

In the case of 1/4 or 1/8 duty

When COM[0] is active, the pixel information (active/inactive) related to the pixel connected to COM[0] (content of the first two LCD_RAM locations) goes through the 8 to 1 mux.

The SEG[*n*] pin *n* [0 to 15] is driven to V_{SS} (indicating pixel *n* is active when COM[0] is active) in phase 0 of the odd frame,

The SEG[*n*] pin is driven to V_{LCD} in phase 0 of the even frame. If pixel *n* is inactive then the SEG[*n*] pin is driven to 2/3 (2/4) V_{LCD} in the odd frame or 1/3 (2/4) V_{LCD} in the even frame (current inversion in V_{LCD} pad). (see *Figure 39*)

In case of 1/2 bias, if the pixel is inactive the SEG[*n*] pin is driven to V_{LCD} in the odd and to V_{SS} in the even frame.

When the LCD controller is disabled (LCDEN bit cleared in the LCD_CR register) then the SEG lines are pulled down to V_{SS}.

Figure 43. 1/3 duty, 1/3 bias

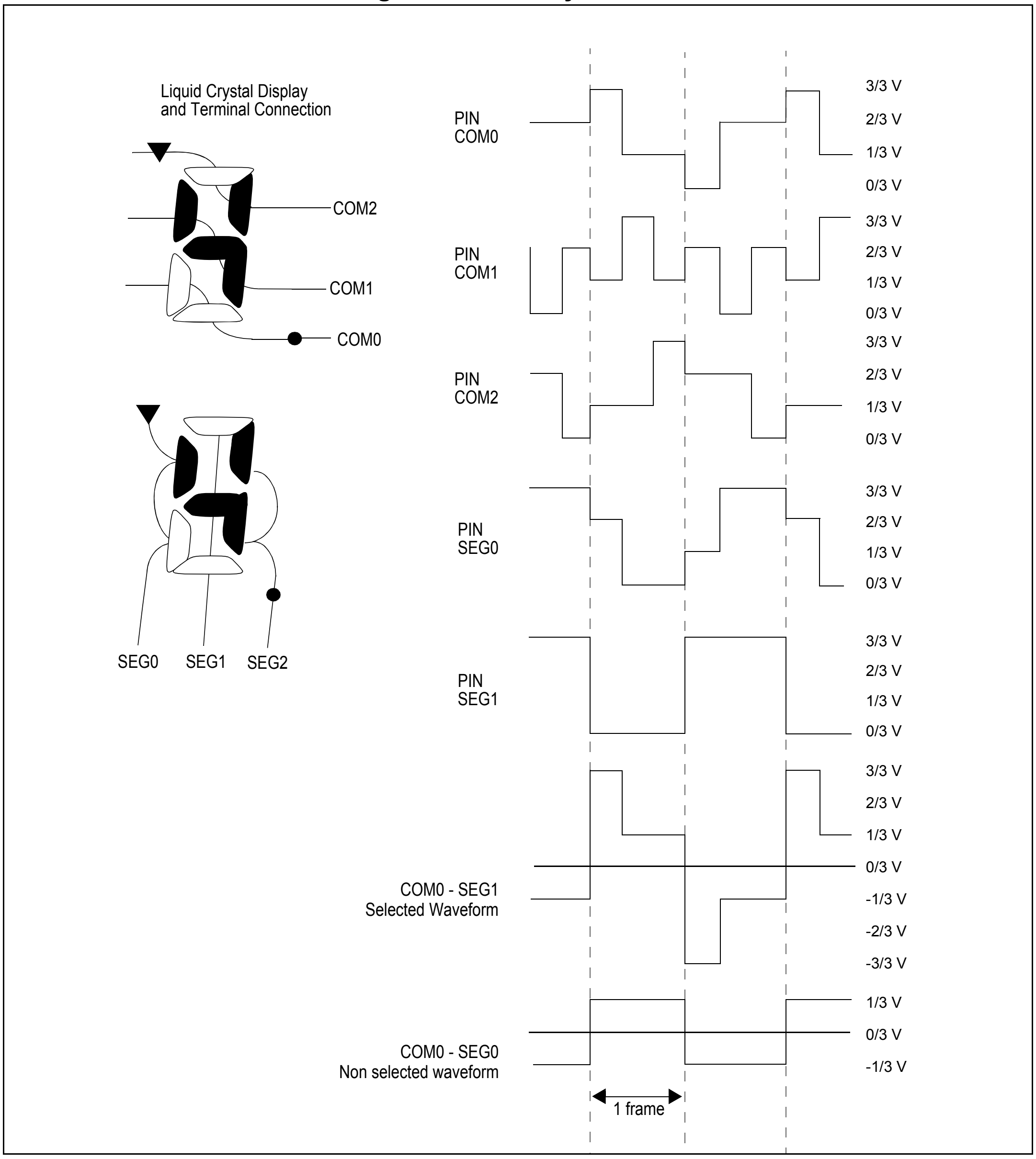


Figure 44. 1/4 duty, 1/3 bias

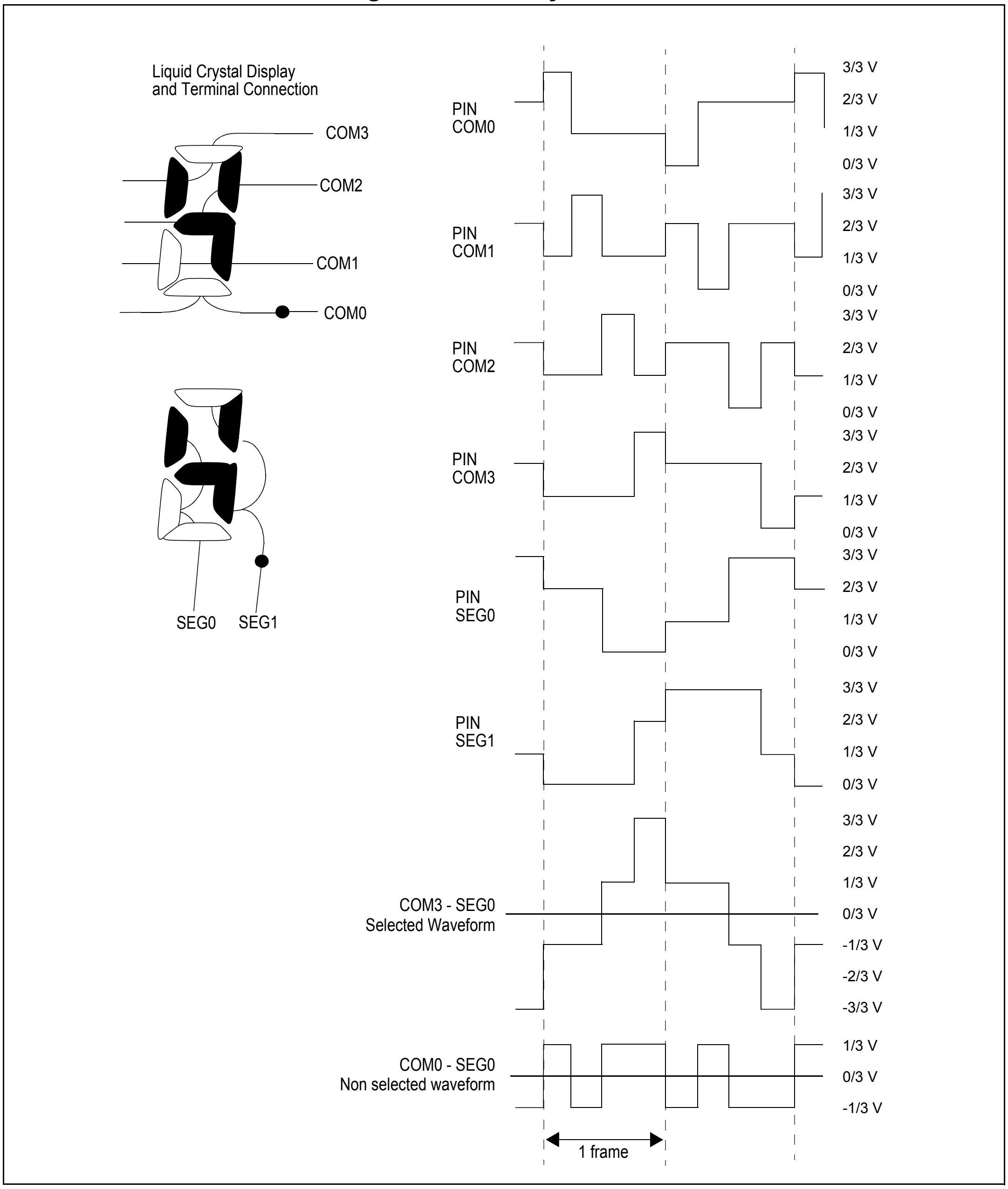
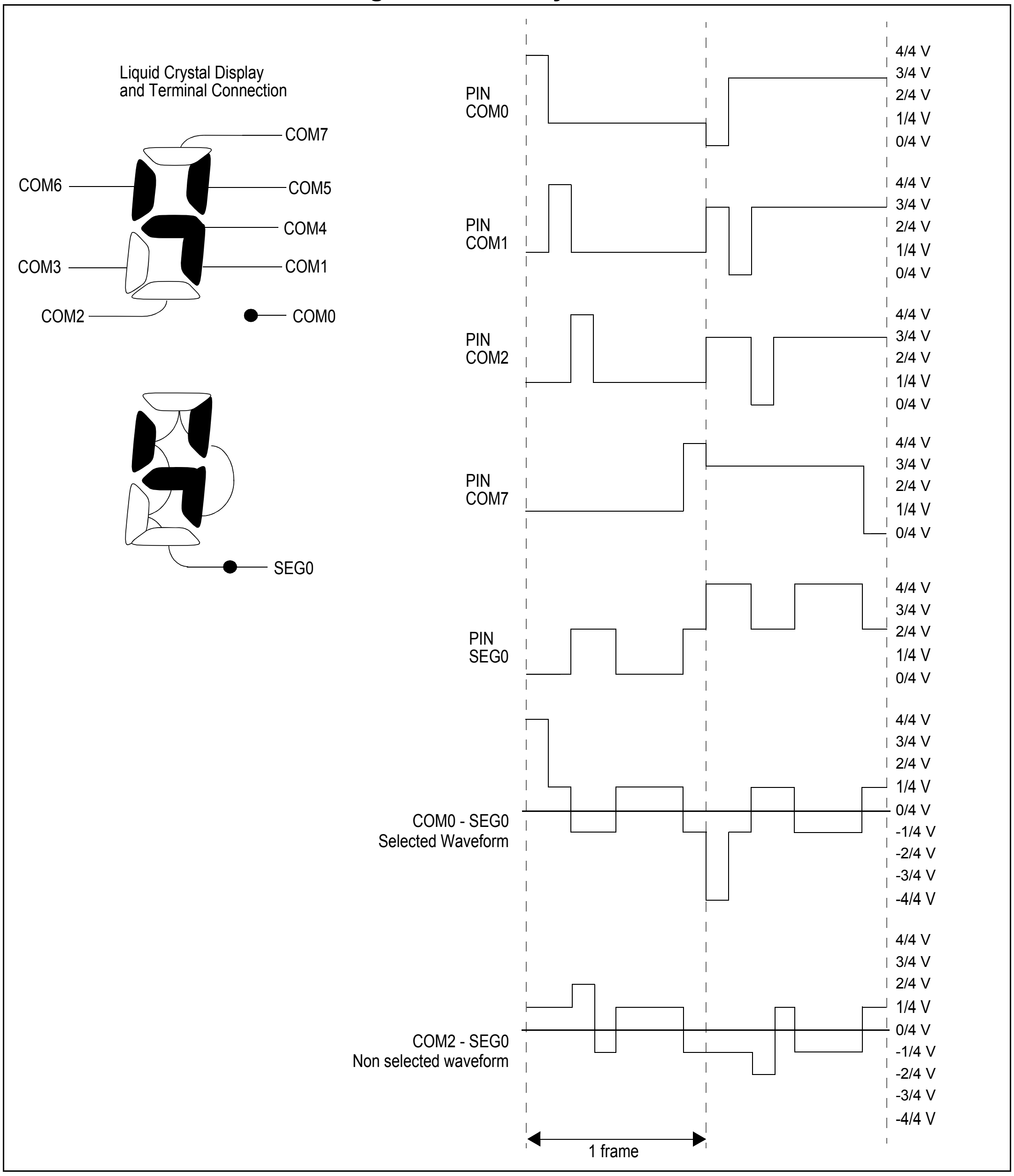


Figure 45. 1/8 duty, 1/4 bias



Blink

The segment driver also implements a programmable blink feature to allow some pixels to continuously switch on at a specific frequency. The blink mode can be configured by the BLINK[1:0] bits in the LCD_FCR register, making possible to blink up to 1, 2, 4, 8 or all pixels (see *Section 16.6.2: LCD frame control register (LCD_FCR)*). The blink frequency can be selected from eight different values using the BLINKF[2:0] bits in the LCD_FCR register.

Table 41 gives examples of different blink frequencies (as a function of ck_div frequency).

Table 41. Blink frequency

BLINKF[2:0] bits			ck_div frequency (with LCDCLK frequency of 32.768 kHz)			
			32 Hz	**64 Hz**	**128 Hz**	**256 Hz**
0	0	0	4.0 Hz	N/A	N/A	N/A
0	0	1	2.0 Hz	4.0 Hz	N/A	N/A
0	1	0	**1.0 Hz**	2.0 Hz	4.0 Hz	N/A
0	1	1	0.5 Hz	**1.0 Hz**	2.0 Hz	4.0 Hz
1	0	0	0.25 Hz	0.5 Hz	**1.0 Hz**	2.0 Hz
1	0	1	N/A	0.25 Hz	0.5 Hz	**1.0 Hz**
1	1	0	N/A	N/A	0.25 Hz	0.5 Hz
1	1	1	N/A	N/A	N/A	0.25 Hz

16.4.5 Voltage generator and contrast control

LCD supply source

The LCD power supply source may come from either the internal step-up converter or from an external voltage applied on the Vlcd pin. Internal or external voltage source can be selected using the VSEL bit in the LCD_CR register. In case of external source selected, the internal boost circuit (step-up converter) is disabled to reduce power consumption.

When the step-up converter is selected as Vlcd source, the V_{LCD} value can be chosen among a wide set of values from V_{LCDmin} to V_{LCDmax} by means of CC[2:0] (Contrast Control) bits inside LCD_FCR (see *Section 16.6.2*) register. New values of V_{LCD} takes effect every beginning of a new frame.

When external power source is selected as Vlcd source, the Vlcd voltage must be choosen in the range of Vlcdmin to Vlcdmax (see datasheets). The contrast can then be controlled by programming a dead time between frames (see *Deadtime on page 317*)

A specific software sequence must be performed to configure the LCD depending on the LCD power supply source to be used. Here we consider the LCD controller is disabled prior to the configuration sequence.

In case the internal step-up converter is used (capacitor C_{EXT} on VLCD pin is required):

- Configure the VLCD pin as alternate function LCD in the GPIO_AFR register.
- Wait for the external capacitor C_{EXT} to be charged (C_{EXT} connected to the VLCD pin, approximately 2 ms for C_{EXT} = 1 uF)
- Set voltage source to internal source by resetting VSEL bit in the LCD_CR register.
- Enable the LCD controller by setting LCDEN bit in the LCD_CR register.

In case of LCD external power source is used:

- Set voltage source to external source by setting VSEL bit in the LCD_CR register
- Configure the VLCD pin as alternate function LCD in the GPIO_SFR
- Enable the LCD controller by setting LCDEN bit in the LCD_CR register.

LCD intermediate voltages

The LCD voltage generator generates up to three intermediate voltage levels (1/3 V_{LCD}, 2/3 V_{LCD} or 1/4 V_{LCD}, 2/4 V_{LCD}, 3/4 V_{LCD}) between V_{SS} and V_{LCD} in case of 1/3 (1/4) bias and only one voltage level (1/2 V_{LCD}) between V_{SS} and V_{LCD} in case of 1/2 bias.

In case of 1/2 bias, one voltage level (1/2 VLCD) is generated and node b voltage is 1/2 VLCD

In case of 1/3 bias, two intermediate voltage levels (1/3 VLCD, 2/3 VLCD) are generated

- Node a is 1/3 VLCD
- Node b is 2/3 VLCD

In case of 1/4 bias, three intermediate voltage levels (1/4 VLCD, 1/2 VLCD and 3/4 VLCD) are generated

- Node a is 1/4 VLCD
- Node b is 1/2 VLCD
- Node c is 3/4 VLCD

LCD drive selection

1. High drive and low drive resistive network

Internal resistor networks are used to generate all intermediate voltages (see *Figure 46*). Actually, one with low value resistors (RL) and one with high value resistors (RH) which are respectively used to increase the current during transitions and to reduce power consumption in static state.

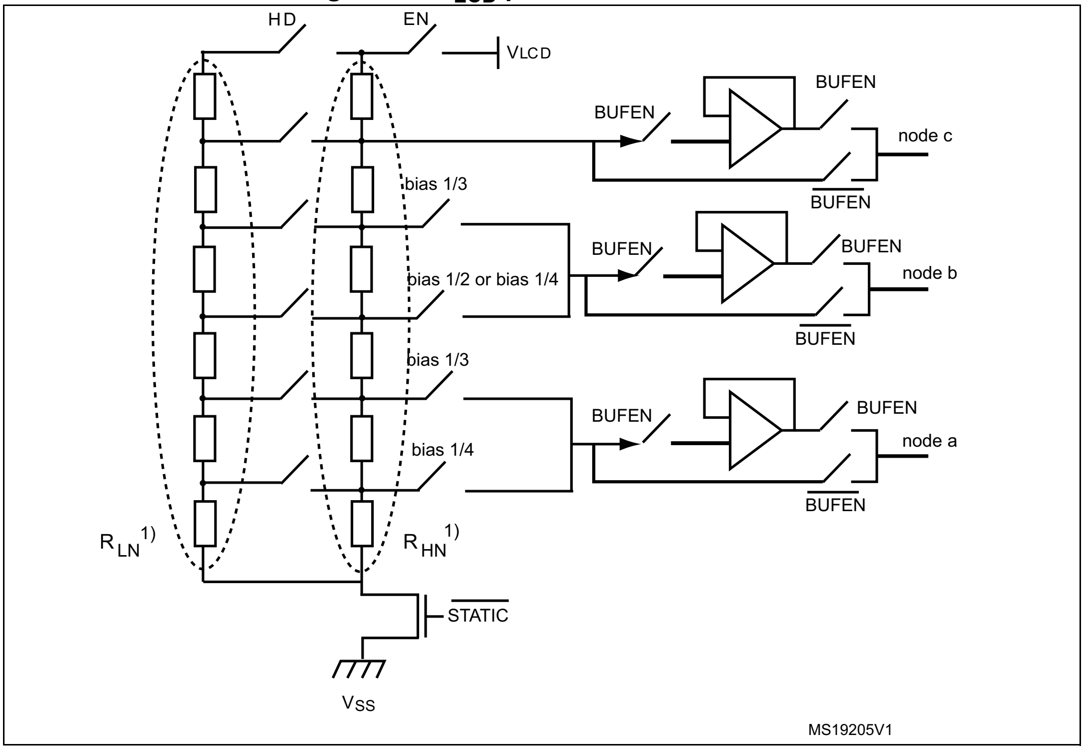
The EN switch allows the following rules :

If LCDEN bit in the LCD_CR register is set, the EN switch is closed.

When clearing the LCDEN bit in the LCD_CR register, the EN switch is open at the end of the even frame in order to avoid a medium voltage level different from 0 considering the entire frame odd plus even.

The PON[2:0] (Pulse ON duration) bits in the LCD_FCR register configure the time during which R_L is enabled (see *Figure 38*) through the HD (high drive) switch when the levels of the common and segment lines change. A short drive time leads to lower power consumption, but displays with high internal resistance may need a longer drive time to achieve satisfactory contrast.

Figure 46. V_{LCD} pin for 1/2 1/3 1/4 bias



The R_L divider can be always switched on using the HD bit in the LCD_FCR configuration register (see *Section 16.6.2*).

The HD switch follows the rules described below:

- If the HD bit and the PON[2:0] bits in the LCD_FCR register are reset, then HD switch is open.
- If the HD bit in the LCD_FCR register is reset and the PON[2:0] bits in the LCD_FCR are different from 00 then, the HD switch is closed during the number of pulses defined in the PON[2:0] bits.
- If HD bit in the LCD_FCR register is 1 then HD switch is always closed.

2. Buffered mode

When voltage output buffers are enabled by setting BUFEN bit in the LCD_CR register, LCD driving capability is improved as buffers prevent the LCD capacitive loads from loading the resistor bridge unacceptably and interfering with its voltage generation. As a result we obtain more stable intermediate voltage levels thus improving RMS voltage applied to the LCD pixels.

In buffer mode, intermediate voltages are generated by the high value resistor bridge Rhn to reduce power consumption, the low value resistror bridge Rln is automatically disabled whatever the HD bit or PON bits configuration.

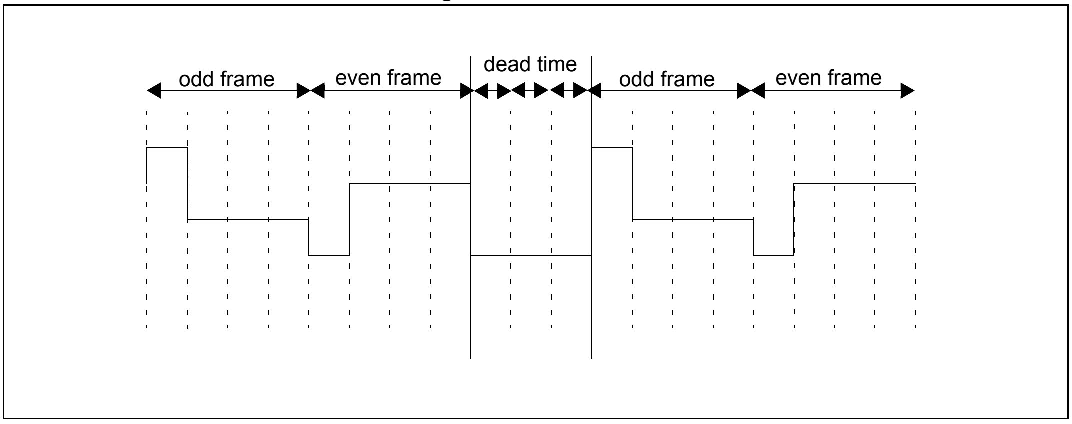
Buffers can be used independently of the Vlcd supply source (internal or external) but can only be enabled or disabled when LCD controller is not activated.

After the LCDEN bit is activated, the RDY bit is set in the LCD_SR register to indicate that voltage levels are stable and the LCD controller can start to work.

Deadtime

In addition to using the CC[2:0] bits, the contrast can be controlled by programming a dead time between each frame. During the dead time the COM and SEG values are put to V_{SS}. The DEAD[2:0] bits in the LCD_FCR register can be used to program a time of up to eight phase periods. This dead time reduces the contrast without modifying the frame rate.

Figure 47. Deadtime



16.4.6 Double buffer memory

Using its double buffer memory the LCD controller ensures the coherency of the displayed information without having to use interrupts to control LCD_RAM modification.

The application software can access the first buffer level (LCD_RAM) through the APB interface. Once it has modified the LCD_RAM, it sets the UDR flag in the LCD_SR register. This UDR flag (update display request) requests the updated information to be moved into the second buffer level (LCD_DISPLAY).

This operation is done synchronously with the frame (at the beginning of the next frame), until the update is completed, the LCD_RAM is write protected and the UDR flag stays high. Once the update is completed another flag (UDD - Update Display Done) is set and generates an interrupt if the UDDIE bit in the LCD_FCR register is set.

The time it takes to update LCD_DISPLAY is, in the worst case, one odd and one even frame.

The update does not occur (UDR = 1 and UDD = 0) until the display is enabled (LCDEN = 1)

16.4.7 LCD controller low power modes

the LCD controller can be displayed in Stop mode or can be fully disabled to reduce power consumption.

Table 42. LCD behavior in low power modes

Mode	Description
Deep	The LCD is still active
Shutdown	The LCD is not active

16.4.8 COM and SEG multiplexing

Output pins versus duty modes

The output pins consists of:

- SEG[15:0]
- COM[3:0]

Depending on the duty configuration, the COM and SEG output pins may have different functions:

- In static, 1/2, 1/3 and 1/4 duty modes there are up to 16 SEG pins and respectively 1, 2, 3 and 4 COM pins
- In 1/8 duty mode (DUTY[2:0] = 100), the COM[7:4] outputs are available on the SEG[15:12]pins, reducing to the number of available segments to 12.

Summary of COM and SEG functions versus duty and remap

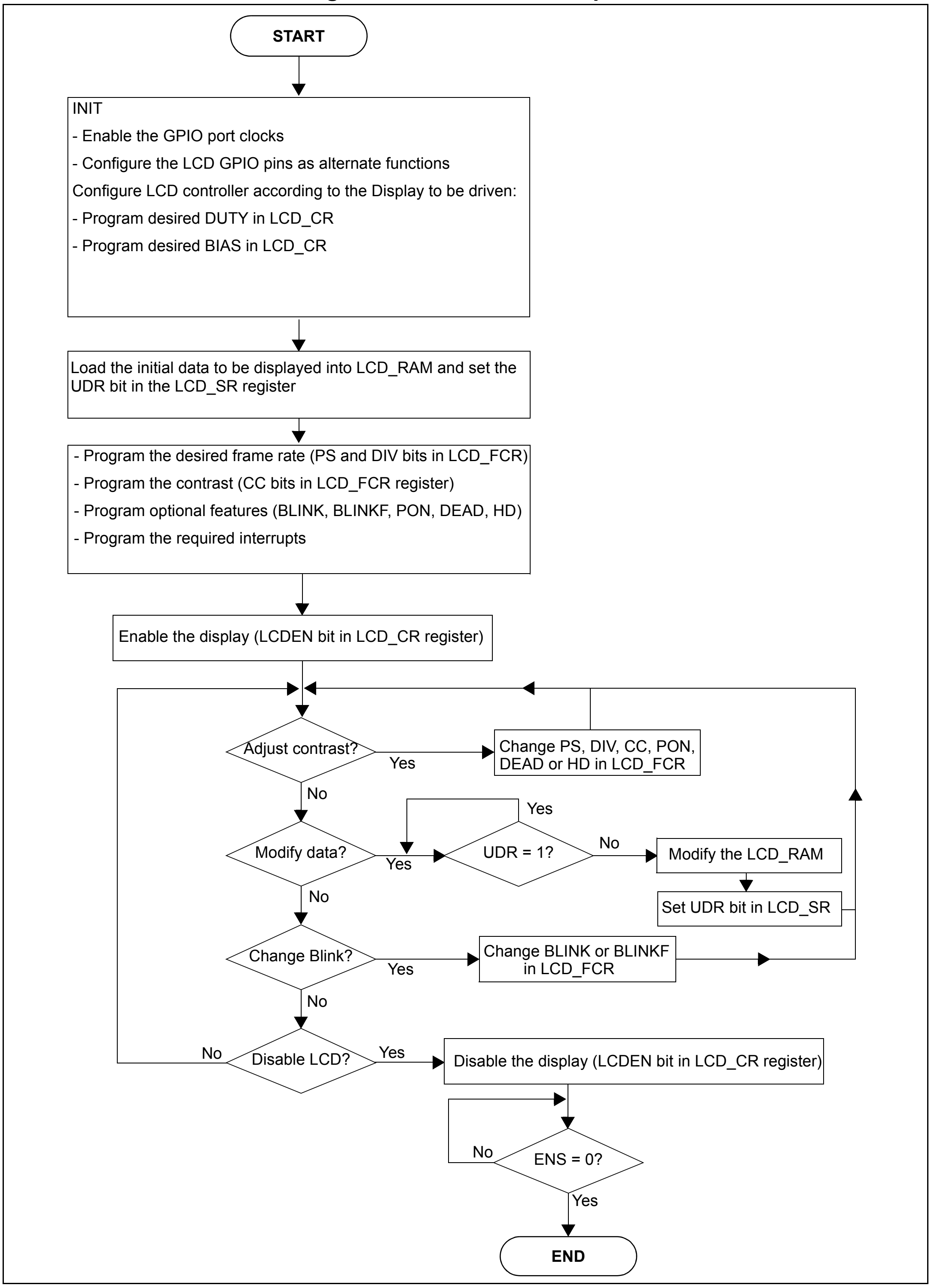
All the possible ways of multiplexing the COM and SEG functions are described in *Table 43*.

Table 43. Remapping capability

Configuration bits		Capability	Output pin	Function
DUTY				
1/8	0	12x8	SEG[15:12]	COM[7:4]
			COM[3:0]	COM[3:0]
			SEG[11:0]	SEG[11:0]
1/4	0	16x4	COM[3:0]	COM[3:0]
			SEG[15:0]	SEG[15:0]
1/3	0	16x3	COM[3]	not used
			COM[2:0]	COM[2:0]
			SEG[15:0]	SEG[15:0]
1/2	0	16x2	COM[3:2]	not used
			COM[1:0]	COM[1:0]
			SEG[15:0]	SEG[15:0]
STATIC	0	16x1	COM[3:1]	not used
			COM[0]	COM[0]
			SEG[15:0]	SEG[15:0]

16.4.9 Flowchart

Figure 48. Flowchart example



16.5 LCD interrupts

Table 44 gives the list of LCD interrupt requests.

Table 44. LCD interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Start Of Frame (SOF)	SOF	Write SOFC =1	SOFIE
Update Display Done (UDD)	UDD	Write UDDC = 1	UDDIE

Start of frame (SOF)

The LCD start of frame interrupt is executed if the SOFIE (start of frame interrupt enable) bit is set (see *Section 16.6.2: LCD frame control register (LCD_FCR)*). SOF is cleared by writing the SOFC bit to 1 in the LCD_CLR register when executing the corresponding interrupt handling vector.

Update display done (UDD)

The LCD update display interrupt is executed if the UDDIE (update display done interrupt enable) bit is set (see *Section 16.6.2: LCD frame control register (LCD_FCR)*). UDD is cleared by writing the UDDC bit to 1 in the LCD_CLR register when executing the corresponding interrupt handling vector.

Depending on the product implementation, all these interrupts events can either share the same interrupt vector (LCD global interrupt), or be grouped into 2 interrupt vectors (LCD SOF interrupt and LCD UDD interrupt). Refer to the *Vector table in the Interrupts and events section* for details.

In order to enable the LCD interrupts, the following sequence is required:

1. Configure and enable the LCD IRQ channel in the NVIC
2. Configure the LCD to generate interrupts

16.6 LCD registers

The peripheral registers have to be accessed by words (32-bit).

16.6.1 LCD control register (LCD_CR)

Address offset: 0x00

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUFEN	Res.	BIAS[1:0]		DUTY[2:0]			VSEL	LCDEN
							rw		rw	rw	rw	rw	rw	rw	rw

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

Bit 8 **BUFEN** : Voltage output buffer enable
This bit is used to enable/disable the voltage output buffer for higher driving capability.
0: Output buffer disabled
1: Output buffer enabled

Bit 7 Reserved, must be kept at reset value

Bits 6:5 **BIAS[1:0]**: Bias selector
These bits determine the bias used. Value 11 is forbidden.
00: Bias 1/4
01: Bias 1/2
10: Bias 1/3
11: Reserved

Bits 4:2 **DUTY[2:0]**: Duty selection
These bits determine the duty cycle. Values 101, 110 and 111 are forbidden.
000: Static duty
001: 1/2 duty
010: 1/3 duty
011: 1/4 duty
100: 1/8 duty
101: Reserved
110: Reserved
111: Reserved

Bit 1 **VSEL**: Voltage source selection
The VSEL bit determines the voltage source for the LCD.
0: Internal source (voltage step-up converter)
1: External source (V_{LCD} pin)

Bit 0 **LCDEN**: LCD controller enable
This bit is set by software to enable the LCD Controller/Driver. It is cleared by software to turn off the LCD at the beginning of the next frame. When the LCD is disabled all COM and SEG pins are driven to V_{SS}.
0: LCD Controller disabled
1: LCD Controller enabled

Note: The VSEL, MUX_SEG, BIAS, DUTY and BUFEN bits are write protected when the LCD is enabled (ENS bit in LCD_SR to 1).

16.6.2 LCD frame control register (LCD_FCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PS[3:0]				DIV[3:0]				BLINK[1:0]	
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BLINKF[2:0]			CC[2:0]			DEAD[2:0]			PON[2:0]			UDDIE	Res.	SOFIE	HD
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:26 Reserved, must be kept at reset value

Bits 25:22 **PS[3:0]**: PS 16-bit prescaler

These bits are written by software to define the division factor of the PS 16-bit prescaler. ck_ps = LCDCLK/(2). See *Section 16.4.2*.

0000: ck_ps = LCDCLK

0001: ck_ps = LCDCLK/2

0010: ck_ps = LCDCLK/4

...

1111: ck_ps = LCDCLK/32768

Bits 21:18 **DIV[3:0]**: DIV clock divider

These bits are written by software to define the division factor of the DIV divider. See *Section 16.4.2*.

0000: ck_div = ck_ps/16

0001: ck_div = ck_ps/17

0010: ck_div = ck_ps/18

...

1111: ck_div = ck_ps/31

Bits 17:16 **BLINK[1:0]**: Blink mode selection

00: Blink disabled

01: Blink enabled on SEG[0], COM[0] (1 pixel)

10: Blink enabled on SEG[0], all COMs (up to 8 pixels depending on the programmed duty)

11: Blink enabled on all SEGs and all COMs (all pixels)

Bits 15:13 **BLINKF[2:0]**: Blink frequency selection

000: $f_{LCD}/8$

100: $f_{LCD}/128$

001: $f_{LCD}/16$

101: $f_{LCD}/256$

010: $f_{LCD}/32$

110: $f_{LCD}/512$

011: $f_{LCD}/64$

111: $f_{LCD}/1024$

Bits 12:10 **CC[2:0]**: Contrast control

These bits specify one of the V_{LCD} maximum voltages (independent of V_{DD}). It ranges from 2.60 V to 3.51 V.

000: V_{LCD0}
100: V_{LCD4}
001: V_{LCD1}
101: V_{LCD5}
010: V_{LCD2}
110 V_{LCD6}
011: V_{LCD3}
111: V_{LCD7}

Note: Refer to the product datasheet for the V_{LCDx} values.

Bits 9:7 **DEAD[2:0]**: Dead time duration

These bits are written by software to configure the length of the dead time between frames. During the dead time the COM and SEG voltage levels are held at 0 V to reduce the contrast without modifying the frame rate.

000: No dead time
001: 1 phase period dead time
010: 2 phase period dead time
......
111: 7 phase period dead time

Bits 6:4 **PON[2:0]**: Pulse ON duration

These bits are written by software to define the pulse duration in terms of ck_ps pulses. A short pulse leads to lower power consumption, but displays with high internal resistance may need a longer pulse to achieve satisfactory contrast. Note that the pulse is never longer than one half prescaled LCD clock period.

000: 0
100: 4/ck_ps
001: 1/ck_ps
101: 5/ck_ps
010: 2/ck_ps
110: 6/ck_ps
011: 3/ck_ps
111: 7/ck_ps

PON duration example with LCDCLK = 32.768 kHz and PS=0x03:

000: 0 µs
100: 976 µs
001: 244 µs
101: 1.22 ms
010: 488 µs
110: 1.46 ms
011: 782 µs
111: 1.71 ms

Bit 3 **UDDIE**: Update display done interrupt enable

This bit is set and cleared by software.

0: LCD Update Display Done interrupt disabled
1: LCD Update Display Done interrupt enabled

Bit 2 Reserved, must be kept at reset value

Bit 1 **SOFIE**: Start of frame interrupt enable

This bit is set and cleared by software.

0: LCD Start of Frame interrupt disabled

1: LCD Start of Frame interrupt enabled

Bit 0 **HD**: High drive enable

This bit is written by software to enable a low resistance divider. Displays with high internal resistance may need a longer drive time to achieve satisfactory contrast. This bit is useful in this case if some additional power consumption can be tolerated.

0: Permanent high drive disabled

1: Permanent high drive enabled. When HD=1, then the PON bits have to be programmed to 001.

Note: The data in this register can be updated any time, however the new values are applied only at the beginning of the next frame (except for UDDIE, SOFIE that affect the device behavior immediately).

The new value of CC[2:0] bits is also applied immediately but its effect on device is delayed at the beginning of next frave by the voltage generator.

Reading this register obtains the last value written in the register and not the configuration used to display the current frame.

Note: 1 When BUFEN bit is set in the LCD_CR register, low resistor divider network is automatically disabled whatever the HD or PON[2:0] bits configuration.

16.6.3 LCD status register (LCD_SR)

Address offset: 0x08

Reset value: 0x0000 0020

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

Bits 31:6 Reserved, must be kept at reset value

Bit 5 **FCRSF**: LCD Frame Control Register Synchronization flag

This bit is set by hardware each time the LCD_FCR register is updated in the LCDCLK domain. It is cleared by hardware when writing to the LCD_FCR register.

0: LCD Frame Control Register not yet synchronized

1: LCD Frame Control Register synchronized

Bit 4 **RDY**: Ready flag

This bit is set and cleared by hardware. It indicates the status of the step-up converter.

0: Not ready

1: Step-up converter is enabled and ready to provide the correct voltage.

Bit 3 **UDD**: Update Display Done

This bit is set by hardware. It is cleared by writing 1 to the UDDC bit in the LCD_CLR register. The bit set has priority over the clear.

0: No event

1: Update Display Request done. A UDD interrupt is generated if the UDDIE bit in the LCD_FCR register is set.

Note: If the device is in STOP mode (PCLK not provided) UDD does not generate an interrupt even if UDDIE = 1.

If the display is not enabled the UDD interrupt never occurs.

Bit 2 **UDR**: Update display request

Each time software modifies the LCD_RAM it must set the UDR bit to transfer the updated data to the second level buffer. The UDR bit stays set until the end of the update and during this time the LCD_RAM is write protected.

0: No effect

1: Update Display request

Note: When the display is disabled, the update is performed for all LCD_DISPLAY locations. When the display is enabled, the update is performed only for locations for which commons are active (depending on DUTY). For example if DUTY = 1/2, only the LCD_DISPLAY of COM0 and COM1 is updated.

Note: Writing 0 on this bit or writing 1 when it is already 1 has no effect. This bit can be cleared by hardware only. It can be cleared only when LCDEN = 1

Bit 1 **SOF**: Start of frame flag

This bit is set by hardware at the beginning of a new frame, at the same time as the display data is updated. It is cleared by writing a 1 to the SOFC bit in the LCD_CLR register. The bit clear has priority over the set.

0: No event

1: Start of Frame event occurred. An LCD Start of Frame Interrupt is generated if the SOFIE bit is set.

Bit 0 **ENS**: LCD enabled status

This bit is set and cleared by hardware. It indicates the LCD controller status.

0: LCD Controller disabled.

1: LCD Controller enabled

Note: The ENS bit is set immediately when the LCDEN bit in the LCD_CR goes from 0 to 1. On deactivation it reflects the real status of LCD so it becomes 0 at the end of the last displayed frame.

16.6.4 LCD clear register (LCD_CLR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit 31:2 Reserved, must be kept at reset value

Bit 3 **UDDC:** Update display done clear

This bit is written by software to clear the UDD flag in the LCD_SR register.

0: No effect

1: Clear UDD flag

Bit 2 Reserved, must be kept at reset value

Bit 1 **SOFC:** Start of frame flag clear

This bit is written by software to clear the SOF flag in the LCD_SR register.

0: No effect

1: Clear SOF flag

Bit 0 Reserved, must be kept at reset value

16.6.5 LCD display memory (LCD_RAM)

Address offset: 0x14-0x50

Reset value: 0x0000 0000

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

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

Bits 31:0 **SEGMENT_DATA[31:0]**

Each bit corresponds to one pixel of the LCD display.

0: Pixel inactive

1: Pixel active

16.7 LCD register map

The following table summarizes the LCD registers.

Table 45. LCD register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LCD_CR	Reserved																							BUFEN	Res.	BIAS[1:0]		DUTY[2:0]			VSEL	LCDEN
	Reset value																								0		0	0	0	0	0	0	0
0x04	LCD_FCR	Reserved						PS[3:0]				DIV[3:0]				BLINK[1:0]		BLINKF[2:0]			CC[2:0]			DEAD [2:0]			PON[2:0]			UDDIE	Res.	SOFIE	HD
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
0x08	LCD_SR	Reserved																										FCRSF	RDY	UDD	UDR	SOF	ENS
	Reset value																											1	0	0	0	0	0
0x0C	LCD_CLR	Reserved																												UDDC	Res.	SOFC	Res.
	Reset value																													0		0	
0x14	LCD_RAM (COM0)	Reserved																S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18		Reserved																															
0x1C	LCD_RAM (COM1)	Reserved																S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20		Reserved																															
0x24	LCD_RAM (COM2)	Reserved																S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28		Reserved																															
0x2C	LCD_RAM (COM3)	Reserved																S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30		Reserved																															
0x34	LCD_RAM (COM4)	Reserved																				S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
																						0	0	0	0	0	0	0	0	0	0	0	0
0x38		Reserved																															

Table 45. LCD register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3C	LCD_RAM (COM5)	Reserved																				S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
																						0	0	0	0	0	0	0	0	0	0	0	0
0x40		Reserved																															
0x44	LCD_RAM (COM6)	Reserved																				S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
																						0	0	0	0	0	0	0	0	0	0	0	0
0x48		Reserved																															
0x4C	LCD_RAM (COM7)	Reserved																				S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
																						0	0	0	0	0	0	0	0	0	0	0	0
0x50		Reserved																															

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the*Register boundary addresses* table.

17 Random number generator (RNG)

The RNG is a random number generator based on a continuous analog noise that provides a 16-bit value to the host when read.

17.1 Features

- AHB slave peripheral
- Deliver 16-bit random number produced by an analog generator
- Minimum period of 1.25 µs (corresponding to 20 RNGCLK cycles) between two consecutive random number. This is automatically managed by a pulling spacer counter that adds wait-state on the AHB bus when reading occurs too closely to the previous one.
- Monitoring of the entropy of the RNG to flag abnormal behavior (generation of stable values or stable sequence of values)
- 2 clock domains:
  - RNGCLK: 16 MHz for the normalization and shifter (specific serial to 16-bit parallel conversion)
  - HCLK: AHB clock (16 MHz, 32 MHz or 64 MHz) for the AHB interface
- Can be disabled to reduce power consumption:

Power consumption and RNG:

The internal free-running oscillators are quite power consuming. It is possible to stop them when the RNG is not used.

- After a PORESETn, the internal free-running oscillators are stopped by default to limit consumption.
- When the RNG clock tree is enabled through the RCC after a PORESETn, the oscillators are automatically restarted.
- If the SW gate the RNG clock tree through the RCC, it sill not stop the internal oscillator. The SW has to first set the RNG_CR[2] = RNG_DIS to stop them.
- On the other side, the SW has to restart them by clearing the RNG_DIS bit once the RNG clock tree is re-enabled.

17.2 RNG registers map

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

Note: Despite RNG registers are addressed through AHB, only 32-bit accesses are allowed. Any 8-bit or 16-bit access generate an AHB error leading to a hard fault on Cortex-M0+.

Table 46. RNG register list

Address offset	Name	RW(1)	Reset	Description
0x00	RNG_CR	RW	0x00000000	RNG Configuration register. See *Table 47: RNG_CR register description on page 332*
0x04	RNG_SR	RO	0x00000000	RNG Status register. See *Table 48: RNG_SR register description on page 333*
0x08	RNG_VAL	RO	0x00000000	RNG 16-bit random value. See *Table 49: RNG_VAL register description on page 333*

1. These acronyms have the following meaning:

RW = Read and Write

RO = Read Only

17.3 RNG registers description

RNG_CR

This register configures the RNG.

Table 47. RNG_CR register description

Address: RNG_BASE + 0x00				
Bit	**Field name**	**Reset**	**R/W**	**Description**
1:0	RESERVED	2'h0	RW	RESERVED
2	RNG_DIS	1'h0	RW	RNG Disable bit. This bit enables or disables the random number generator. – 0: RNG is enable (default) – 1: RNG is disabled. The internal free-running oscillators are put in power-down mode and the RNG clock is stopped at the input of the block.
3	TST_CLK	1'h0	RW	RNG Test Clock bit. Writing this bit with 1b starts the logic that detects the presence of the RNG_CLK. Then wait (with a timeout of at least four RNGCLK cycles) for REVCLK = 1b in RNG_SR register. If REVCLK = 0b after timeout elapsed, it means that RNGCLK is not present and reading RNG_VAL register triggers an AHB error response. For security reason, software should check before reading random values that the RNGCLK is present. This bit is auto-cleared and always read as 0.
31:4	RESERVED	28'h0000000	R	RESERVED

RNG_SR

This register provides status flags of the RNG.

Table 48. RNG_SR register description

Address: RNG_BASE + 0x04				
Bit	**Field name**	**Reset**	**R/W**	**Description**
0	RNGRDY	1'h0	R	New Random Value Ready. – 0: the RNG_VAL register value is not yet valid. If performing a read access to RNG_VAL, the host is put on hold (by wait-states insertion on the AHB bus) until a random value is available. – 1: the RNG_VAL register contains a valid random number. This bit remains at 0b when the RNG is disabled (RNGDIS bit = 1b in RNG_CR)
1	REVCLK	1'h0	R	RNGCLK Clock Reveal bit. A write with 1b to bit TSTCLK in RNG_CR resets this bit. If the RNGCLK is present, this bit is 1b after four RNGCLK cycles after the end of the write to RNG_CR.If REVCLK = 0b after this period, it means the RNGCLK is not present and reading RNG_VAL triggers a AHB error response.
2	FAULT	1'h0	RC_W1	Fault Reveal bit. This bit is set by hardware when a faulty sequence of bits occurs. The faulty sequences are: – sequence of more than 32 consecutive bits of same value (0b or 1b) – sequence of more than 16 consecutive alternation of 0b and 1b (010101...01b). Writing this bit with 1b clear it. Writing with 0b has no effect.
31:3	RESERVED	29'h00000000	R	RESERVED

RNG_VAL

This register delivers a 16-bit random value when read. After being read, this register delivers a new random value only after 20 cycles of RNGCLK. If the host performs a new read before the period has elapsed, the RNG inserts a wait-state on the AHB bus.

Table 49. RNG_VAL register description

Address: RNG_BASE + 0x08				
Bit	**Field name**	**Reset**	**R/W**	**Description**
15:0	RANDOM_VALUE	16'h----	R	Random Value.
31:16	RESERVED	16'h0000	R	RESERVED

18 AES hardware accelerator (AES)

18.1 Introduction

The AES hardware accelerator (AES) encrypts or decrypts data, using an algorithm and implementation fully compliant with the advanced encryption standard (AES) defined in Federal information processing standards (FIPS) publication 197.

The peripheral supports CTR, GCM, GMAC, CCM, ECB, and CBC chaining modes for key sizes of 128 or 256 bits.

AES is an AMBA AHB slave peripheral accessible through 32-bit single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels required).

18.2 AES main features

- Compliance with NIST *"Advanced encryption standard (AES)*, *FIPS publication 197*" from November 2001
- 128-bit data block processing
- Support for cipher key lengths of 128-bit and 256-bit
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block of data with, respectively, 128-bit or 256-bit key
- Integrated round key scheduler to compute the last round key for ECB/CBC decryption
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only
- 256-bit register for storing the cryptographic key (eight 32-bit registers)
- 128-bit register for storing initialization vector (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control with support of single-transfer direct memory access (DMA) using two channels (one for incoming data, one for processed data)
- Data-swapping logic to support 1-, 8-, 16- or 32-bit data
- Possibility for software to suspend a message if AES needs to process another message with a higher priority, then resume the original message

18.3 AES implementation

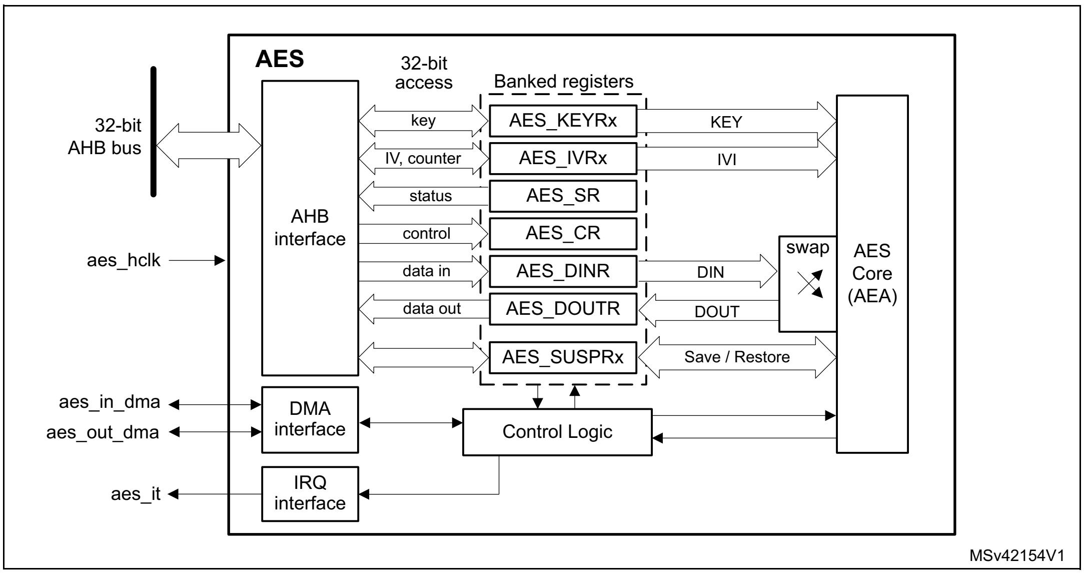
The devices have two AES peripherals, AES1 and AES2.

18.4 AES functional description

18.4.1 AES block diagram

Figure 49 shows the block diagram of AES.

Figure 49. AES block diagram



18.4.2 AES internal signals

Table 50 describes the user relevant internal signals interfacing the AES peripheral.

Table 50. AES internal input/output signals

Signal name	Signal type	Description
aes_hclk	Input	AHB bus clock
aes_it	Output	AES interrupt request
aes_in_dma	Input/Output	Input DMA single request/acknowledge
aes_out_dma	Input/Output	Output DMA single request/acknowledge

18.4.3 AES cryptographic core

Overview

The AES cryptographic core consists of the following components:

- AES core algorithm (AEA)
- multiplier over a binary Galois field (GF2mul)
- key input
- initialization vector (IV) input
- chaining algorithm logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks (four words) with 128-bit or 256-bit key length. Depending on the chaining mode, the AES requires zero or one 128-bit initialization vector IV.

The AES features the following modes of operation:

- **Mode 1:**
  Plaintext encryption using a key stored in the AES_KEYRx registers
- **Mode 2:**
  ECB or CBC decryption key preparation. It must be used prior to selecting Mode 3 with ECB or CBC chaining modes. The key prepared for decryption is stored automatically in the AES_KEYRx registers. Now the AES peripheral is ready to switch to Mode 3 for executing data decryption.
- **Mode 3:**
  Ciphertext decryption using a key stored in the AES_KEYRx registers. When ECB and CBC chaining modes are selected, the key must be prepared beforehand, through Mode 2.
- **Mode 4:**
  ECB or CBC ciphertext single decryption using the key stored in the AES_KEYRx registers (the initial key is derived automatically).

Note: Mode 2 and mode 4 are only used when performing ECB and CBC decryption.

When Mode 4 is selected only one decryption can be done, therefore usage of Mode 2 and Mode 3 is recommended instead.

The operating mode is selected by programming the MODE[1:0] bitfield of the AES_CR register. It may be done only when the AES peripheral is disabled.

Typical data processing

Typical usage of the AES is described in *Section 18.4.4: AES procedure to perform a cipher operation on page 341*.

Note: The outputs of the intermediate AEA stages are never revealed outside the cryptographic boundary, with the exclusion of the IVI bitfield.

Chaining modes

The following chaining modes are supported by AES, selected through the CHMOD[2:0] bitfield of the AES_CR register:

- Electronic code book (ECB)
- Cipher block chaining (CBC)
- Counter (CTR)
- Galois counter mode (GCM)
- Galois message authentication code (GMAC)
- Counter with CBC-MAC (CCM)

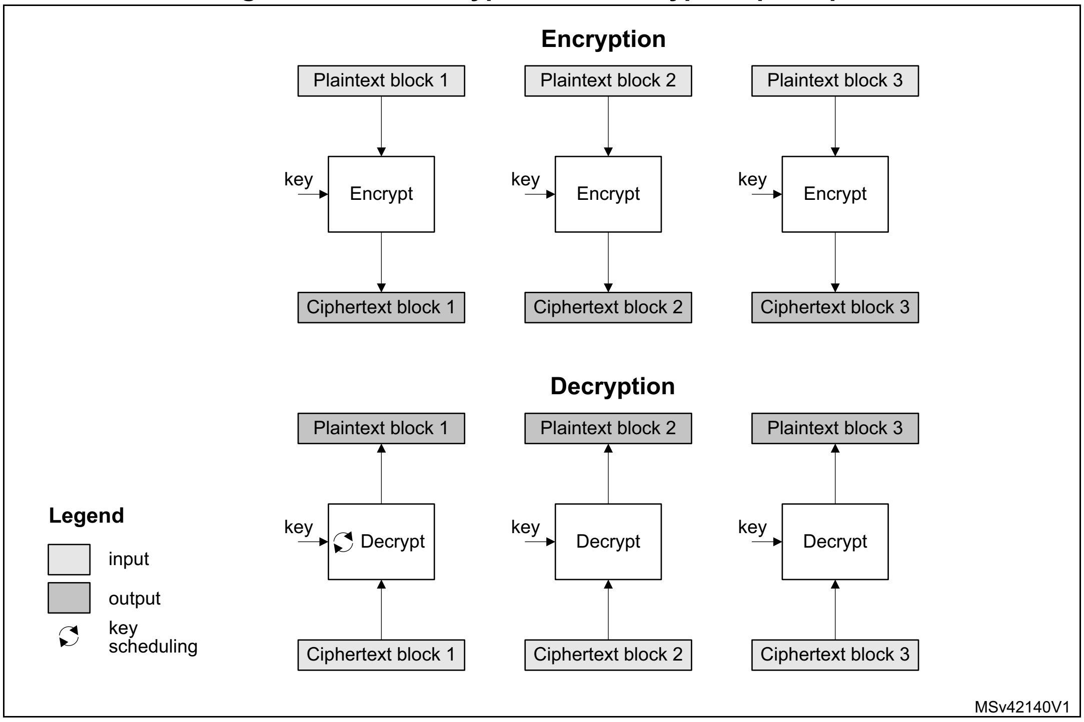
Note: The chaining mode may be changed only when AES is disabled (bit EN of the AES_CR register cleared).

Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from *Section 18.4.8: AES basic chaining modes (ECB, CBC)*.

Electronic codebook (ECB) mode

Figure 50. ECB encryption and decryption principle

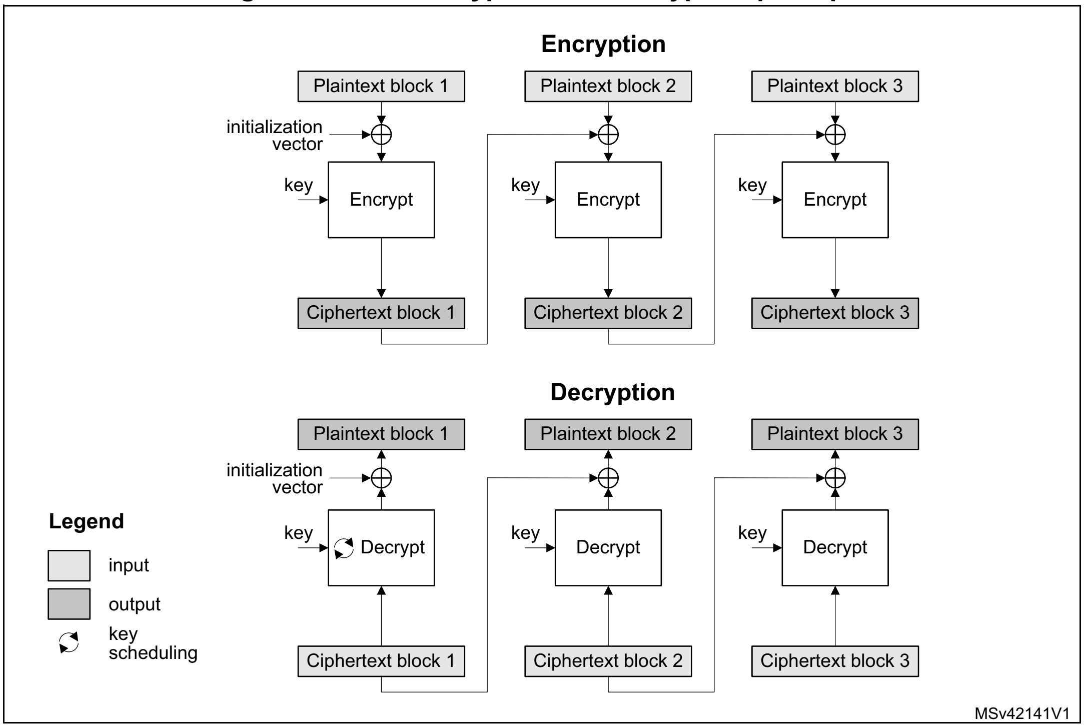


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.

Cipher block chaining (CBC) mode

Figure 51. CBC encryption and decryption principle

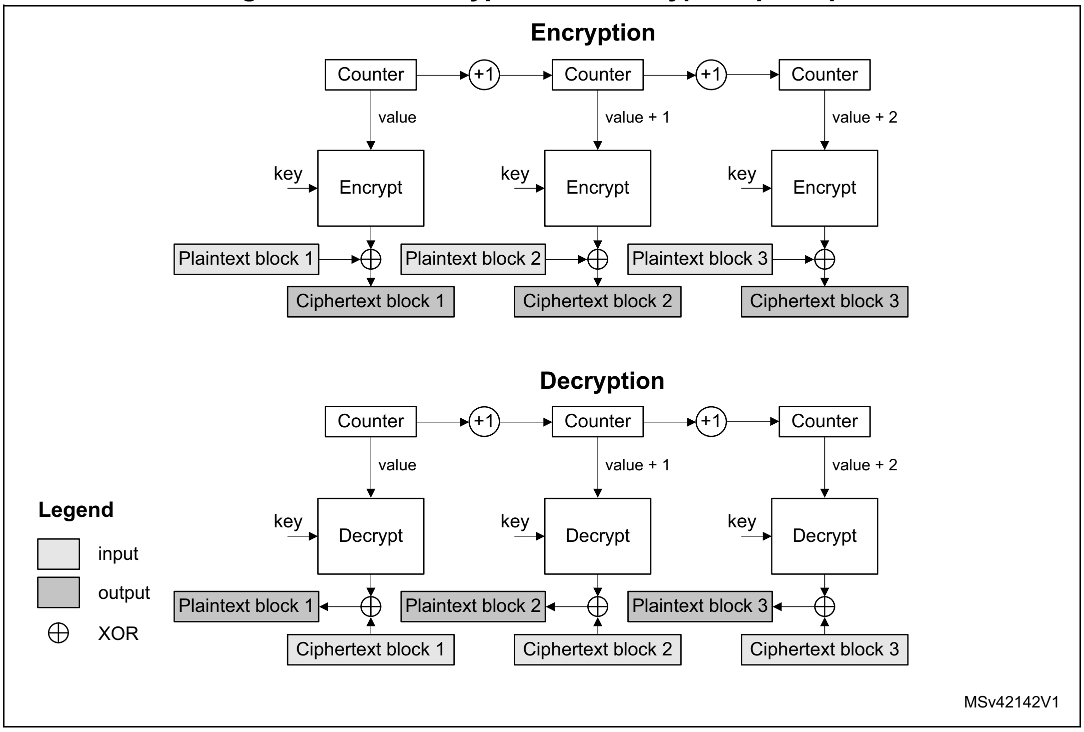


In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

Note: For decryption, a special key scheduling is required before processing the first block.

Counter (CTR) mode

Figure 52. CTR encryption and decryption principle

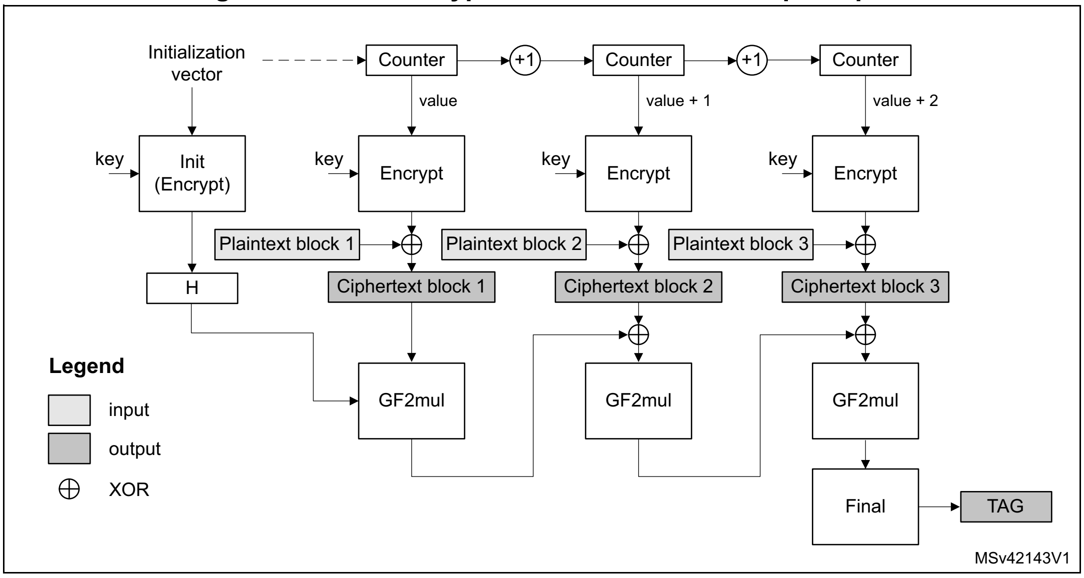


The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Note: Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption, since in this chaining scheme the AES core is always used in encryption mode for producing the key stream, or counter blocks.

Galois/counter mode (GCM)

Figure 53. GCM encryption and authentication principle

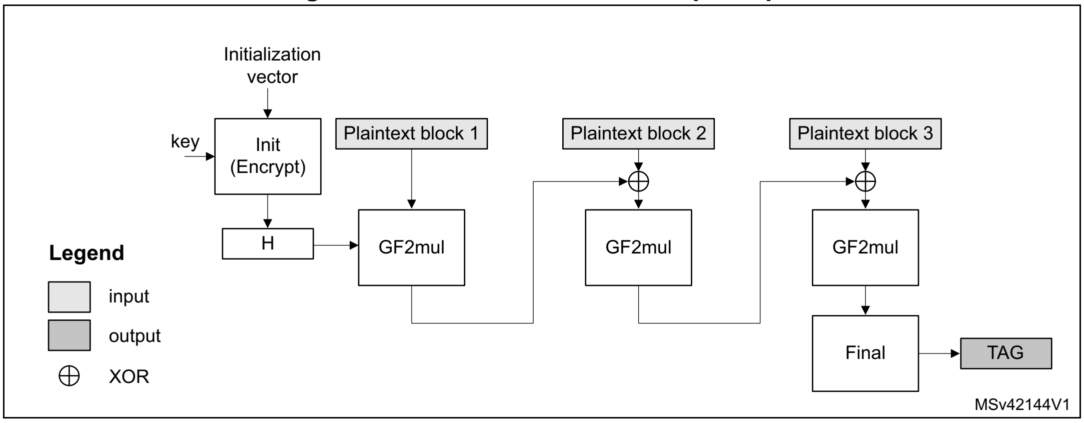


In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. It requires an initial value and a particular 128-bit block at the end of the message.

Galois message authentication code (GMAC) principle

Figure 54. GMAC authentication principle

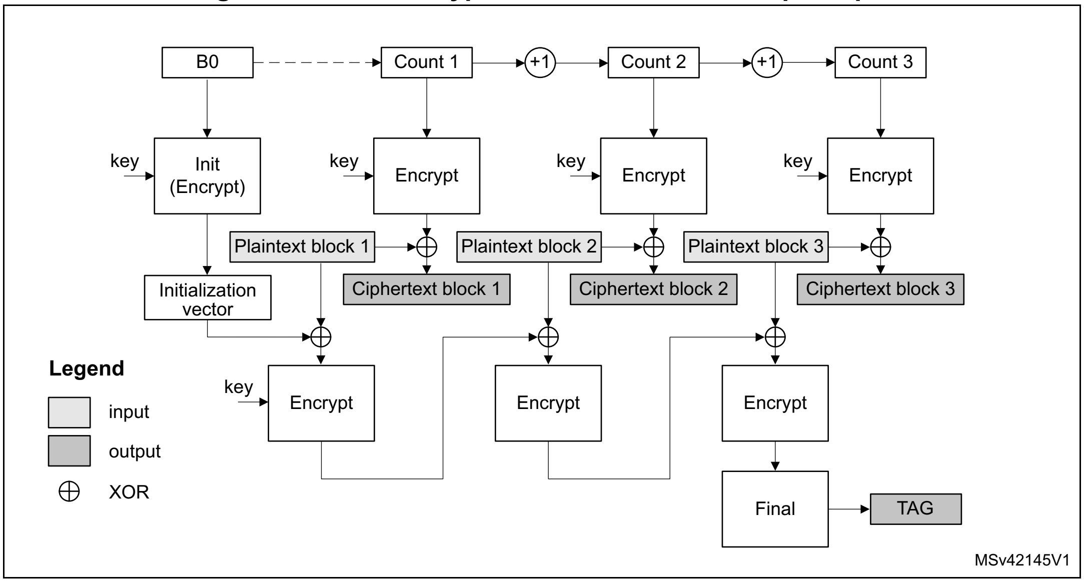


Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload).

Counter with CBC-MAC (CCM) principle

Figure 55. CCM encryption and authentication principle



In Counter with cipher block chaining-message authentication code (CCM) mode, the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and the corresponding MAC (also known as tag). It is described by NIST in *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

CCM mode is based on AES in counter mode for confidentiality and it uses CBC for computing the message authentication code. It requires an initial value.

Like GCM, the CCM chaining mode can be applied on a message composed only by plaintext authenticated data (that is, only header, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM/GMAC), and its use is not recommended by NIST.

18.4.4 AES procedure to perform a cipher operation

Introduction

A typical cipher operation is explained below. Detailed information is provided in sections starting from *Section 18.4.8: AES basic chaining modes (ECB, CBC)*.

Initialization of AES

To initialize AES, first disable it by clearing the EN bit of the AES_CR register. Then perform the following steps in any order:

- Configure the AES mode, by programming the MODE[1:0] bitfield of the AES_CR register.
  - For encryption, select Mode 1 (MODE[1:0] = 00).
  - For decryption, select Mode 3 (MODE[1:0] = 10), unless ECB or CBC chaining modes are used. In this latter case, perform an initial key derivation of the encryption key, as described in *Section 18.4.5: AES decryption round key preparation*.
- Select the chaining mode, by programming the CHMOD[2:0] bitfield of the AES_CR register.
- Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE[1:0] bitfield in the AES_CR register.
- When it is required (for example in CBC or CTR chaining modes), write the initialization vector into the AES_IVRx registers.
- Configure the key size (128-bit or 256-bit), with the KEYSIZE bitfield of the AES_CR register.
- Write a symmetric key into the AES_KEYRx registers (4 or 8 registers depending on the key size).

Data append

This section describes different ways of appending data for processing, where the size of data to process is not a multiple of 128 bits.

For ECB or CBC mode, refer to *Section 18.4.6: AES ciphertext stealing and data padding*. The last block management in these cases is more complex than in the sequence described in this section.

Data append through polling

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral by setting the EN bit of the AES_CR register.
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the AES_DINR register.
   b) Wait until the status flag CCF is set in the AES_SR, then read the four data words from the AES_DOUTR register.
   c) Clear the CCF flag, by setting the CCFC bit of the AES_CR register.
   d) If the data block just processed is the second-last block of the message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB bitfield of the AES_CR register, for AES to compute a correct tag;.
3. As it is the last block, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES_CR register.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the AES_DINR register, to allow sending the key to the AES processor.

NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.

Data append using interrupt

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES by setting the CCFIE bit of the AES_CR register.
2. Enable the AES peripheral by setting the EN bit of the AES_CR register.
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine, upon interrupt:
   a) Read four output data words from the AES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCFC bit of the AES_CR register.
   c) If the data block just processed is the second-last block of an message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB bitfield of the AES_CR register, for AES to compute a correct tag;. Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES_CR register and quit the interrupt service routine.
   e) Write next four input data words into the AES_DINR register and quit the interrupt service routine.

Note: AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations.

NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.

Data append using DMA

With this method, all the transfers and processing are managed by DMA and AES. To use the method, proceed as follows:

1. Prepare the last four-word data block (if the data to process does not fill it completely), by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in *Section 18.4.16: AES DMA interface*. Configure the DMA controller so as to generate an interrupt on transfer completion. In case of GCM payload encryption or CCM payload decryption, DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Data append through polling* must be used instead for this last block, because NPBLB bits must be setup before processing the block, for AES to compute a correct tag.
3. Enable the AES peripheral by setting the EN bit of the AES_CR register
4. Enable DMA requests by setting the DMAINEN and DMAOUTEN bits of the AES_CR register.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

Note: The CCF flag has no use with this method, because the reading of the AES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

NPBLB bits are not used in header phase of GCM, GMAC, and CCM chaining modes.

18.4.5 AES decryption round key preparation

Internal key schedule is used to generate AES round keys. In AES encryption, the round 0 key is the one stored in the key registers. AES decryption must start using the last round key. As the encryption key is stored in memory, a special key scheduling must be performed to obtain the decryption key. This key scheduling is only required for AES decryption in ECB and CBC modes.

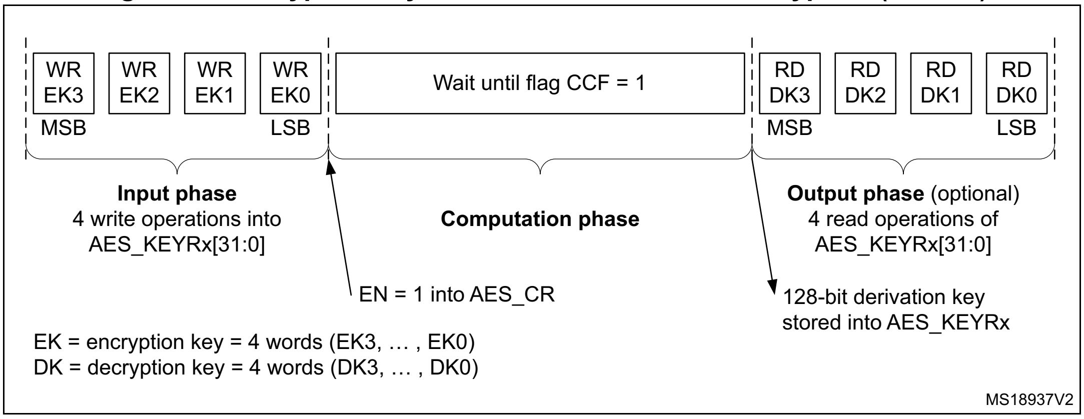
Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES_CR (key process only), then proceed with the decryption by setting MODE[1:0] to 10 (Mode 3, decryption only). Mode 2 usage is described below:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES_CR. The CHMOD[2:0] bitfield is not significant in this case because this key derivation mode is independent of the chaining algorithm selected.
3. Set key length to 128 or 256 bits, via KEYSIZE bit of AES_CR register.
4. Write the AES_KEYRx registers (128 or 256 bits) with encryption key, as shown in *Figure 56*. Writes to the AES_IVRx registers have no effect.
5. Enable the AES peripheral, by setting the EN bit of the AES_CR register.
6. Wait until the CCF flag is set in the AES_SR register.
7. Clear the CCF flag. Derived key is available in AES core, ready to use for decryption. Application can also read the AES_KEYRx register to obtain the derived key if needed, as shown in *Figure 56* (the processed key is loaded automatically into the AES_KEYRx registers).

Note: The AES is disabled by hardware when the derivation key is available.

To restart a derivation key computation, repeat steps 4, 5, 6, and 7.

Figure 56. Encryption key derivation for ECB/CBC decryption (Mode 2)



If the software stores the initial key prepared for decryption, it is enough to do the key schedule operation only once for all the data to be decrypted with a given cipher key.

Note: The operation of the key preparation lasts 59 or 82 clock cycles, depending on the key size (128- or 256-bit).

18.4.6 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (128 bits), ciphertext stealing techniques are used, such as those described in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since the AES peripheral does not support such techniques, the application must complete the last block of input data using data from the second last block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, when AES is used in other modes than ECB or CBC, an incomplete input data block (that is, block with input data shorter than 128 bits) must be padded with zeros prior to encryption (that is, extra bits must be appended to the trailing end of the data string). After decryption, the extra bits must be discarded. As AES does not implement automatic data padding operation to **the last block**, the application must follow the recommendation given in *Section 18.4.4: AES procedure to perform a cipher operation on page 341* to manage messages the size of which is not a multiple of 128 bits.

Note: Padding data are swapped in a similar way as normal data, according to the DATATYPE[1:0] field of the AES_CR register (see Section 18.4.13: AES data registers and data swapping for details).

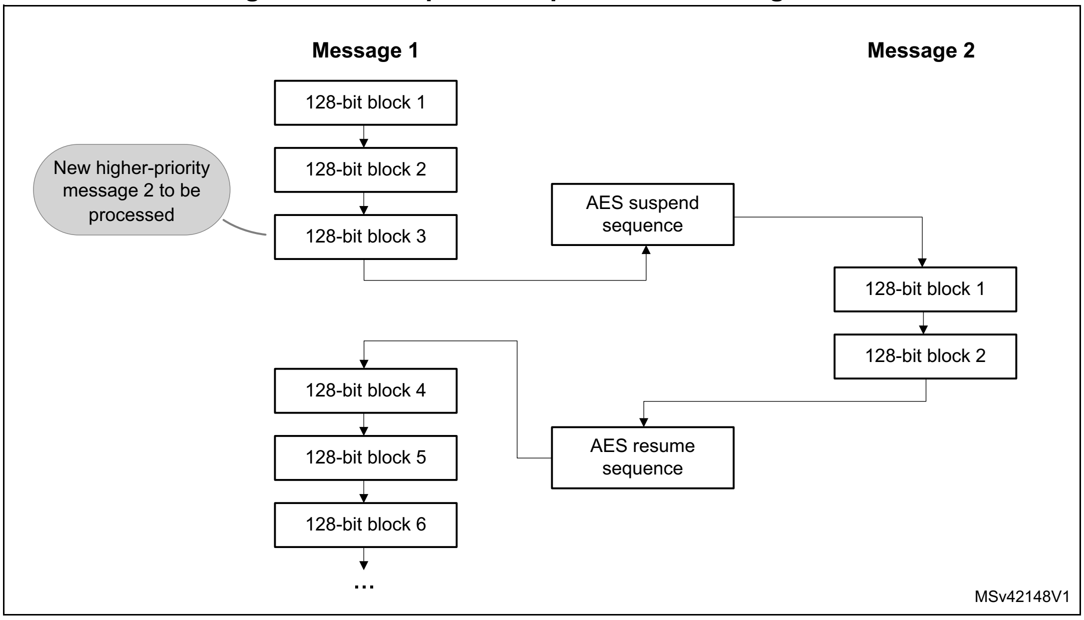
18.4.7 AES task suspend and resume

A message can be suspended if another message with a higher priority must be processed. When this highest priority message is sent, the suspended message can resume in both encryption or decryption mode.

Suspend/resume operations do not break the chaining operation and the message processing can resume as soon as AES is enabled again to receive the next data block.

Figure 57 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 57. Example of suspend mode management



A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

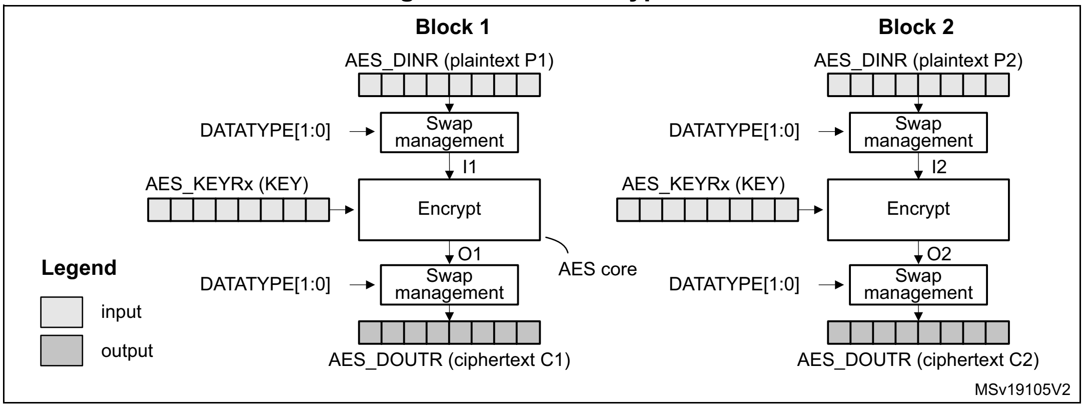
18.4.8 AES basic chaining modes (ECB, CBC)

Overview

This section gives a brief explanation of the four basic operation modes provided by the AES core: ECB encryption, ECB decryption, CBC encryption and CBC decryption. For detailed information, refer to the FIPS publication 197 from November 26, 2001.

Figure 58 illustrates the electronic codebook (ECB) encryption.

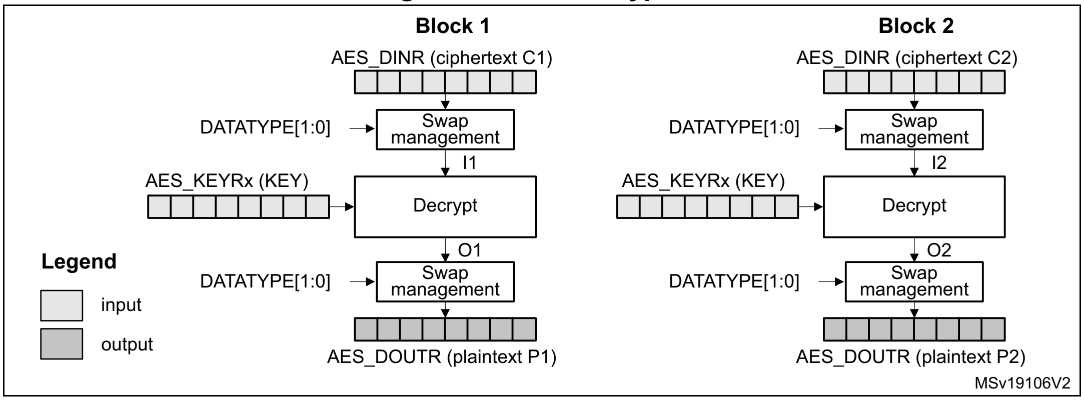
Figure 58. ECB encryption



In ECB encrypt mode, the 128-bit plaintext input data block Px in the AES_DINR register first goes through bit/byte/half-word swapping. The swap result Ix is processed with the AES core set in encrypt mode, using a 128- or 256-bit key. The encryption result Ox goes through bit/byte/half-word swapping, then is stored in the AES_DOUTR register as 128-bit ciphertext output data block Cx. The ECB encryption continues in this way until the last complete plaintext block is encrypted.

Figure 59 illustrates the electronic codebook (ECB) decryption.

Figure 59. ECB decryption

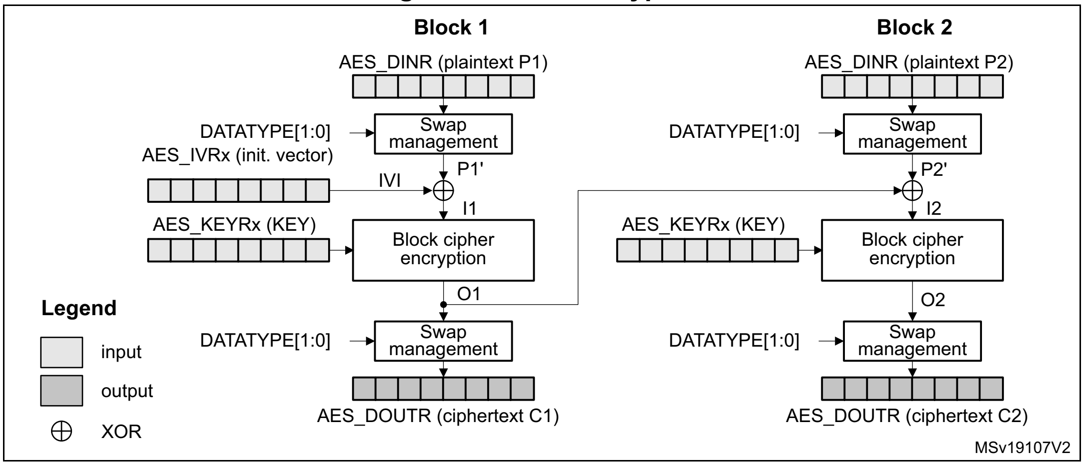


To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported by the AES core.

In ECB decrypt mode, the 128-bit ciphertext input data block C1 in the AES_DINR register first goes through bit/byte/half-word swapping. The keying sequence is reversed compared to that of the ECB encryption. The swap result I1 is processed with the AES core set in decrypt mode, using the formerly prepared decryption key. The decryption result goes through bit/byte/half-word swapping, then is stored in the AES_DOUTR register as 128-bit plaintext output data block P1. The ECB decryption continues in this way until the last complete ciphertext block is decrypted.

Figure 60 illustrates the cipher block chaining (CBC) encryption.

Figure 60. CBC encryption

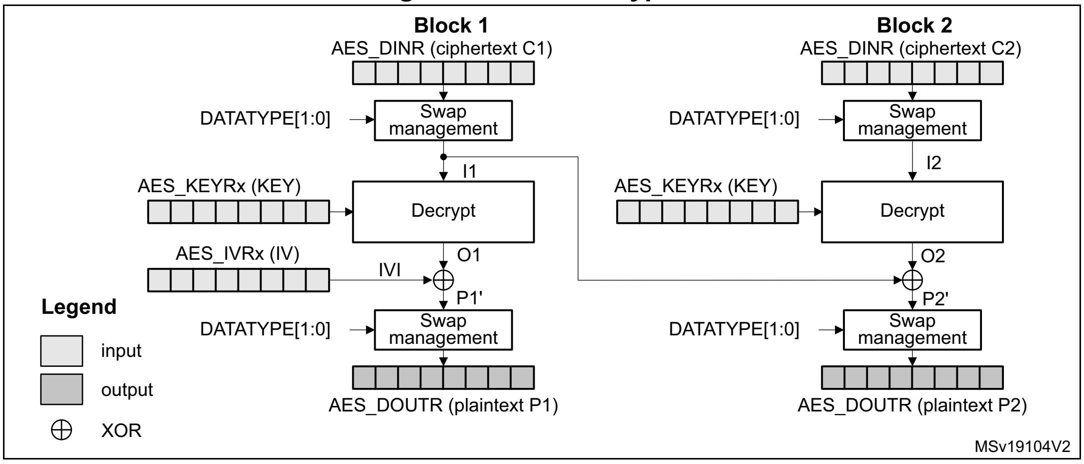


In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after swapping operation, is used as ciphertext C1. The O1 data is then XOR-ed with the second-block plaintext data P2' to produce the I2 input data for the AES core to produce the second block of ciphertext data. The chaining of data blocks continues in this way until the last plaintext block in the message is encrypted.

If the message size is not a multiple of 128 bits, the final partial data block is encrypted in the way explained in *Section 18.4.6: AES ciphertext stealing and data padding*.

Figure 61 illustrates the cipher block chaining (CBC) decryption.

Figure 61. CBC decryption



In CBC decrypt mode, like in ECB decrypt mode, the secret key must be prepared to perform an AES decryption.

After the key preparation process, the decryption goes as follows: the first 128-bit ciphertext block (after the swap operation) is used directly as the AES core input block I1 for decrypt operation, using the 128-bit or 256-bit key. Its output O1 is XOR-ed with the 128-bit IVI field (that must be identical to that used during encryption) to produce the first plaintext block P1.

The second ciphertext block is processed in the same way as the first block, except that the I1 data from the first block is used in place of the initialization vector.

The decryption continues in this way until the last complete ciphertext block is decrypted.

If the message size is not a multiple of 128 bits, the final partial data block is decrypted in the way explained in *Section 18.4.6: AES ciphertext stealing and data padding*.

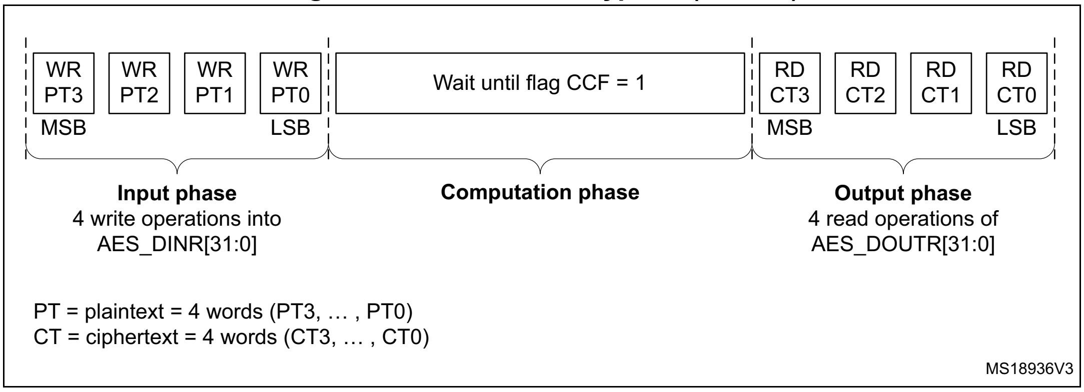
For more information on data swapping, refer to *Section 18.4.13: AES data registers and data swapping*.

ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in *Section 18.4.4*):

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select the Mode 1 by setting to 00 the MODE[1:0] bitfield of the AES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES_CR register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield.
3. Select 128- or 256-bit key length through the KEYSIZE bit of the AES_CR register.
4. Write the AES_KEYRx registers (128 or 256 bits) with encryption key. Fill the AES_IVRx registers with the initialization vector data if CBC mode has been selected.
5. Enable the AES peripheral by setting the EN bit of the AES_CR register.
6. Write the AES_DINR register four times to input the plaintext (MSB first), as shown in *Figure 62*.
7. Wait until the CCF flag is set in the AES_SR register.
8. Read the AES_DOUTR register four times to get the ciphertext (MSB first) as shown in *Figure 62*. Then clear the CCF flag by setting the CCFC bit of the AES_CR register.
9. Repeat steps *6*-*7*-*8* to process all the blocks with the same encryption key.

Figure 62. ECB/CBC encryption (Mode 1)

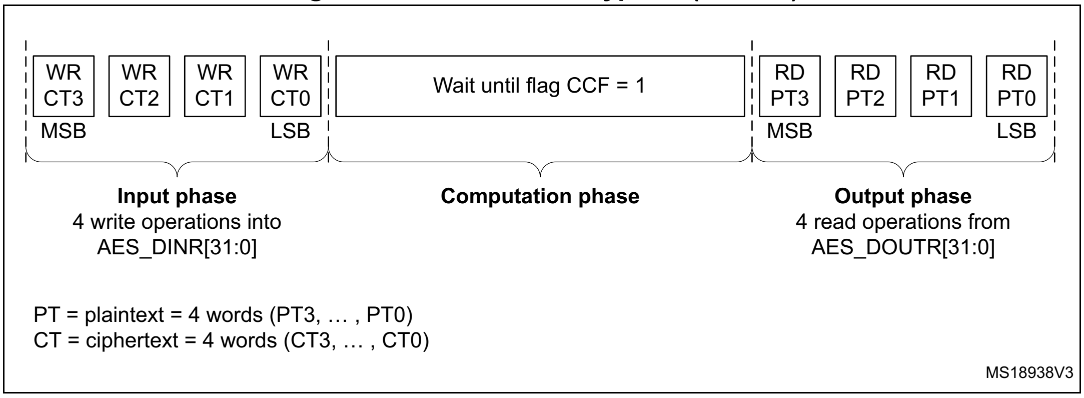


ECB/CBC decryption sequence

The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in *Section 18.4.4*).

1. Follow the steps described in *Section 18.4.5: AES decryption round key preparation*, in order to prepare the decryption key in AES core.
2. Select the Mode 3 by setting to 10 the MODE[1:0] bitfield of the AES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES_CR register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield. KEYSIZE bitfield must be kept as-is.
3. Write the AES_IVRx registers with the initialization vector (required in CBC mode only).
4. Enable AES by setting the EN bit of the AES_CR register.
5. Write the AES_DINR register four times to input the cipher text (MSB first), as shown in *Figure 63*.
6. Wait until the CCF flag is set in the AES_SR register.
7. Read the AES_DOUTR register four times to get the plain text (MSB first), as shown in *Figure 63*. Then clear the CCF flag by setting the CCFC bit of the AES_CR register.
8. Repeat steps *5*-*6*-*7* to process all the blocks encrypted with the same key.

Figure 63. ECB/CBC decryption (Mode 3)



Suspend/resume operations in ECB/CBC modes

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register.
2. If DMA is not used, read four times the AES_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.
3. If DMA is not used, poll the CCF flag of the AES_SR register until it becomes 1 (computation completed).
4. Clear the CCF flag by setting the CCFC bit of the AES_CR register.
5. Save initialization vector registers (only required in CBC mode as AES_IVRx registers are altered during the data processing).

6. Disable the AES peripheral by clearing the bit EN of the AES_CR register.
7. Save the AES_CR register and clear the key registers if they are not needed, to process the higher priority message.
8. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, and so on).

Note: In point 7, the derived key information stored in AES_KEYRx registers can optionally be saved in memory if the interrupted process is a decryption. Otherwise those registers do not need to be saved as the original key value is known by the application

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Restore AES_CR register (with correct KEYSIZE) then restore AES_KEYRx registers. In case of decryption, derived key information can be written in AES_KEYRx register instead of the original key value.
4. Prepare the decryption key as described in *Section 18.4.5: AES decryption round key preparation* (only required for ECB or CBC decryption). This step is not necessary if derived key information is loaded in AES_KEYRx registers.
5. Restore AES_IVRx registers using the saved configuration (only required in CBC mode).
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.
7. If DMA is used, enable AES DMA transfers by setting the DMAINEN and DMAOUTEN bits of the AES_CR register.

Alternative single ECB/CBC decryption using Mode 4

The sequence of events to perform a single round of ECB/CBC decryption using Mode 4 is:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select the Mode 4 by setting to 11 the MODE[1:0] bitfield of the AES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES_CR register to 0x0 or 0x1, respectively.
3. Select key length of 128 or 256 bits via KEYSIZE bitfield of the AES_CR register.
4. Write the AES_KEYRx registers with the encryption key. Write the AES_IVRx registers if the CBC mode is selected.
5. Enable the AES peripheral by setting the EN bit of the AES_CR register.
6. Write the AES_DINR register four times to input the cipher text (MSB first).
7. Wait until the CCF flag is set in the AES_SR register.
8. Read the AES_DOUTR register four times to get the plain text (MSB first). Then clear the CCF flag by setting the CCFC bit of the AES_CR register.

Note: When mode 4 is selected mode 3 cannot be used.

In mode 4, the AES_KEYRx registers contain the encryption key during all phases of the processing. No derivation key is stored in these registers. It is stored internally in AES.

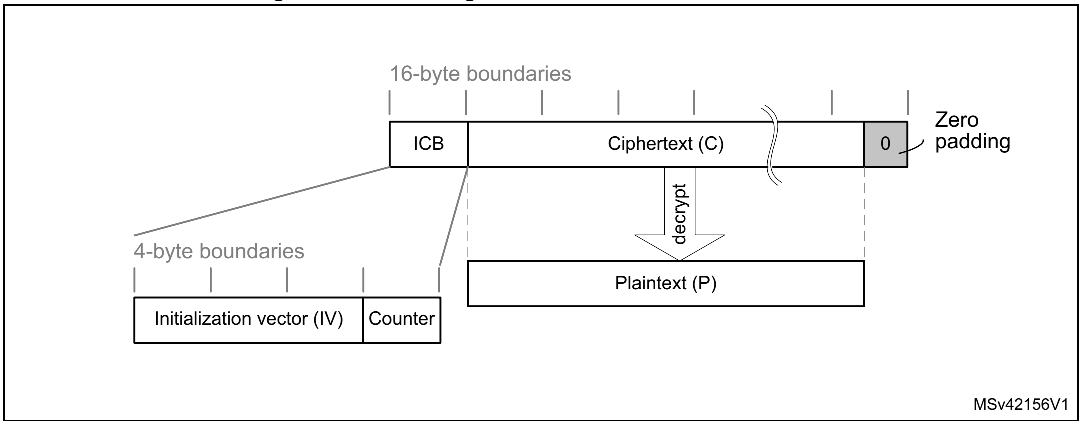
18.4.9 AES counter (CTR) mode

Overview

The counter mode (CTR) uses AES as a key-stream generator. The generated keys are then XOR-ed with the plaintext to obtain the ciphertext.

CTR chaining is defined in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.* A typical message construction in CTR mode is given in *Figure 64*.

Figure 64. Message construction in CTR mode



The structure of this message is:

- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

CTR encryption and decryption

Figure 65 and *Figure 66* describe the CTR encryption and decryption process, respectively, as implemented in the AES peripheral. The CTR mode is selected by writing 010 to the CHMOD[2:0] bitfield of AES_CR register.

Figure 65. CTR encryption

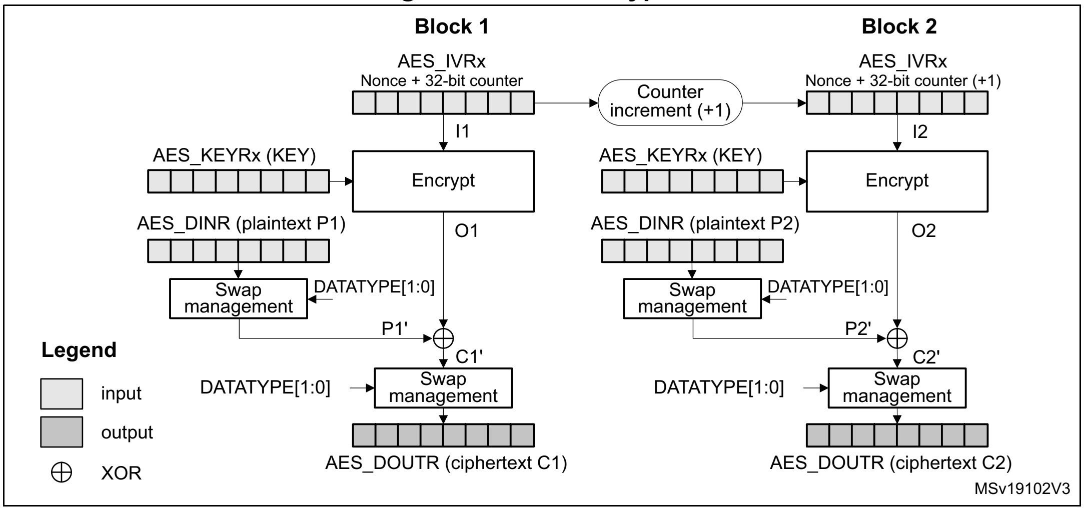
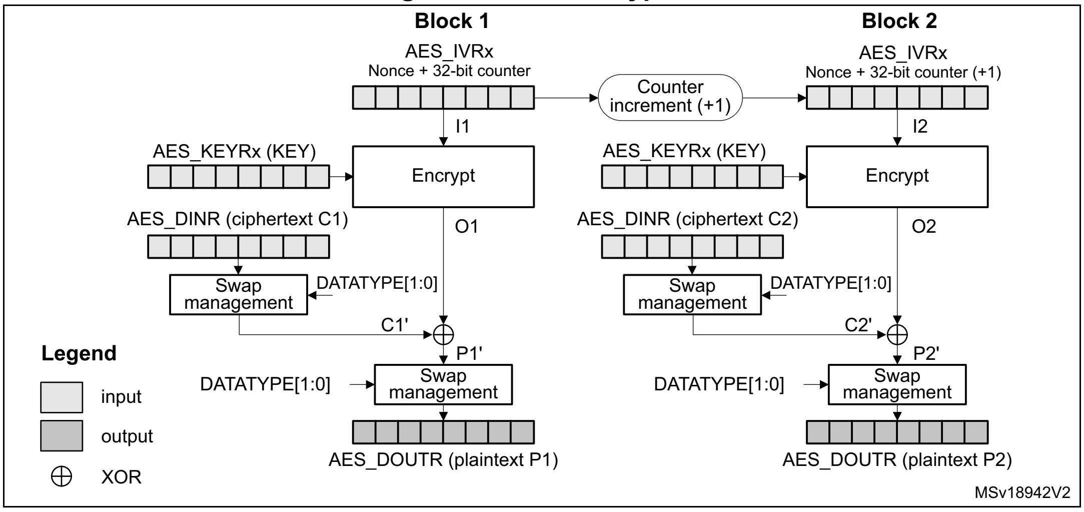


Figure 66. CTR decryption



In CTR mode, the cryptographic core output (also called keystream) Ox is XOR-ed with relevant input block (Px' for encryption, Cx' for decryption), to produce the correct output block (Cx' for encryption, Px' for decryption). Initialization vectors in AES must be initialized as shown in *Table 51*.

Table 51. CTR mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0} 32-bit counter = 0x0001

Unlike in CBC mode that uses the AES_IVRx registers only once when processing the first data block, in CTR mode AES_IVRx registers are used for processing each data block, and the AES peripheral increments the counter bits of the initialization vector (leaving the nonce bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that is then XOR-ed with the plaintext (CTR encryption) or ciphertext (CTR decryption) input. In CTR mode, the MODE[1:0] bitfield setting 01 (key derivation) is forbidden and all the other settings default to encryption mode.

The sequence of events to perform an encryption or a decryption in CTR chaining mode:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select CTR chaining mode by setting to 010 the CHMOD[2:0] bitfield of the AES_CR register. Set MODE[1:0] bitfield to any value other than 01.
3. Initialize the AES_KEYRx registers, and load the AES_IVRx registers as described in *Table 51*.
4. Set the EN bit of the AES_CR register, to start encrypting the current counter (EN is automatically reset when the calculation finishes).
5. If it is the last block, pad the data with zeros to have a complete block, if needed.
6. Append data in AES, and read the result. The three possible scenarios are described in *Section 18.4.4: AES procedure to perform a cipher operation*.
7. Repeat the previous step till the second-last block is processed. For the last block, apply the two previous steps and discard the bits that are not part of the payload (if the size of the significant data in the last input block is less than 16 bytes).

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, and resume the message that was interrupted. Detailed CBC suspend/resume sequence is described in *Section 18.4.8: AES basic chaining modes (ECB, CBC)*.

Note: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.

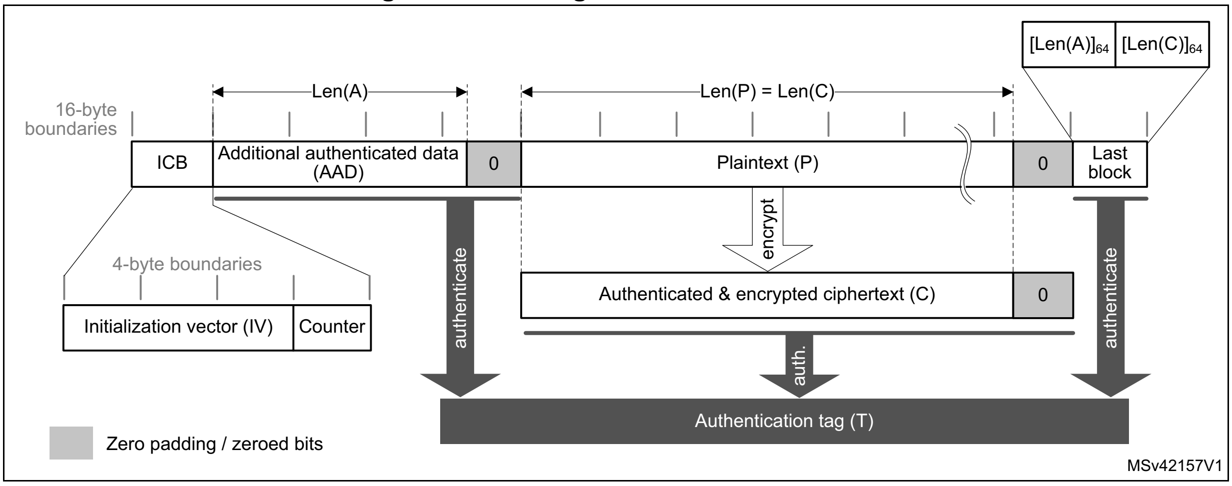
18.4.10 AES Galois/counter mode (GCM)

Overview

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, GCM algorithm is based on AES counter mode. It uses a multiplier over a fixed finite field to generate the tag.

GCM chaining is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.* A typical message construction in GCM mode is given in *Figure 67*.

Figure 67. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. Note that the GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} - 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed 2^{32} - 2 128-bit blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 52*.

The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

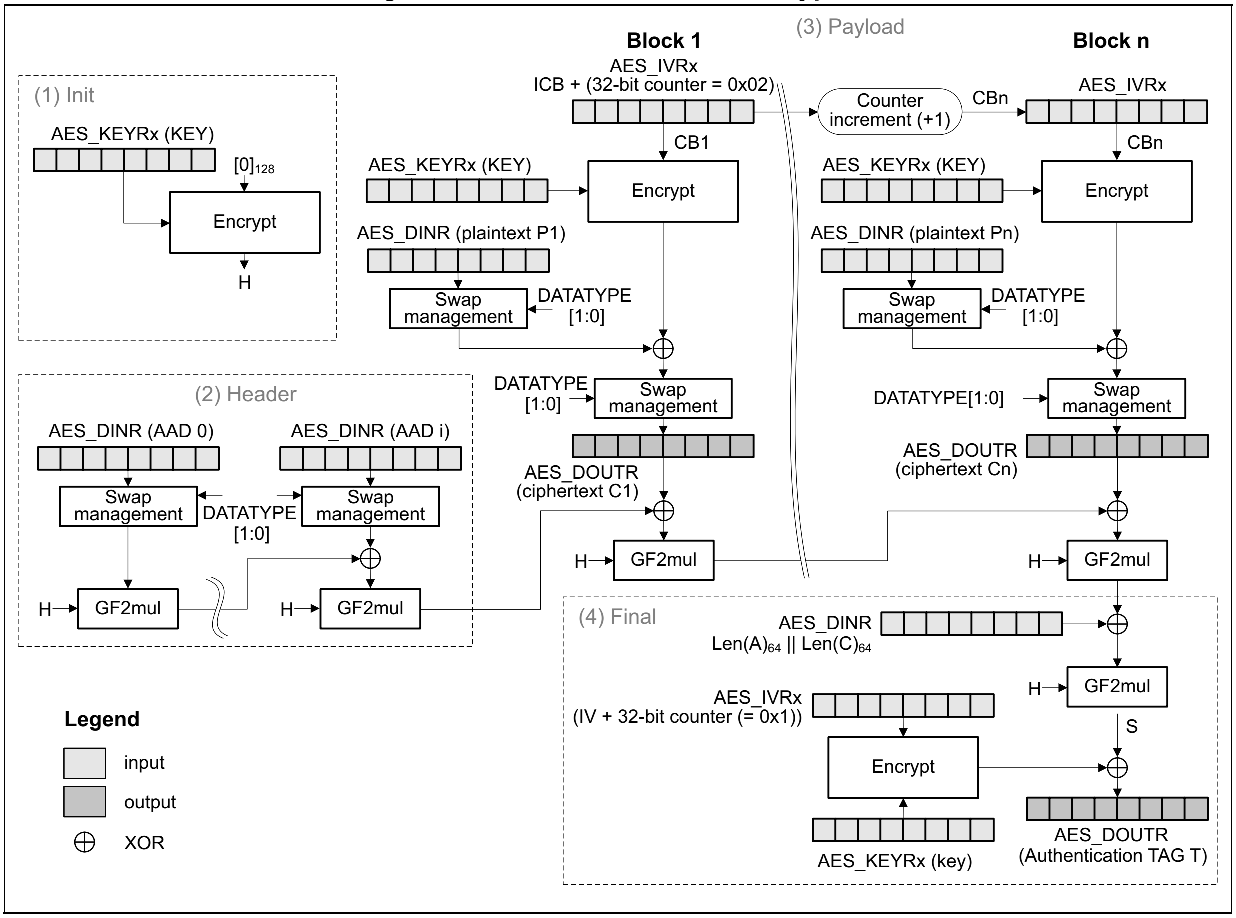
Table 52. GCM last block definition

Endianness	Bit[0] ---------- Bit[31]	Bit[32]---------- Bit[63]	Bit[64] -------- Bit[95]	Bit[96] --------- Bit[127]
Input data	0x0	AAD length[31:0]	0x0	Payload length[31:0]

GCM processing

Figure 68 describes the GCM implementation in the AES peripheral. The GCM is selected by writing 011 to the CHMOD[2:0] bitfield of the AES_CR register.

Figure 68. GCM authenticated encryption



The mechanism for the confidentiality of the plaintext in GCM mode is similar to that in the Counter mode, with a particular increment function (denoted 32-bit increment) that generates the sequence of input counter blocks.

AES_IVRx registers keeping the **counter block** of data are used for processing each data block. The AES peripheral automatically increments the Counter[31:0] bitfield. The first counter block (CB1) is derived from the initial counter block ICB by the application software (see *Table 53*).

Table 53. Initialization of AES_IVRx registers in GCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

Note: In this mode, the settings 01 and 11 of the MODE[1:0] bitfield are forbidden.

The authentication mechanism in GCM mode is based on a hash function called **GF2mul** that performs multiplication by a fixed parameter, called hash subkey (H), within a binary Galois field.

A GCM message is processed through the following phases, further described in next subsections:

- **Init phase**: AES prepares the GCM hash subkey (H).
- **Header phase**: AES processes the additional authenticated data (AAD), with hash computation only.
- **Payload phase**: AES processes the plaintext (P) with hash computation, counter block encryption and data XOR-ing. It operates in a similar way for ciphertext (C).
- **Final phase**: AES generates the authenticated tag (T) using the last block of the message.

GCM init phase

During this first step, the GCM hash subkey (H) is calculated and saved internally, to be used for processing all the blocks. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select GCM chaining mode, by setting to 011 the CHMOD[2:0] bitfield of the AES_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES_CR register.
4. Set the MODE[1:0] bitfield of the AES_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES_KEYRx registers with a key, and initialize AES_IVRx registers with the information as defined in *Table 53*.
6. Start the calculation of the hash key, by setting to 1 the EN bit of the AES_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag of the AES_SR register, by setting the CCFC bit of the AES_CR register.

GCM header phase

This phase coming after the GCM Init phase must be completed before the payload phase. The sequence to execute, identical for encryption and decryption, is:

1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. Enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways described in *Section 18.4.4: AES procedure to perform a cipher operation*. No data is read during this phase.
4. Repeat the step *3* until the last additional authenticated data block is processed.

Note: The header phase can be skipped if there is no AAD, that is, Len(A) = 0.

GCM payload phase

This phase, identical for encryption and decryption, is executed after the GCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES_DOUTR register. The sequence to execute is:

1. Indicate the payload phase, by setting to 10 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in *Section 18.4.4: AES procedure to perform a cipher operation on page 341*, and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block, discard the bits that are not part of the payload when the last block size is less than 16 bytes.

Note: The payload phase can be skipped if there is no payload data, that is, Len(C) = 0 (see GMAC mode).

GCM final phase

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCMPH[1:0] bitfield of the AES_CR register.
2. Compose the data of the block, by concatenating the AAD bit length and the payload bit length, as shown in *Table 52*. Write the block into the AES_DINR register.
3. Wait until the end of computation, indicated by the CCF flag of the AES_SR transiting to 1.
4. Get the GCM authentication tag, by reading the AES_DOUTR register four times.
5. Clear the CCF flag of the AES_SR register, by setting the CCFC bit of the AES_CR register.
6. Disable the AES peripheral, by clearing the bit EN of the AES_CR register. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data is written to AES_DINR normally (no swapping), while swapping is applied to tag data read from AES_DOUTR.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES_SR register set to 1.
2. In the payload phase, if DMA is not used, read four times the AES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.
3. Clear the CCF flag of the AES_SR register, by setting the CCFC bit of the AES_CR register.
4. Save the AES_SUSPxR registers in the memory, where x is from 0 to 7.
5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.
7. Save the current AES configuration in the memory, excluding the initialization vector registers AES_IVRx. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPxR registers, where x is from 0 to 7.
4. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
5. Restore the initial setting values in the AES_CR and AES_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.

If DMA is used, enable AES DMA requests by setting the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES_CR register.

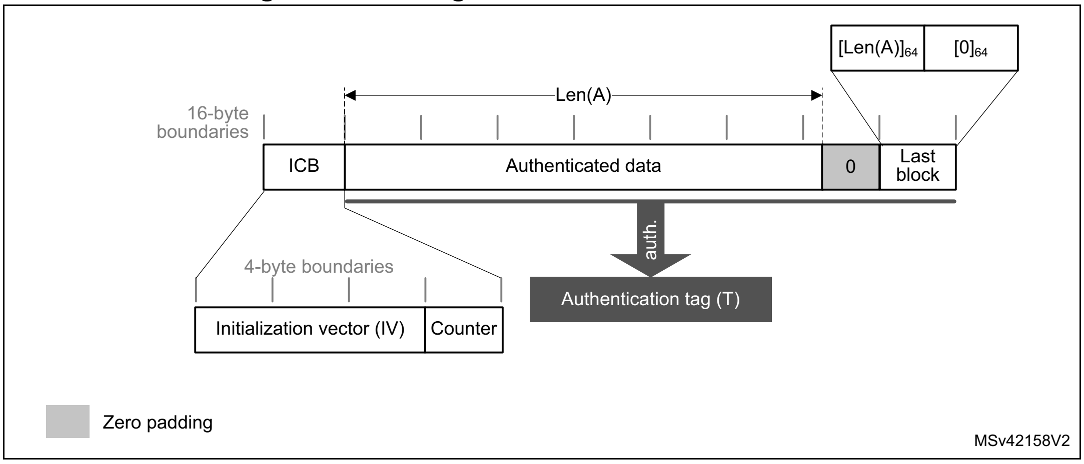
18.4.11 AES Galois message authentication code (GMAC)

Overview

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code). It is based on GCM algorithm, as defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

A typical message construction for GMAC is given in *Figure 69*.

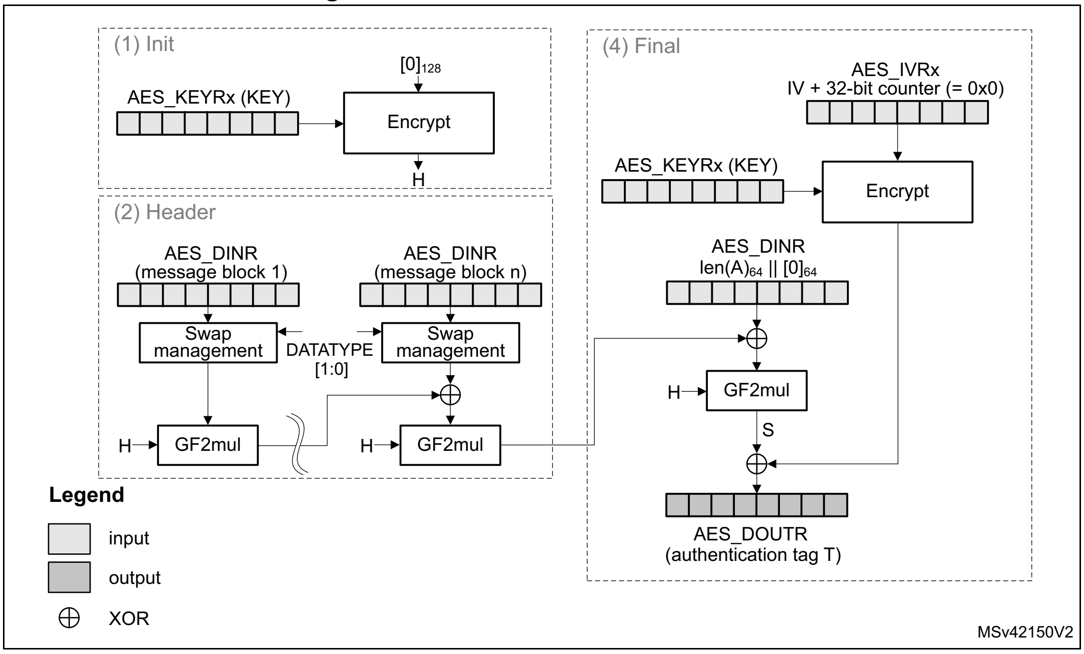
Figure 69. Message construction in GMAC mode



AES GMAC processing

Figure 70 describes the GMAC mode implementation in the AES peripheral. This mode is selected by writing 011 to the CHMOD[2:0] bitfield of the AES_CR register.

Figure 70. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message only containing a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

18.4.12 AES counter with CBC-MAC (CCM)

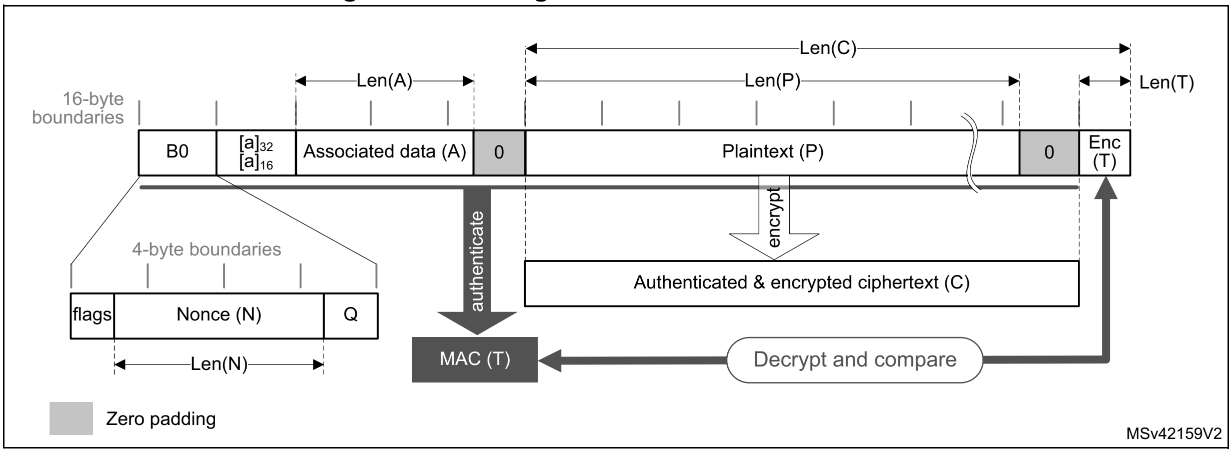
Overview

The AES **counter with cipher block chaining-message authentication code** (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES in counter mode. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve this CBC-MAC as an authentication mode outside the context of the CCM specification.

CCM chaining is specified in NIST *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*. A typical message construction for CCM is given in *Figure 71*.

Figure 71. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the Associated Data (A).
  This part of the message is only authenticated, not encrypted. This section has a

known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 71*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:

- If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
- If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
- If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 71*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

CTR0: 0710111213 141516 00000000 00000000

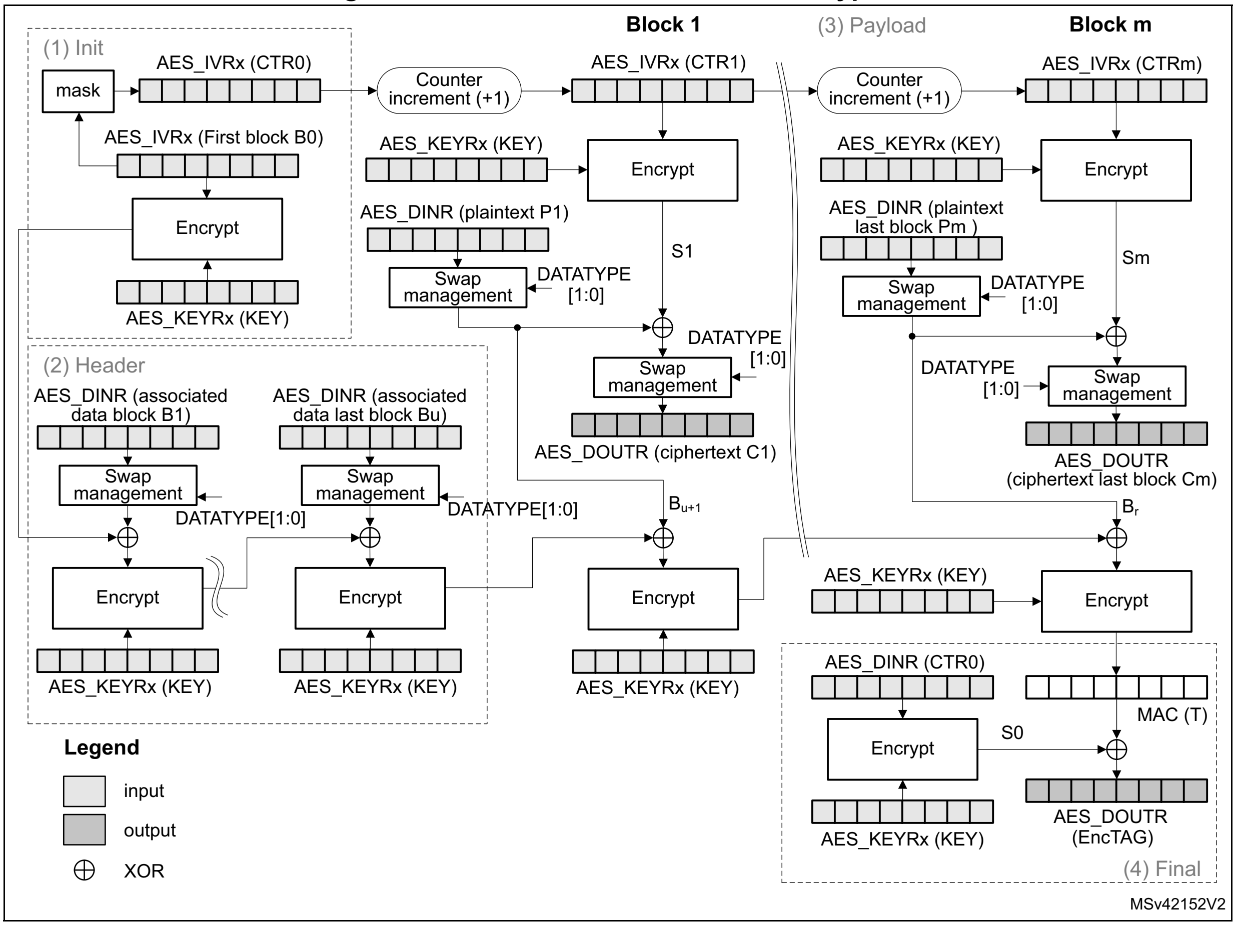
CTR1: 0710111213 141516 00000000 00000001

Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.

CCM processing

Figure 72 describes the CCM implementation within the AES peripheral (encryption example). This mode is selected by writing 100 into the CHMOD[2:0] bitfield of the AES_CR register.

Figure 72. CCM mode authenticated encryption



The data input to the generation-encryption process are a valid nonce, a valid payload string, and a valid associated data string, all properly formatted. The CBC chaining mechanism is applied to the formatted plaintext data to generate a MAC, with a known length. Counter mode encryption that requires a sufficiently long sequence of counter blocks as input, is applied to the payload string and separately to the MAC. The resulting ciphertext C is the output of the generation-encryption process on plaintext P.

AES_IVRx registers are used for processing each data block, AES automatically incrementing the CTR counter with a bit length defined by the first block B0. *Table 54* shows how the application must load the B0 data.

Note: The AES peripheral in CCM mode supports counters up to 64 bits, as specified by NIST.

Table 54. Initialization of AES_IVRx registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96]	B0[95:64]	B0[63:32]	B0[31:0]

Note: In this mode, the settings 01 and 11 of the MODE[1:0] bitfield are forbidden.

A CCM message is processed through the following phases, further described in next subsections:

- **Init phase**: AES processes the first block and prepares the first counter block.
- **Header phase**: AES processes associated data (A), with tag computation only.
- **Payload phase**: IP processes plaintext (P), with tag computation, counter block encryption, and data XOR-ing. It works in a similar way for ciphertext (C).
- **Final phase**: AES generates the message authentication code (MAC).

CCM Init phase

In this phase, the first block B0 of the CCM message is written into the AES_IVRx register. The AES_DOUTR register does not contain any output data. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select CCM chaining mode, by setting to 100 the CHMOD[2:0] bitfield of the AES_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES_CR register.
4. Set the MODE[1:0] bitfield of the AES_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES_KEYRx registers with a key, and initialize AES_IVRx registers with B0 data as described in *Table 54*.
6. Start the calculation of the counter, by setting to 1 the EN bit of the AES_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag in the AES_SR register, by setting to 1 the CCFC bit of the AES_CR register.

CCM header phase

This phase coming after the GCM Init phase must be completed before the payload phase. During this phase, the AES_DOUTR register does not contain any output data.

The sequence to execute, identical for encryption and decryption, is:

1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. Enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways described in *Section 18.4.4: AES procedure to perform a cipher operation*. No data is read during this phase.
4. Repeat the step *3* until the last additional authenticated data block is processed.

Note: The header phase can be skipped if there is no associated data, that is, Len(A) = 0.

The first block of the associated data (B1) must be formatted by software, with the associated data length.

CCM payload phase (encryption or decryption)

This phase, identical for encryption and decryption, is executed after the CCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES_DOUTR register. The sequence to execute is:

1. Indicate the payload phase, by setting to 10 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last data block to encrypt and the plaintext size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in *Section 18.4.4: AES procedure to perform a cipher operation on page 341*, and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), apply the two previous steps. For the last block, discard the data that is not part of the payload when the last block size is less than 16 bytes.

Note: The payload phase can be skipped if there is no payload data, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}$(C) encrypted tag information when decrypting ciphertext C.

CCM final phase

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCMPH[1:0] bitfield of the AES_CR register.
2. Wait until the end-of-computation flag CCF of the AES_SR register is set.
3. Read four times the AES_DOUTR register: the output corresponds to the CCM authentication tag.
4. Clear the CCF flag of the AES_SR register by setting the CCFC bit of the AES_CR register.
5. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.
6. For authenticated decryption, compare the generated encrypted tag with the encrypted tag padded in the ciphertext.

Note: In this final phase, swapping is applied to tag data read from AES_DOUTR register.

When transiting from the header phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Application must mask the authentication tag output with tag length to obtain a valid tag.

Suspend/resume operations in CCM mode

To suspend the processing of a message in header or payload phase, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES_SR register set to 1.
2. In the payload phase, if DMA is not used, read four times the AES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the

AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.

3. Clear the CCF flag of the AES_SR register, by setting to 1 the CCFC bit of the AES_CR register.
4. Save the AES_SUSPxR registers (where x is from 0 to 7) in the memory.
5. Save the AES_IVRx registers as, during the data processing, they changed from their initial values.
6. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.
7. Save the current AES configuration in the memory, excluding the initialization vector registers AES_IVRx. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPxR registers (where x is from 0 to 7).
4. Write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers.
5. Restore the initial setting values in the AES_CR and AES_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.
7. If DMA is used, enable AES DMA requests by setting to 1 the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES_CR register.

18.4.13 AES data registers and data swapping

Data input and output

A 128-bit data block is entered into the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 128-bit data block is retrieved from the AES peripheral with four successive 32-bit word reads from the AES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The 32-bit data word for AES_DINR register or from AES_DOUTR register is organized in big endian order, that is:

- the most significant byte of a word to write into AES_DINR must be put on the lowest address out of the four adjacent memory locations keeping the word to write, or
- the most significant byte of a word read from AES_DOUTR goes to the lowest address out of the four adjacent memory locations receiving the word

For using DMA for input data block write into AES, the four words of the input block must be stored in the memory consecutively and in big-endian order, that is, the most significant word on the lowest address. See *Section 18.4.16: AES DMA interface*.

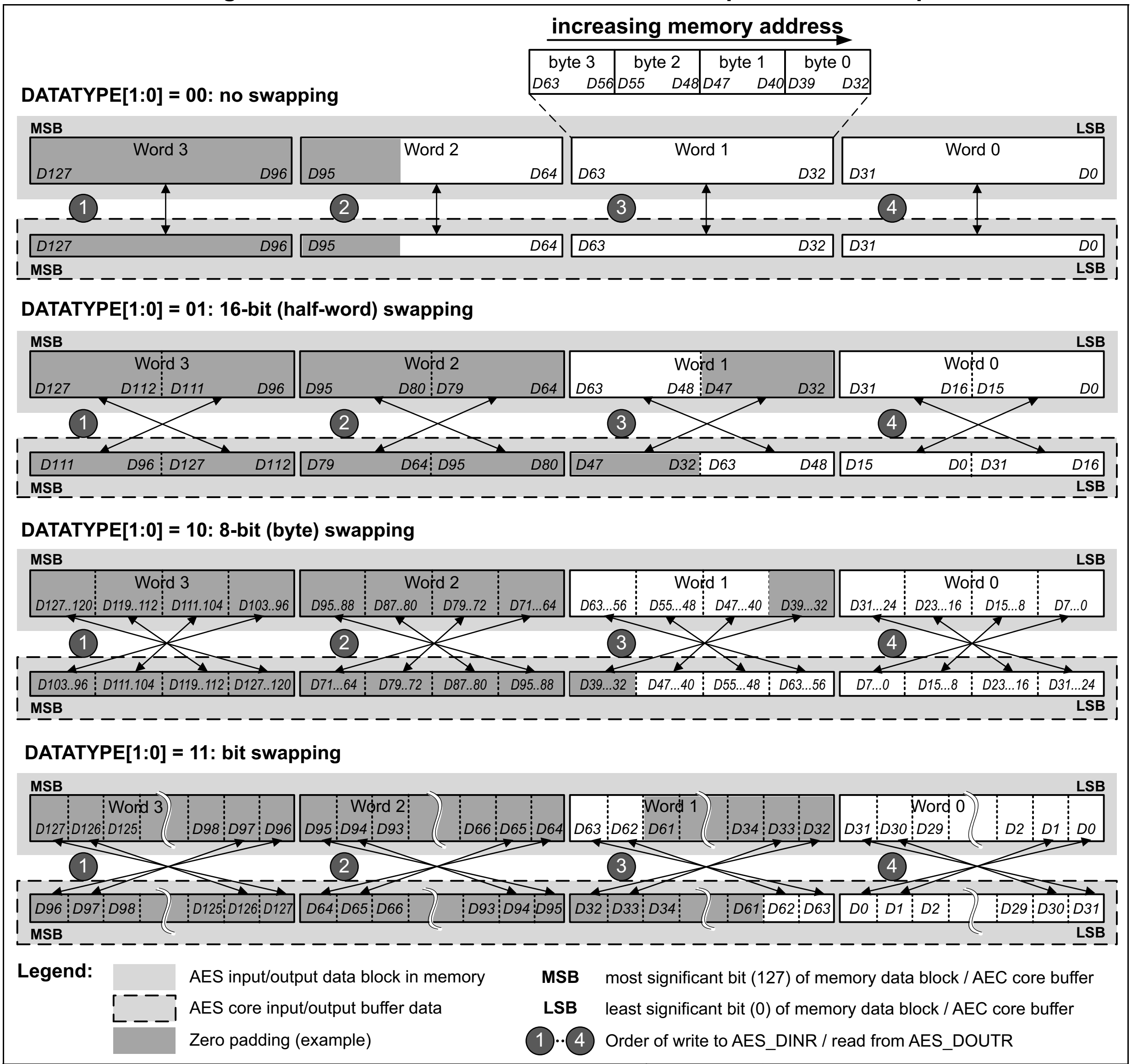
Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0] bitfield of the AES_CR register. The selection applies both to the input and the output of the AES core.

For different data swap types, *Figure 73* shows the construction of AES processing core input buffer data P127 to P0, from the input data entered through the AES_DINR register, or the construction of the output data available through the AES_DOUTR register, from the AES processing core output buffer data P127 to P0.

Figure 73. 128-bit block construction with respect to data swap



Note: The data in AES key registers (AES_KEYRx) and initialization registers (AES_IVRx) are not sensitive to the swap mode selection.

Data padding

Figure 73 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 48 message bits, with DATATYPE[1:0] = 01
- 56 message bits, with DATATYPE[1:0] = 10
- 34 message bits, with DATATYPE[1:0] = 11

18.4.14 AES key registers

The AES_KEYRx registers store the encryption or decryption key bitfield KEY[127:0] or KEY[255:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address.

The key is spread over eight registers as shown in *Table 55*.

Table 55. Key endianness in AES_KEYRx registers (128- or 256-bit key length)

AES_KEYR7 [31:0]	AES_KEYR6 [31:0]	AES_KEYR5 [31:0]	AES_KEYR4 [31:0]	AES_KEYR3 [31:0]	AES_KEYR2 [31:0]	AES_KEYR1 [31:0]	AES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key for encryption or decryption may be written into these registers when the AES peripheral is disabled, by clearing the EN bit of the AES_CR register.

The key registers are not affected by the data swapping controlled by DATATYPE[1:0] bitfield of the AES_CR register.

18.4.15 AES initialization vector registers

The four AES_IVRx registers keep the initialization vector input bitfield IVI[127:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address. The registers are also ordered from lowest address (AES_IVR0) to highest address (AES_IVR3).

The signification of data in the bitfield depends on the chaining mode selected. When used, the bitfield is updated upon each computation cycle of the AES core.

Write operations to the AES_IVRx registers when the AES peripheral is enabled have no effect to the register contents. For modifying the contents of the AES_IVRx registers, the EN bit of the AES_CR register must first be cleared.

Reading the AES_IVRx registers returns the latest counter value (useful for managing suspend mode).

The AES_IVRx registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield of the AES_CR register.

18.4.16 AES DMA interface

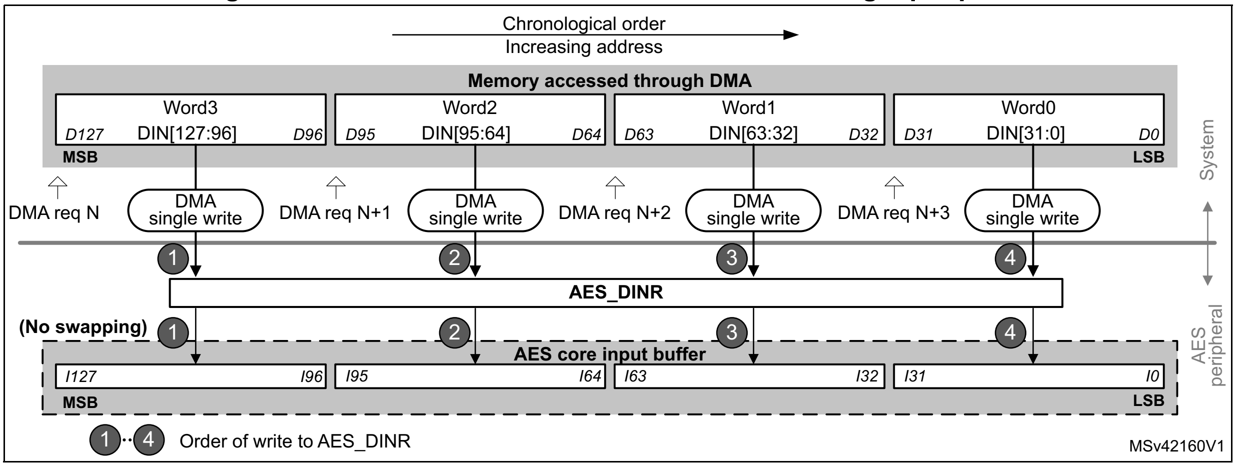
The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the AES_CR register.

Data input using DMA

Setting the DMAINEN bit of the AES_CR register enables DMA writing into AES. The AES peripheral then initiates a DMA request during the input phase each time it requires to write a 128-bit block (quadruple word) to the AES_DINR register, as shown in Figure 74.

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required. For example, in case of AES GCM encryption or AES CCM decryption, a DMA transfer must not include the last block. For details, refer to Section 18.4.4: AES procedure to perform a cipher operation.

Figure 74. DMA transfer of a 128-bit data block during input phase

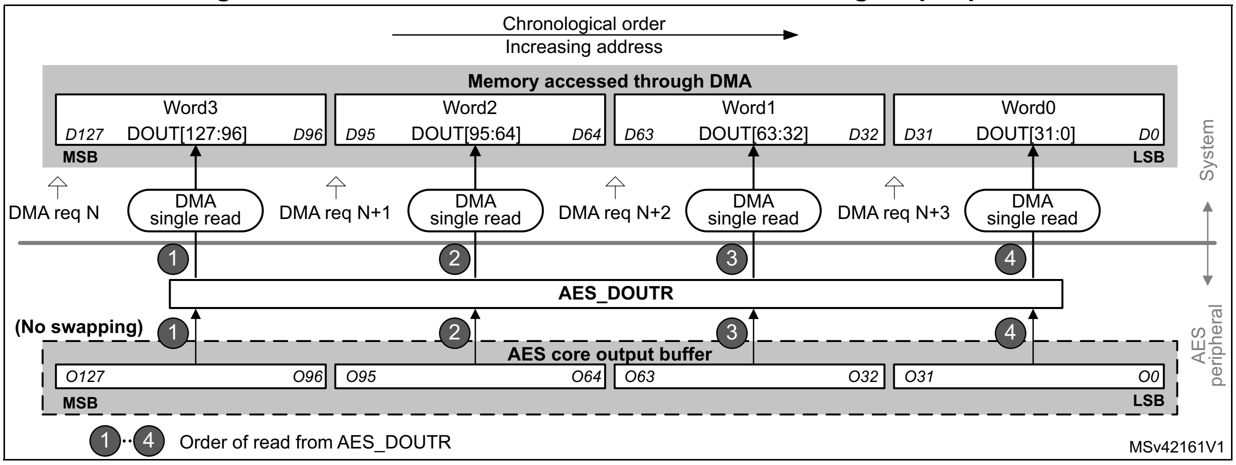


Data output using DMA

Setting the DMAOUTEN bit of the AES_CR register enables DMA reading from AES. The AES peripheral then initiates a DMA request during the Output phase each time it requires to read a 128-bit block (quadruple word) to the AES_DINR register, as shown in Figure 75.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Figure 75. DMA transfer of a 128-bit data block during output phase



DMA operation in different operating modes

DMA operations are usable when Mode 1 (encryption) or Mode 3 (decryption) are selected via the MODE[1:0] bitfield of the register AES_CR. As in Mode 2 (key derivation) the AES_KEYRx registers must be written by software, enabling the DMA transfer through the DMAINEN and DMAOUTEN bits of the AES_CR register have no effect in that mode.

DMA single requests are generated by AES until it is disabled. So, after the data output phase at the end of processing of a 128-bit data block, AES switches automatically to a new data input phase for the next data block, if any.

When the data transferring between AES and memory is managed by DMA, the CCF flag has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to data transferring managed by software. See *Section 18.4.4: AES procedure to perform a cipher operation*, subsection *Data append*, for details.

18.4.17 AES error management

AES configuration can be changed at any moment by clearing the EN bit of the AES_CR register.

Read error flag (RDERR)

Unexpected read attempt of the AES_DOUTR register sets the RDERR flag of the AES_SR register, and returns zero.

RDERR is triggered during the computation phase or during the input phase.

Note: AES is not disabled upon a RDERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES_CR register is set. For more details, refer to *Section 18.5: AES interrupts*.

The RDERR flag is cleared by setting the ERRIE bit of the AES_CR register.

Write error flag (WDERR)

Unexpected write attempt of the AES_DINR register sets the WRERR flag of the AES_SR register, and has no effect on the AES_DINR register. The WRERR is triggered during the computation phase or during the output phase.

Note: AES is not disabled after a WRERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES_CR register is set. For more details, refer to *Section 18.5: AES interrupts*.

The WRERR flag is cleared by setting the ERRC bit of the AES_CR register.

18.5 AES interrupts

Individual maskable interrupt sources generated by the AES peripheral signal the following events:

- computation completed
- read error
- write error

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Arm® Cortex® interrupt controller. Each can individually be enabled/disabled, by setting/clearing the corresponding enable bit of the AES_CR register, and cleared by setting the corresponding bit of the AES_CR register.

The status of each can be read from the AES_SR register.

Table 56 gives a summary of the interrupt sources, their event flags and enable bits.

Table 56. AES interrupt requests

Interrupt acronym	AES interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCFC(1)
	read error flag	RDERR	ERRIE	set ERRC(1)
	write error flag	WRERR		

1. Bit of the AES_CR register.

18.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

Table 57. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Algorithm	Clock cycles
128-bit	Mode 1: Encryption	ECB, CBC, CTR	51
	Mode 2: Key derivation	-	59
	Mode 3: Decryption	ECB, CBC, CTR	51
	Mode 4: Key derivation then decryption	ECB, CBC	106

Table 57. Processing latency for ECB, CBC and CTR (continued)

Key size	Mode of operation	Algorithm	Clock cycles
256-bit	Mode 1: Encryption	ECB, CBC, CTR	75
	Mode 2: Key derivation	-	82
	Mode 3: Decryption	ECB, CBC, CTR	75
	Mode 4: Key derivation then decryption	ECB, CBC	145

Table 58. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Algorithm	Init Phase	Header phase(1)	Payload phase(1)	Tag phase(1)
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	88	35	75	75
		CCM	87	79	162	82

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

18.7 AES registers

18.7.1 AES control register (AES_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
								rw	rw	rw	rw		rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

The bitfield sets the number of padding bytes in last block of payload:
0000: All bytes are valid (no padding)
0001: Padding for one least-significant byte of last block
...
1111: Padding for 15 least-significant bytes of last block

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the length of the key used in the AES cryptographic core, in bits:
0: 128
1: 256
Attempts to write the bit are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]:** GCM or CCM phase selection

This bitfield selects the phase of GCM, GMAC or CCM algorithm:
00: Init phase
01: Header phase
10: Payload phase
11: Final phase
The bitfield has no effect if other than GCM, GMAC or CCM algorithms are selected (through the ALGOMODE bitfield).

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables/disables data transferring with DMA, in the output phase:
0: Disable
1: Enable
When the bit is set, DMA requests are automatically generated by AES during the output data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).
Use of DMA with Mode 4 (single decryption) is not recommended.

Bit 11 **DMAINEN**: DMA input enable

This bit enables/disables data transferring with DMA, in the input phase:
0: Disable
1: Enable
When the bit is set, DMA requests are automatically generated by AES during the input data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).
Use of DMA with Mode 4 (single decryption) is not recommended.

Bit 10 **ERRIE**: Error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RDERR and/or WRERR is set:
0: Disable (mask)
1: Enable

Bit 9 **CCFIE**: CCF interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set:
0: Disable (mask)
1: Enable

Bit 8 **ERRC**: Error flag clear

Upon written to 1, this bit clears the RDERR and WRERR error flags in the AES_SR register:
0: No effect
1: Clear RDERR and WRERR flags
Reading the flag always returns zero.

Bit 7 **CCFC:** Computation complete flag clear

Upon written to 1, this bit clears the computation complete flag (CCF) in the AES_SR register:
0: No effect
1: Clear CCF
Reading the flag always returns zero.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode selection

This bitfield selects the AES chaining mode:
000: Electronic codebook (ECB)
001: Cipher-block chaining (CBC)
010: Counter mode (CTR)
011: Galois counter mode (GCM) and Galois message authentication code (GMAC)
100: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: AES operating mode

This bitfield selects the AES operating mode:
00: Mode 1: encryption
01: Mode 2: key derivation (or key preparation for ECB/CBC decryption)
10: Mode 3: decryption
11: Mode 4: key derivation then single decryption
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access. Any attempt to selecting Mode 4 while either ECB or CBC chaining mode is not selected, defaults to effective selection of Mode 3. It is not possible to select a Mode 3 following a Mode 4.

Bits 2:1 **DATATYPE[1:0]**: Data type selection

This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register, through selecting the mode of data swapping:
00: None
01: Half-word (16-bit)
10: Byte (8-bit)
11: Bit
For more details, refer to *Section 18.4.13: AES data registers and data swapping*.
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: AES enable

This bit enables/disables the AES peripheral:
0: Disable
1: Enable
At any moment, clearing then setting the bit re-initializes the AES peripheral.
This bit is automatically cleared by hardware upon the completion of the key preparation (Mode 2) and upon the completion of GCM/GMAC/CCM initial phase.

18.7.2 AES status register (AES_SR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

Bit 3 **BUSY:** Busy

This flag indicates whether AES is idle or busy during GCM payload **encryption** phase:

0: Idle

1: Busy

When the flag indicates "idle", the current GCM encryption processing may be suspended to process a higher-priority message. In other chaining modes, or in GCM phases other than payload encryption, the flag must be ignored for the suspend process.

Bit 2 **WRERR**: Write error

This flag indicates the detection of an unexpected write operation to the AES_DINR register (during computation or data output phase):

0: Not detected

1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES_CR register.

The flag setting has no impact on the AES operation. Unexpected write is ignored.

Bit 1 **RDERR**: Read error flag

This flag indicates the detection of an unexpected read operation from the AES_DOUTR register (during computation or data input phase):

0: Not detected

1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES_CR register.

The flag setting has no impact on the AES operation. Unexpected read returns zero.

Bit 0 **CCF**: Computation completed flag

This flag indicates whether the computation is completed:

0: Not completed

1: Completed

The flag is set by hardware upon the completion of the computation. It is cleared by software, upon setting the CCFC bit of the AES_CR register.

Upon the flag setting, an interrupt is generated if enabled through the CCFIE bit of the AES_CR register.

The flag is significant only when the DMAOUTEN bit is 0. It may stay high when DMA_EN is 1.

18.7.3 AES data input register (AES_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

Only 32-bit access type is supported.

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

Bits 31:0 **DIN[31:0]**: Input data word

A four-fold sequential write to this bitfield during the input phase results in writing a complete 128-bit block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 128-bit input buffer.

The data signification of the input data block depends on the AES operating mode:

- **Mode 1** (encryption): plaintext
- **Mode 2** (key derivation): the bitfield is not used (AES_KEYRx registers used for input)
- **Mode 3** (decryption) and **Mode 4** (key derivation then single decryption): ciphertext

The data swap operation is described in *Section 18.4.13: AES data registers and data swapping on page 365*.

18.7.4 AES data output register (AES_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

Only 32-bit read access type is supported.

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

Bits 31:0 **DOUT[31:0]**: Output data word

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF set), virtually reads a complete 128-bit block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

The data signification of the output data block depends on the AES operating mode:

- **Mode 1** (encryption): ciphertext
- **Mode 2** (key derivation): the bitfield is not used (AES_KEYRx registers used for output)
- **Mode 3** (decryption) and **Mode 4** (key derivation then single decryption): plaintext

The data swap operation is described in *Section 18.4.13: AES data registers and data swapping on page 365*.

18.7.5 AES key register 0 (AES_KEYR0)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

This bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode:

- In **Mode 1** (encryption), **Mode 2** (key derivation) and **Mode 4** (key derivation then single decryption): the value to write into the bitfield is the encryption key.

- In **Mode 3** (decryption): the value to write into the bitfield is the encryption key to be derived before being used for decryption. After writing the encryption key into the bitfield, its reading before enabling AES returns the same value. Its reading after enabling AES and after the CCF flag is set returns the decryption key derived from the encryption key.

Note: In mode 4 (key derivation then single decryption) the bitfield always contains the encryption key.

The AES_KEYRx registers may be written only when KEYSIZE value is correct and when the AES peripheral is disabled (EN bit of the AES_CR register cleared). Note that, if, the key is directly loaded to AES_KEYRx registers (hence writes to key register is ignored and KEIF is set).

Refer to *Section 18.4.14: AES key registers on page 367* for more details.

18.7.6 AES key register 1 (AES_KEYR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

18.7.7 AES key register 2 (AES_KEYR2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

18.7.8 AES key register 3 (AES_KEYR3)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

18.7.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

Refer to *Section 18.4.15: AES initialization vector registers on page 367* for description of the IVI[127:0] bitfield.

The initialization vector is only used in chaining modes other than ECB.

The AES_IVRx registers may be written only when the AES peripheral is disabled

18.7.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

18.7.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

18.7.12 AES initialization vector register 3 (AES_IVR3)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

18.7.13 AES key register 4 (AES_KEYR4)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
KEY[143:128]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

18.7.14 AES key register 5 (AES_KEYR5)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

18.7.15 AES key register 6 (AES_KEYR6)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

18.7.16 AES key register 7 (AES_KEYR7)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

Note: The key registers from 4 to 7 are used only when the key length of 256 bits is selected. They have no effect when the key length of 128 bits is selected (only key registers 0 to 3 are used in that case).

18.7.17 AES suspend registers (AES_SUSPxR)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

These registers contain the complete internal register states of the AES processor when the AES processing of the current task is suspended to process a higher-priority task.

Upon suspend, the software reads and saves the AES_SUSPxR register contents (where x is from 0 to 7) into memory, before using the AES processor for the higher-priority task. Upon completion, the software restores the saved contents back into the corresponding suspend registers, before resuming the original task.

Note: These registers are used only when GCM, GMAC, or CCM chaining mode is selected.

These registers can be read only when AES is enabled. Reading these registers while AES is disabled returns 0x0000 0000.

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

Bits 31:0 **SUSP[31:0]**: AES suspend

Upon suspend operation, this bitfield of the corresponding AES_SUSPxR register takes the value of one of internal AES registers.

18.7.18 AES register map

Table 59. AES register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	AES_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	WRERR	RDERR	CCF
	Reset value																													0	0	0	0
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	AES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	AES_SUSP0R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59. AES register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x044	AES_SUSP1R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	AES_SUSP2R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	AES_SUSP3R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	AES_SUSP4R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	AES_SUSP5R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	AES_SUSP6R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	AES_SUSP7R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x3FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.2.2 on page 48* for the register boundary addresses.

19 Cyclic redundancy check calculation unit (CRC)

19.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

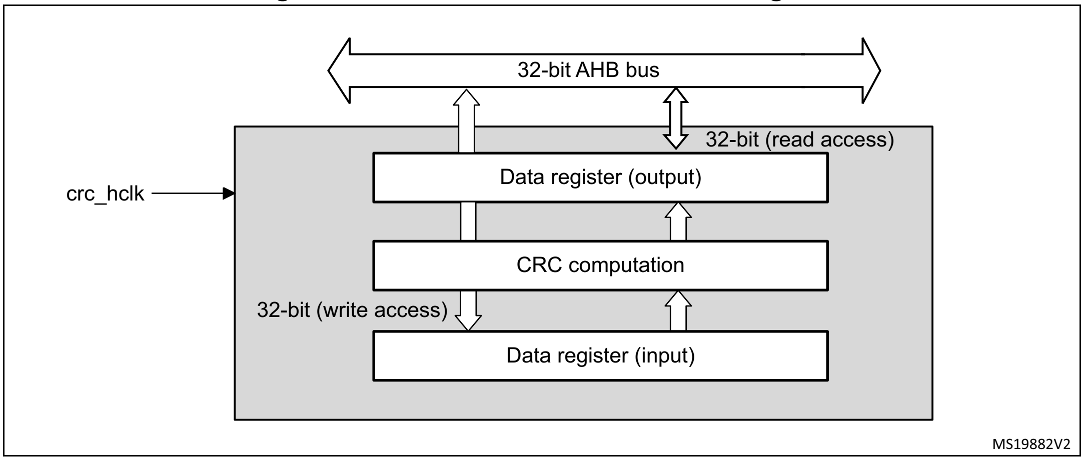
19.2 CRC main features

- Fully programmable polynomial with programmable size (7, 8, 16, 32 bits).
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data

19.3 CRC functional description

19.3.1 CRC block diagram

Figure 76. CRC calculation unit block diagram



19.3.2 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit access is allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32-bit
- 2 AHB clock cycles for 16-bit
- 1 AHB clock cycles for 8-bit

An input buffer allows to immediately write a second data without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example: input data 0x1A2B3C4D is used for CRC calculation as:

0x58D43CB2 with bit-reversal done by byte

0xD458B23C with bit-reversal done by half-word

0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level: for example, output data 0x11223344 is converted into 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

19.4 CRC registers

19.4.1 Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

19.4.2 Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

19.4.3 Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rs

Bits 31:8 Reserved, must be kept cleared.

Bit 7 **REV_OUT**: Reverse output data

This bit controls the reversal of the bit order of the output data.

0: Bit order not affected

1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data

These bits control the reversal of the bit order of the input data

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept cleared.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

19.4.4 Initial CRC value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

Bits 31:0 **CRC_INIT**: *Programmable initial CRC value*

This register is used to write the CRC initial value.

19.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C11DB7

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

Bits 31:0 **POL[31:0]**: Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

19.5 CRC register map

Table 60. CRC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																									0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0x04C11DB7																															

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

20 General purpose timer (TIM2)

In this section, “TIMx” should be understood as “TIM2” since there is only one instance of this timer in the STM32WL30xx/31xx/33xx.

20.1 TIM2 introduction

The general purpose timers (TIM2) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

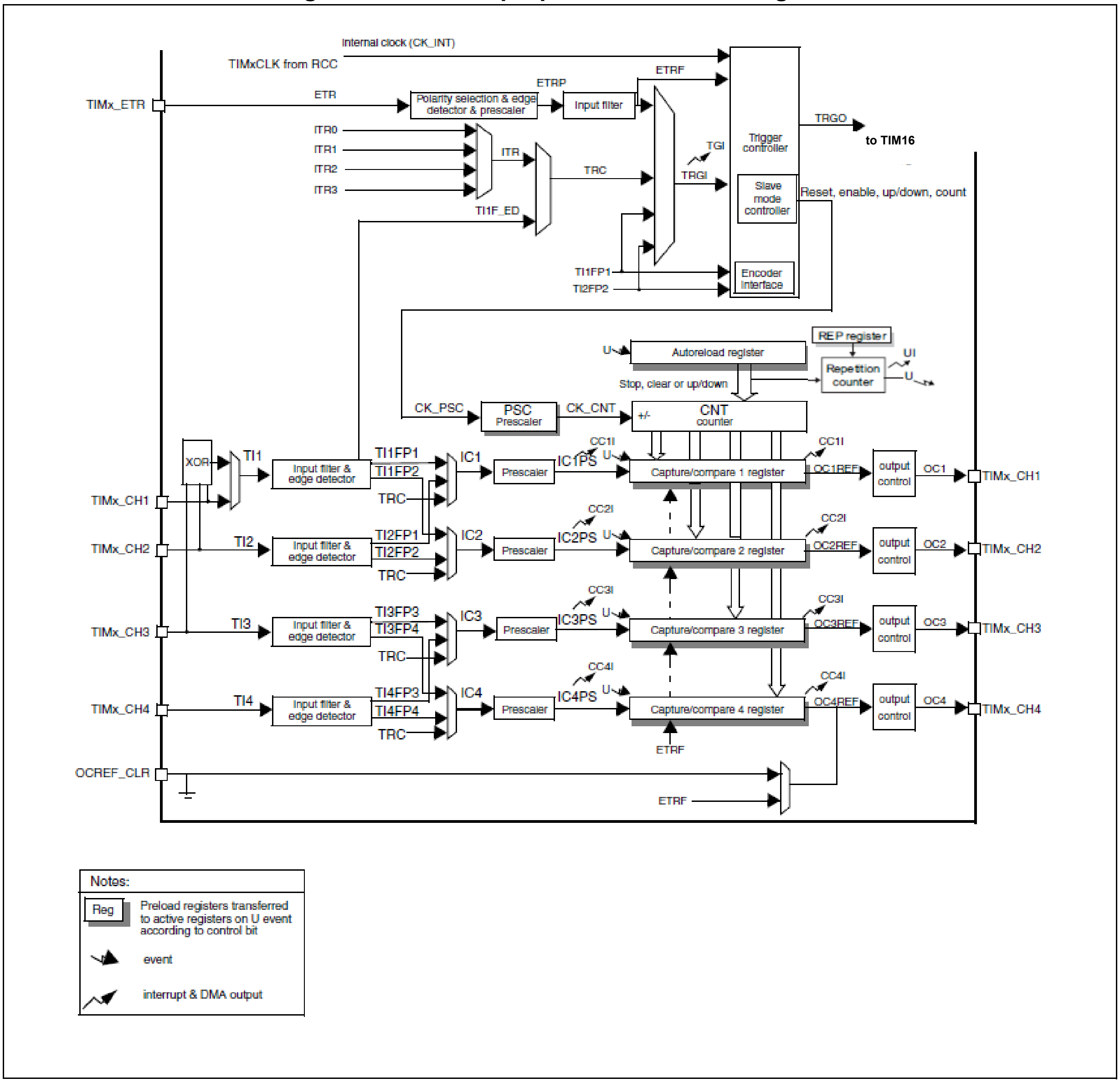
Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler on the timer input clock which is at 32 MHz.

20.2 TIM2 main features

TIM2 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 4independent channels for:
  - Input Capture
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 77. General purpose timer block diagram



20.3 TIM2 functional description

20.3.1 Time-base unit

The main block of the programmable general purpose timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 78 and *Figure 79* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 78. Counter timing diagram with prescaler division change from 1 to 2

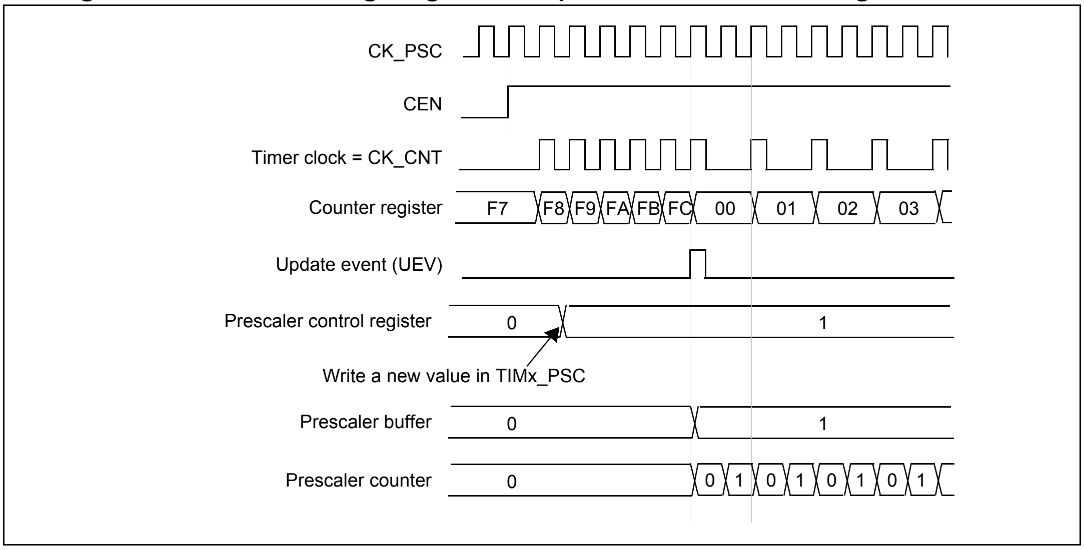
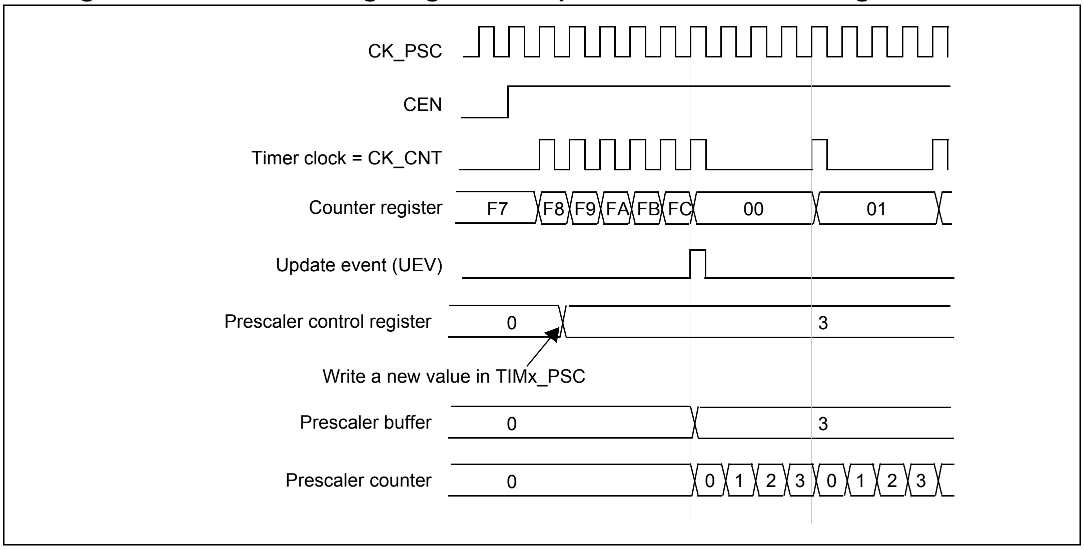


Figure 79. Counter timing diagram with prescaler division change from 1 to 4



20.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 80. Counter timing diagram, internal clock divided by 1

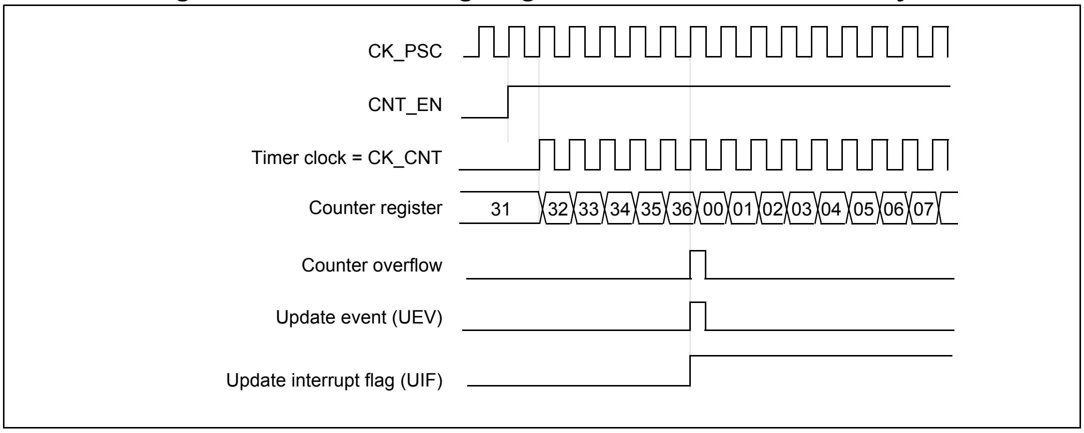


Figure 81. Counter timing diagram, internal clock divided by 2

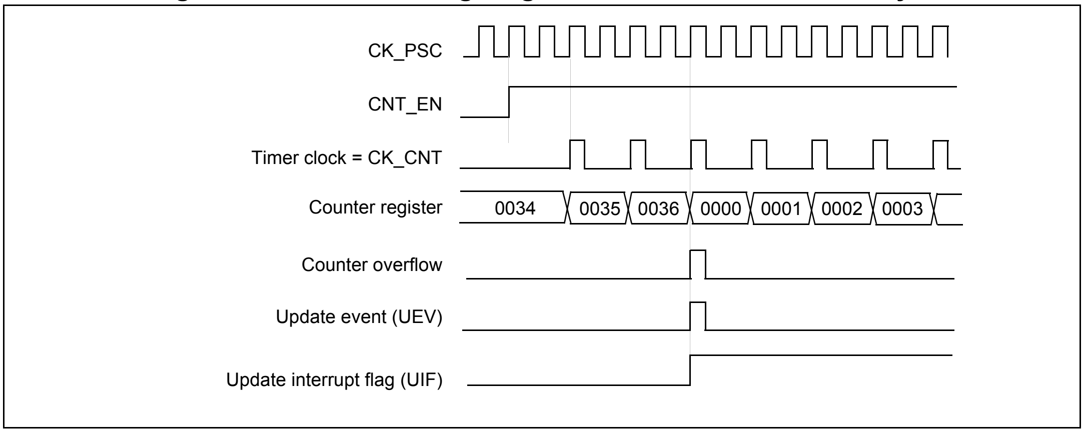


Figure 82. Counter timing diagram, internal clock divided by 4

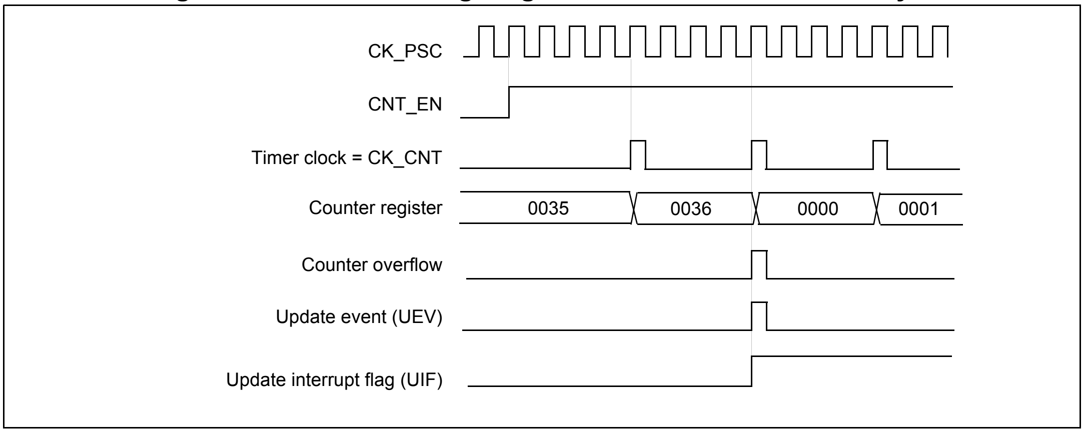


Figure 83. Counter timing diagram, internal clock divided by N

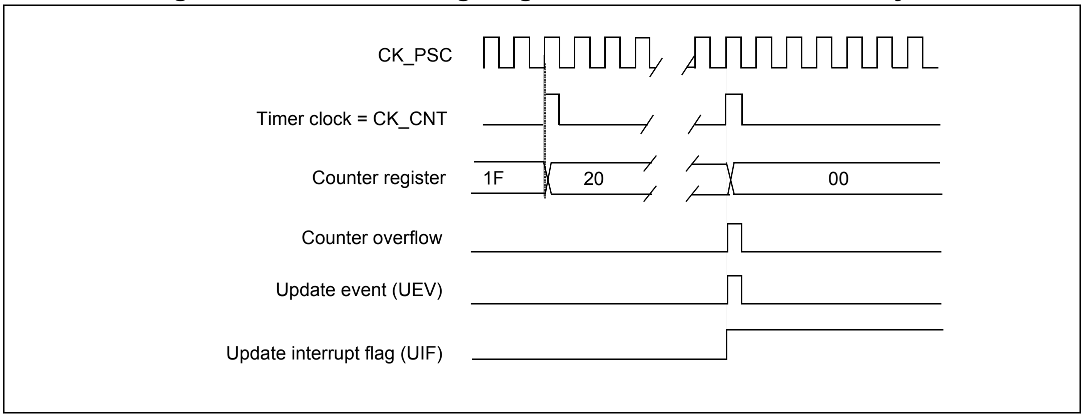


Figure 84. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

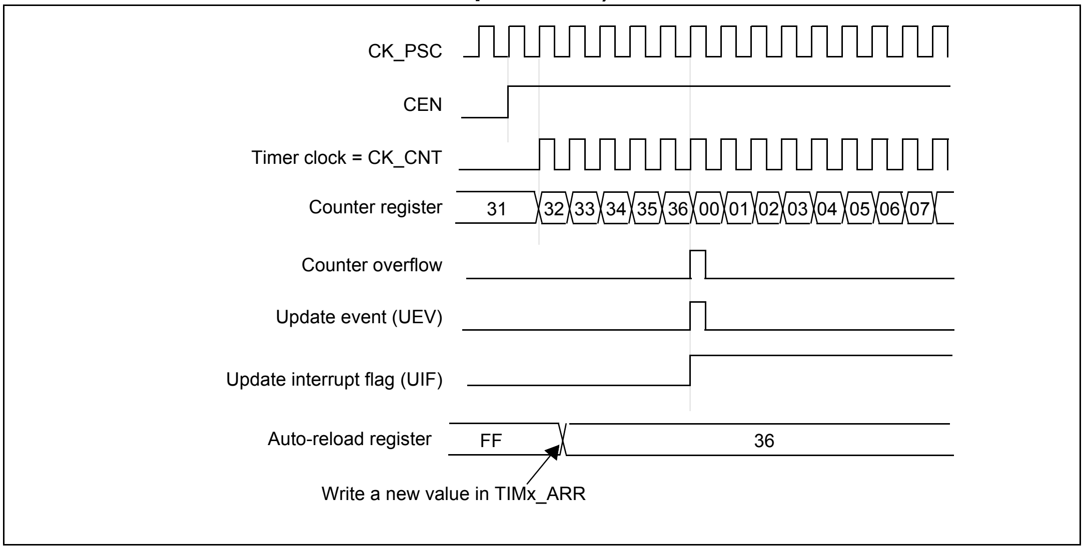
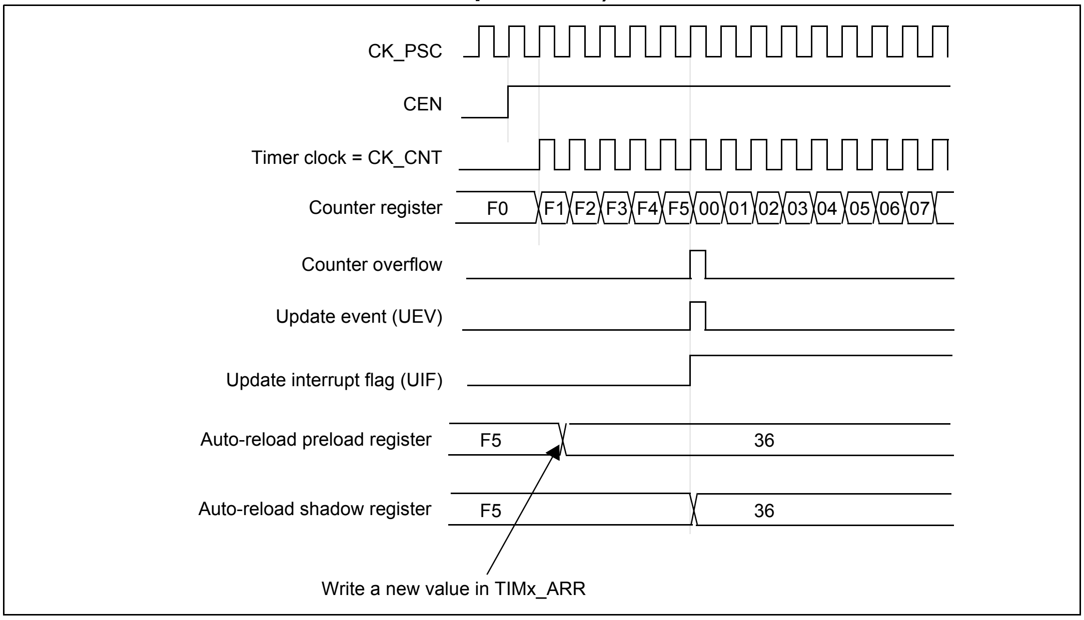


Figure 85. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 86. Counter timing diagram, internal clock divided by 1

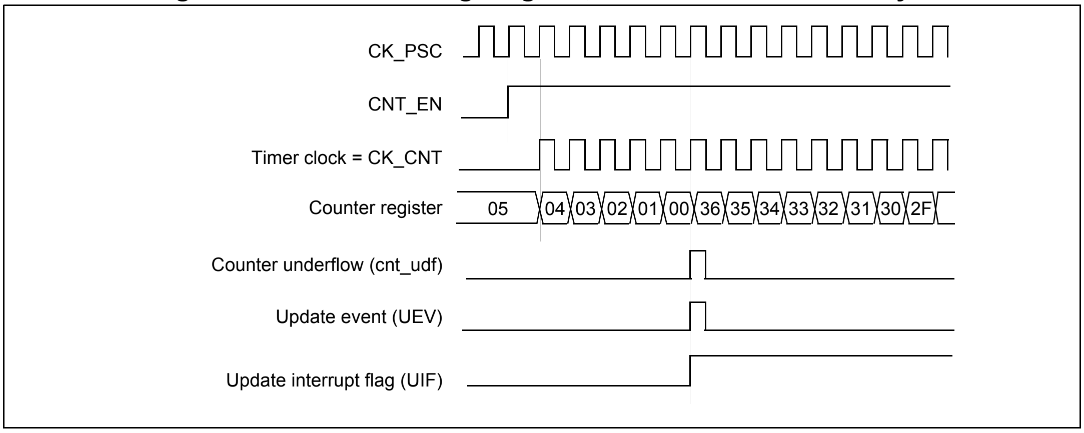


Figure 87. Counter timing diagram, internal clock divided by 2

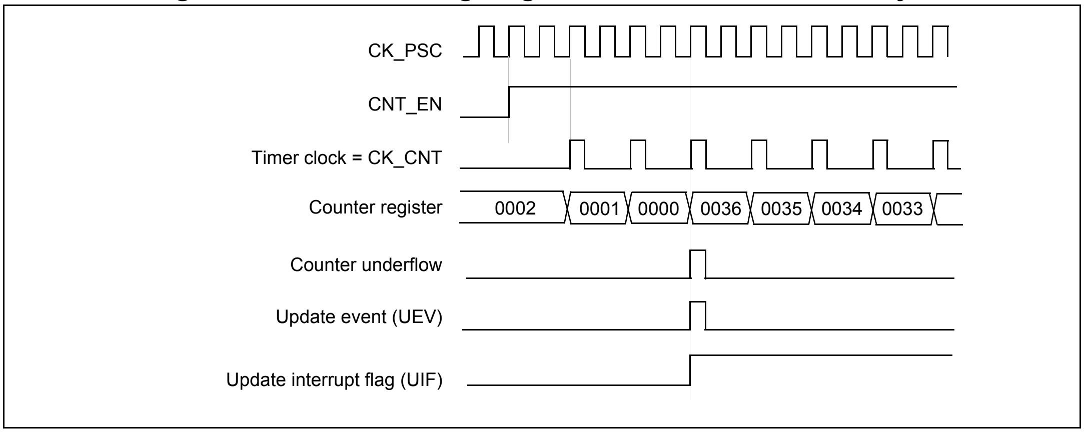


Figure 88. Counter timing diagram, internal clock divided by 4

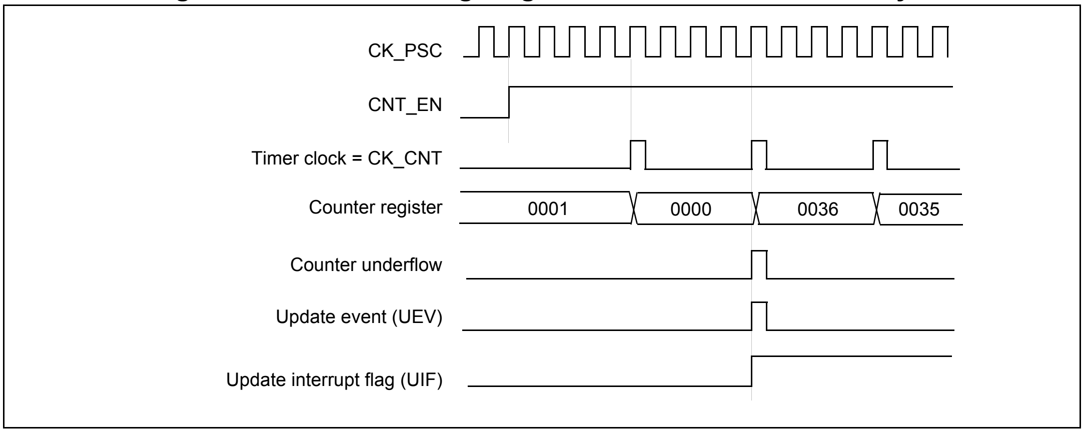


Figure 89. Counter timing diagram, internal clock divided by N

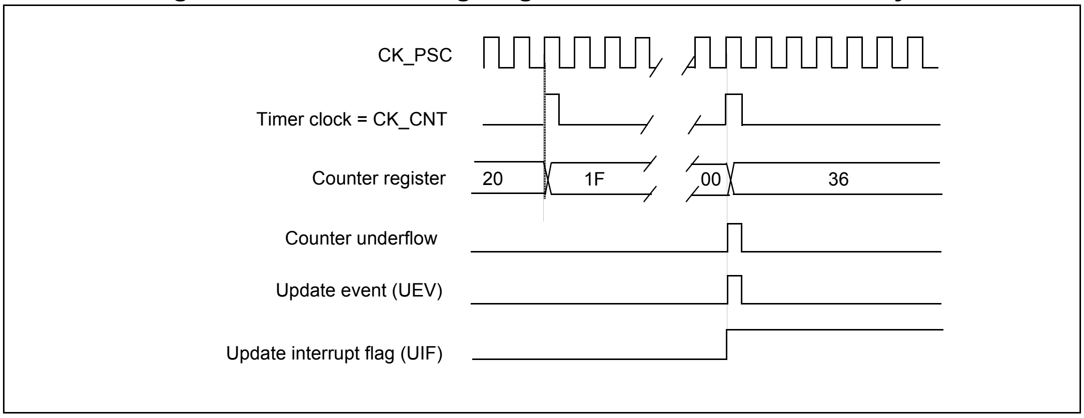
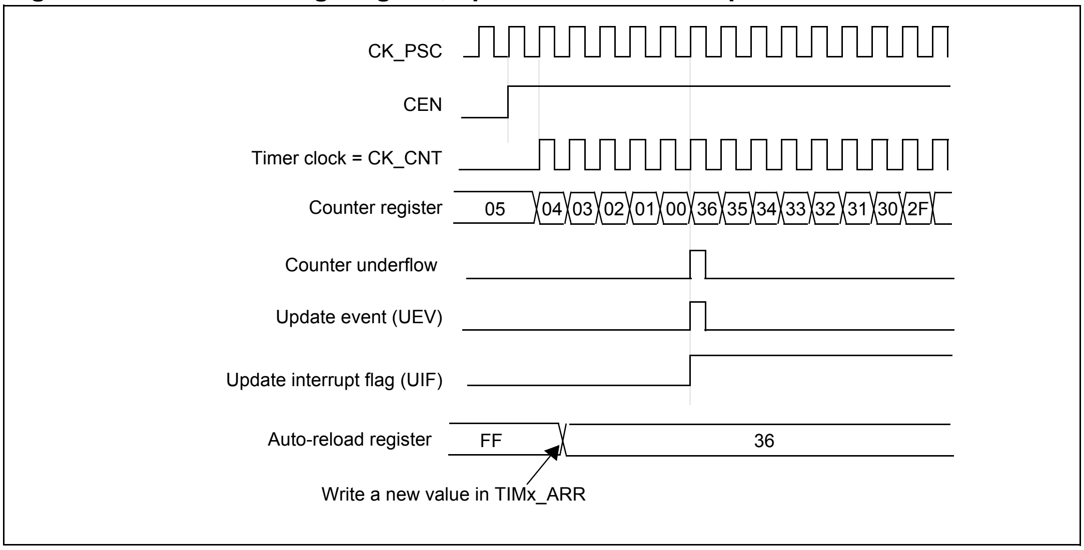


Figure 90. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

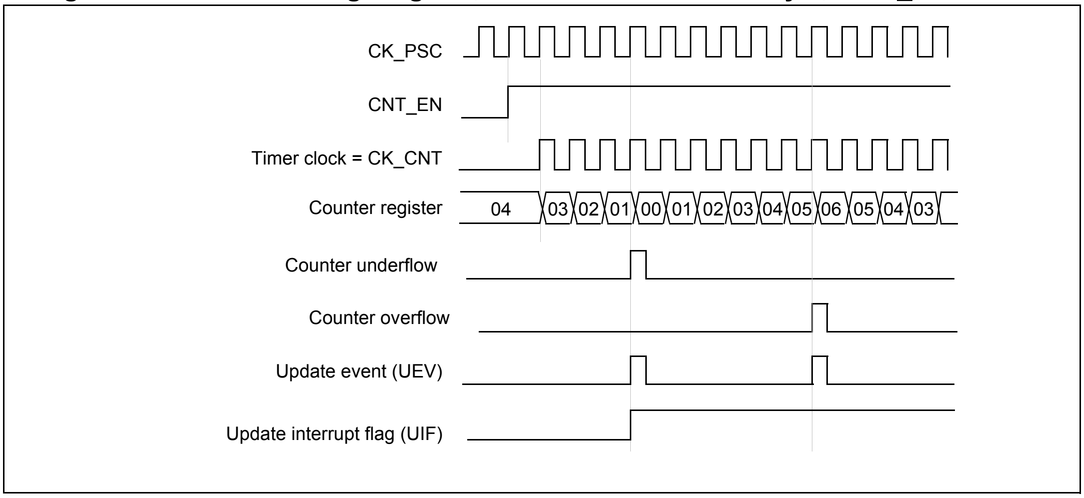
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 91. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 20.4: TIM2 registers*).

Figure 92. Counter timing diagram, internal clock divided by 2

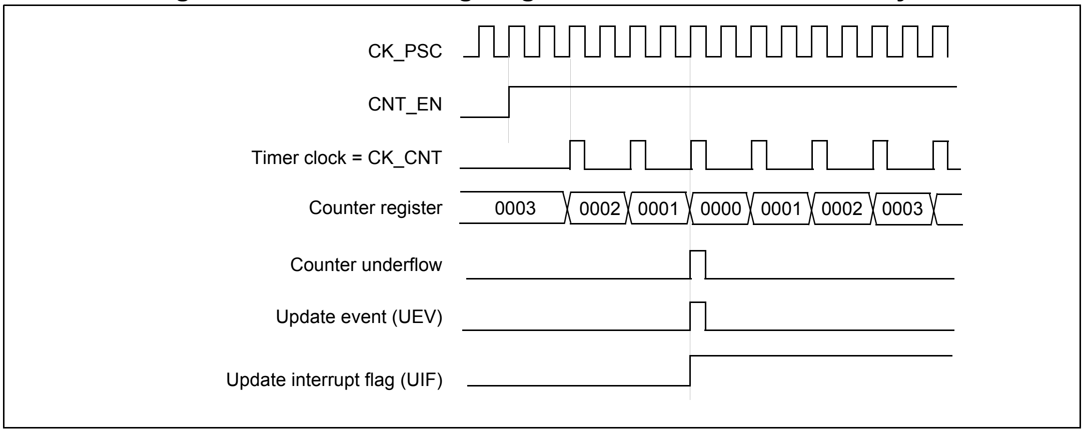


Figure 93. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

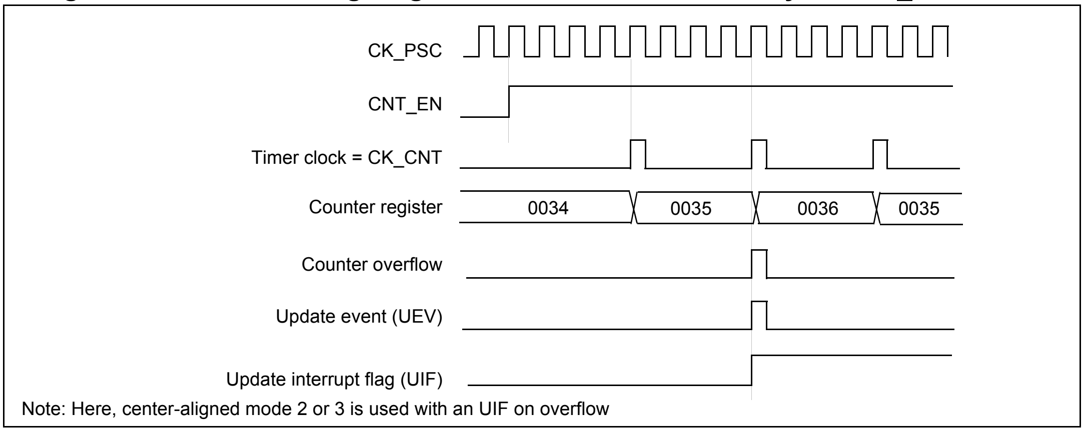


Figure 94. Counter timing diagram, internal clock divided by N

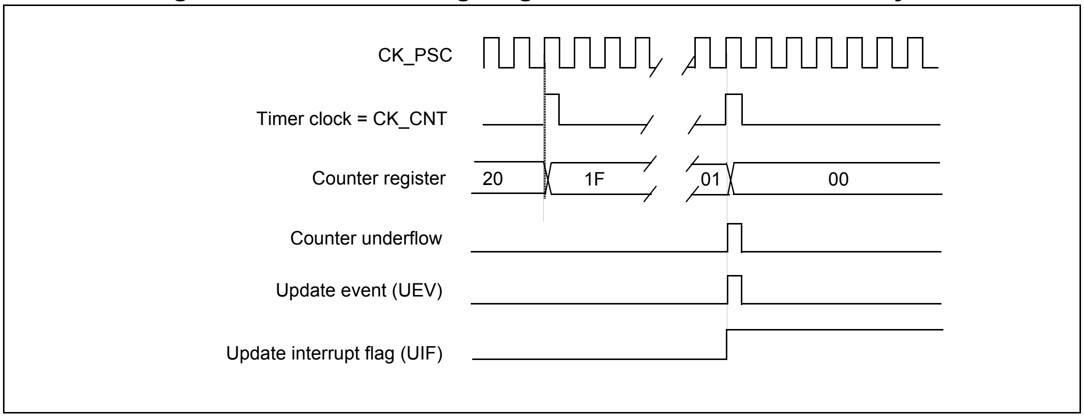


Figure 95. Counter timing diagram, update event with ARPE=1 (counter underflow)

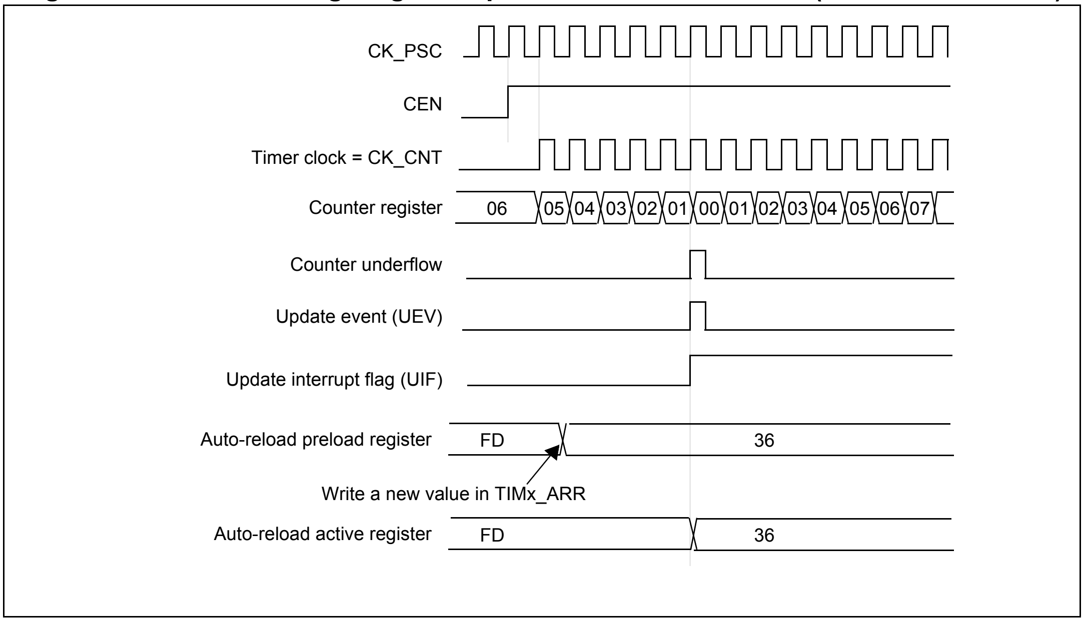
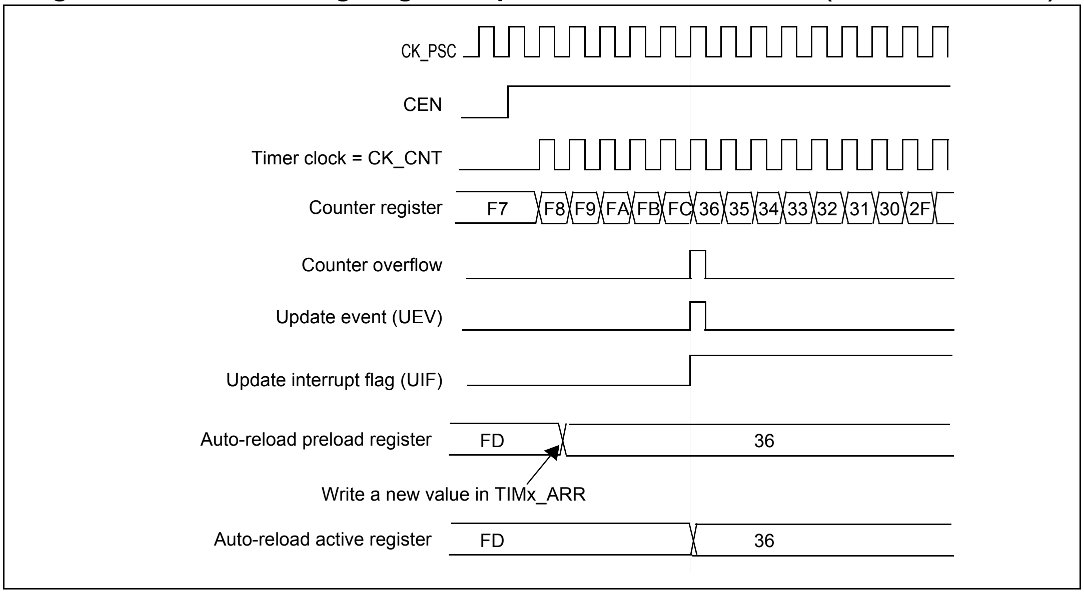


Figure 96. Counter timing diagram, Update event with ARPE=1 (counter overflow)



20.3.3 Repetition counter

Section 20.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

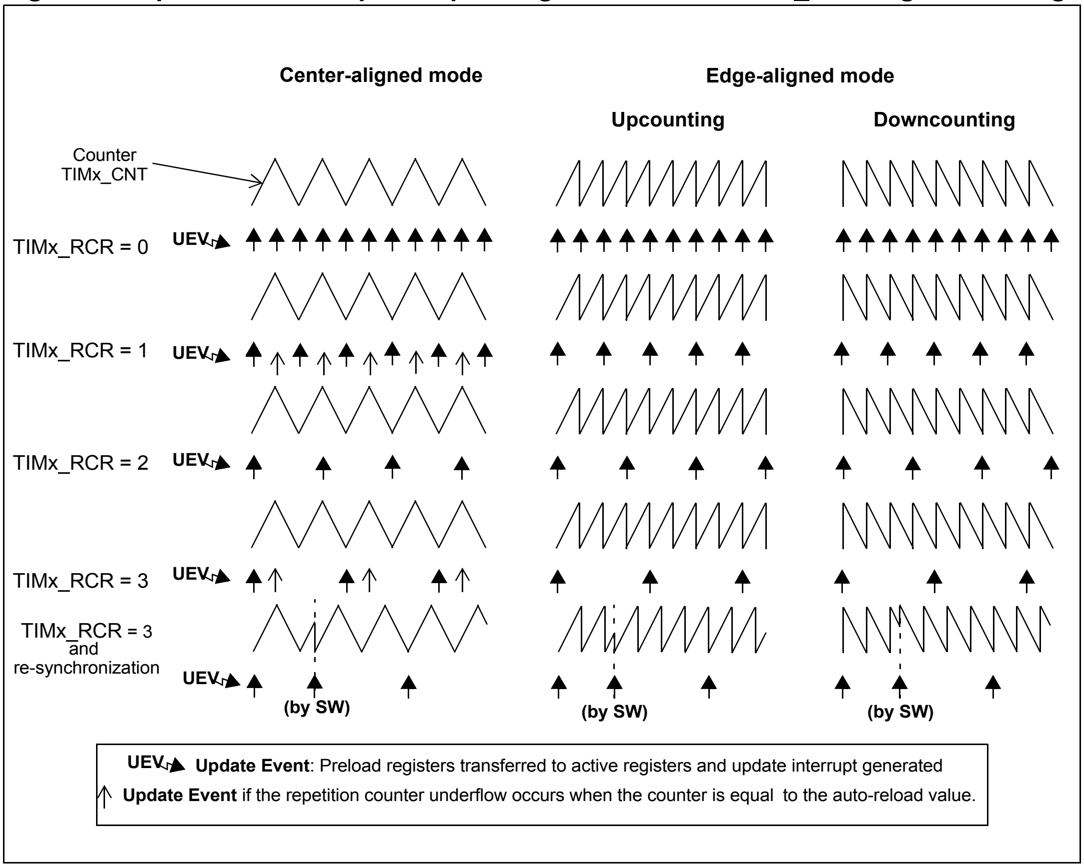
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 97*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the overflow. If the RCR was written after launching the counter, the UEV occurs on the underflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 97. Update rate examples depending on mode and TIMx_RCR register settings



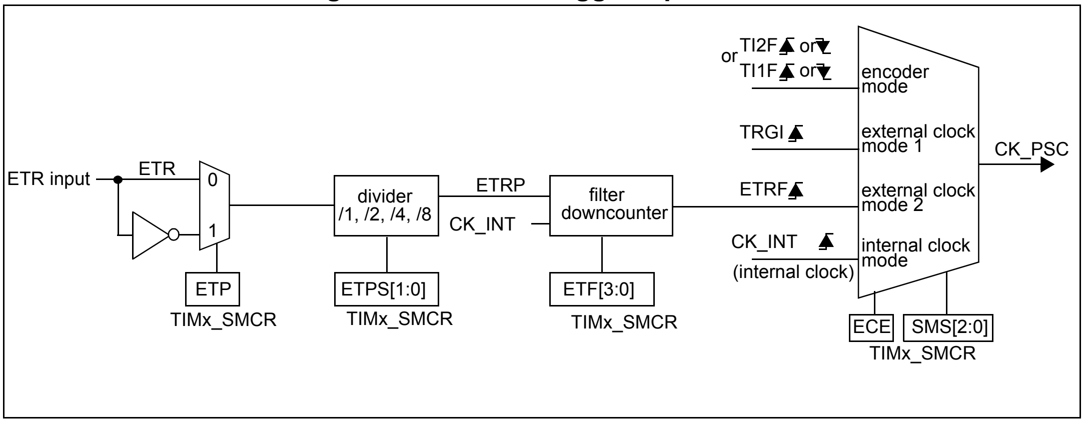
20.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see *Section 20.3.5*)
- trigger for the slave mode (see *Section 20.3.4*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 20.3.14*)

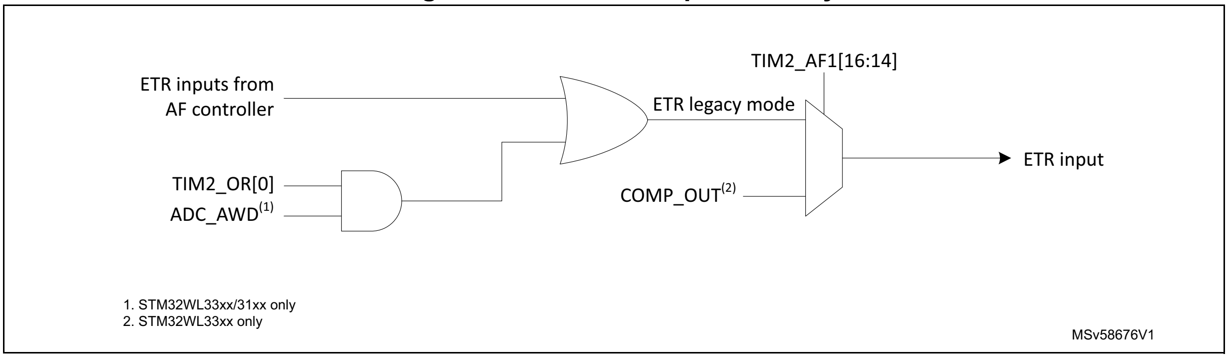
The *Figure 98* below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMx_SMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bit field.

Figure 98. External trigger input block



The ETR input comes from multiple sources: input pins (default configuration, see *Table 8: GPIO alternate options AF0 - AF3* and *Table 9: GPIO alternate options AF4 - AF6*), comparator output and analog watchdog. The selection is done with the TIM2_AF1.ETRSEL[2:0] and the TIM2_OR[0] bit.

Figure 99. TIM2 ETR input circuitry



20.3.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)

Note: Only channel 1 and channel 2 support the external clock mode 1.

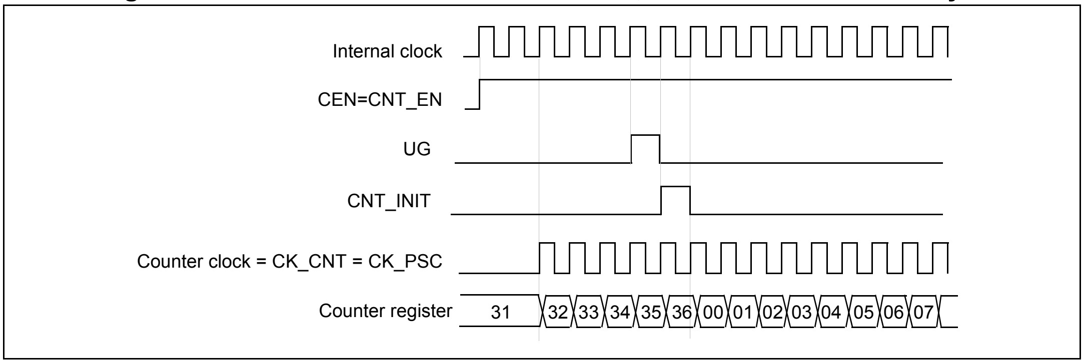
- External clock mode2: external trigger input (ETR)
- Encoder mode

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 100 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

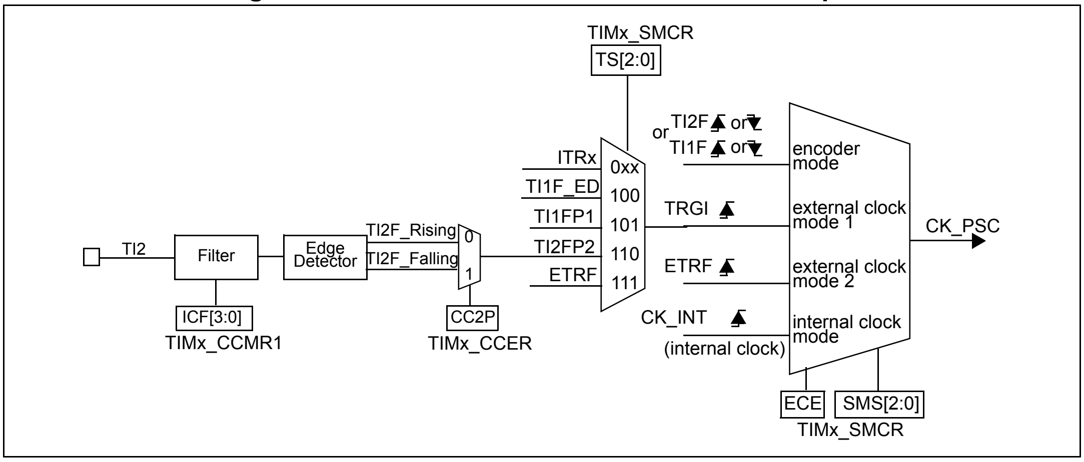
Figure 100. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 101. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

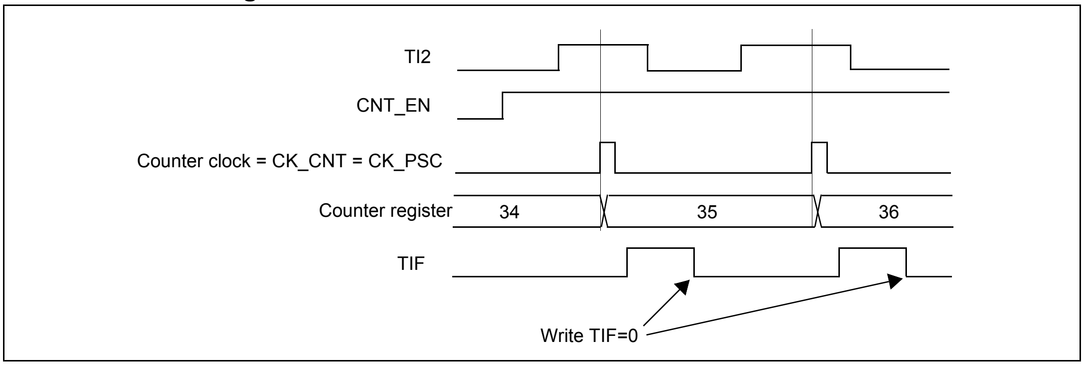
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don’t need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 102. Control circuit in external clock mode 1



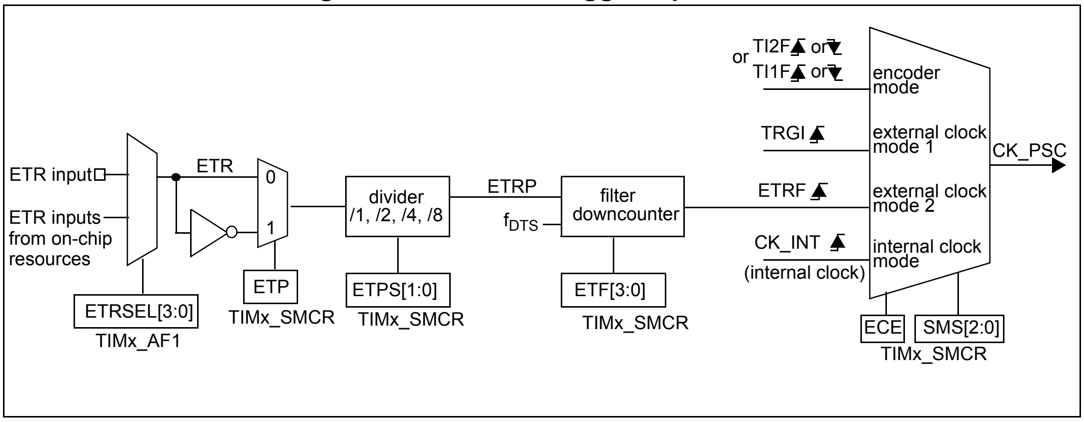
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The *Figure 103* gives an overview of the external trigger input block.

Figure 103. External trigger input block



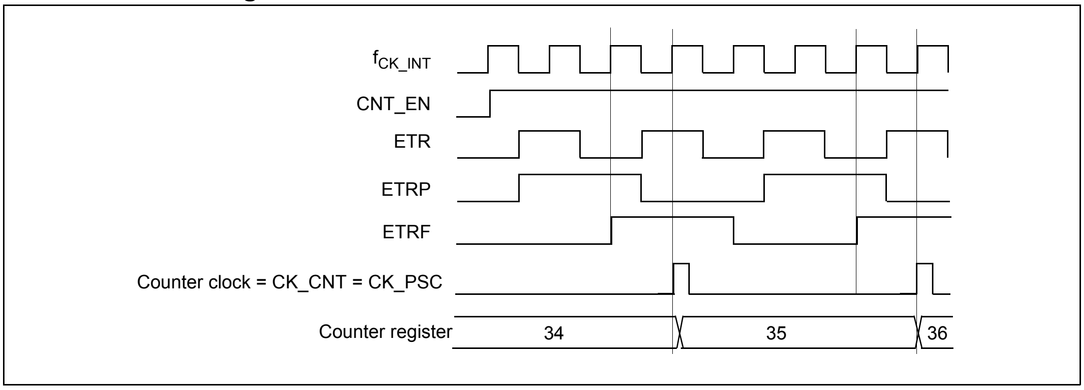
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register and the ETR1_RMP bit in the TIM2_OR1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
4. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
5. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 104. Control circuit in external clock mode 2



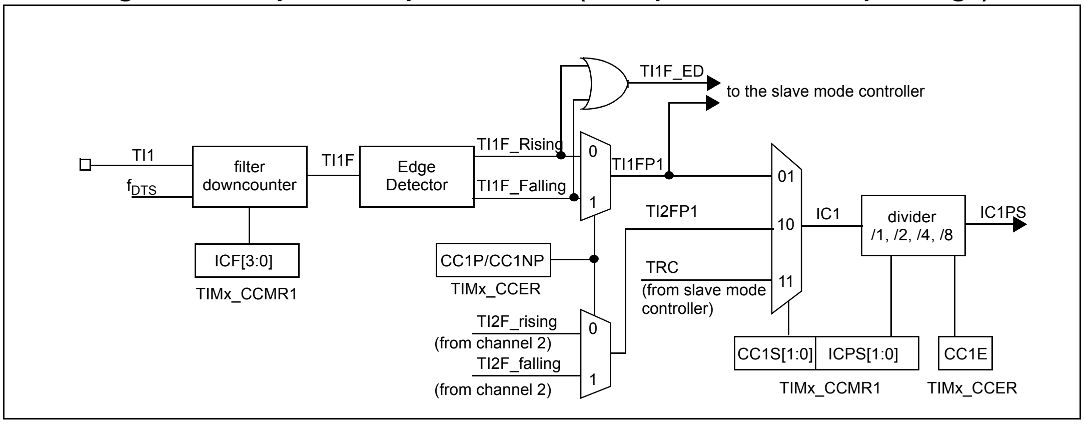
20.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 105 to *Figure 107* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 105. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 106. Capture/compare channel 1 main circuit

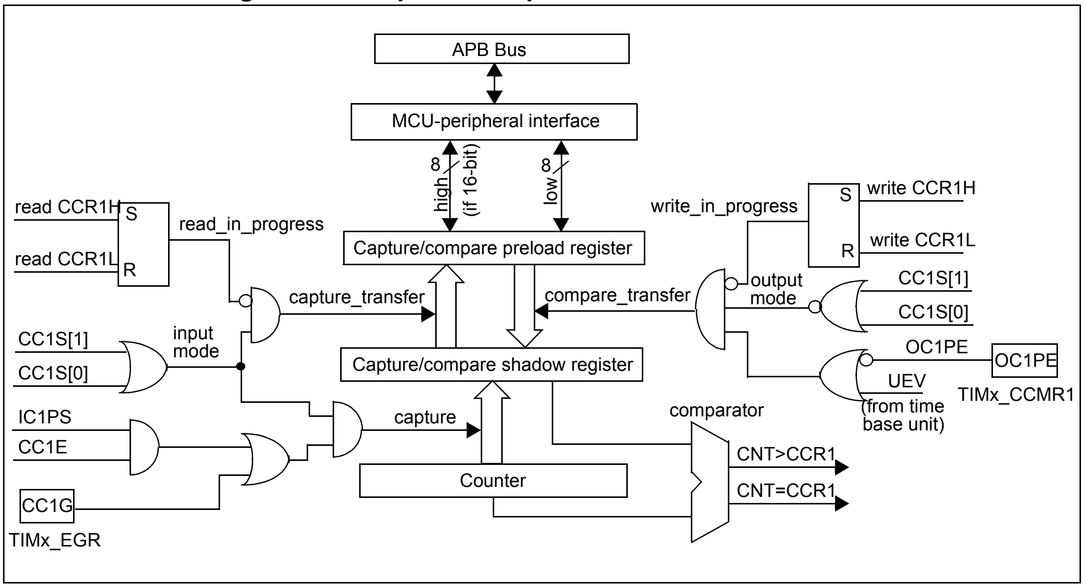
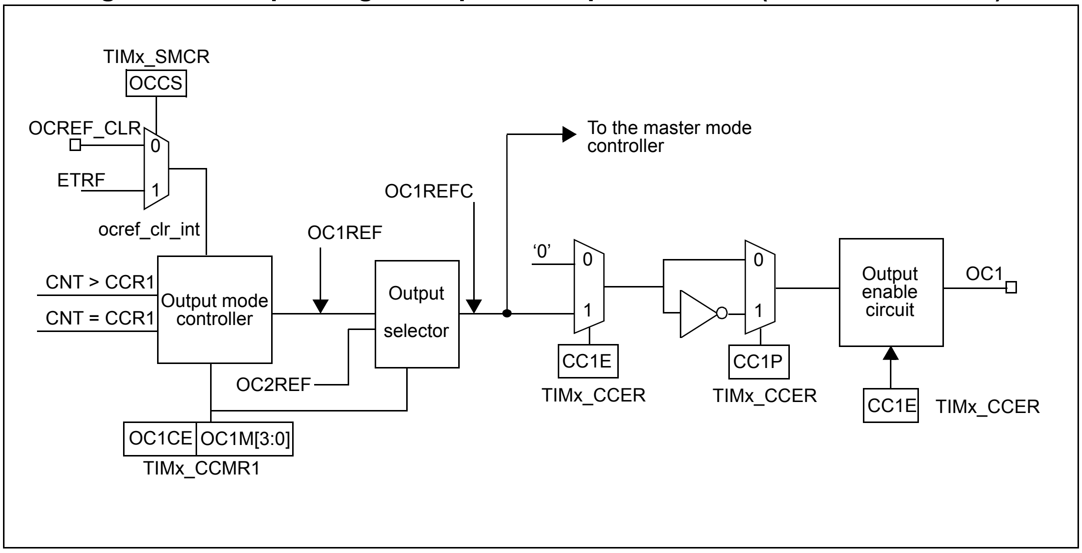


Figure 107. Output stage of capture/compare channel (channels 1, 2, 3, 4)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

20.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).Program the input

prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).

- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

20.3.8 PWM input mode

Note: Only channel 1 and channel 2 support this PWM input mode.

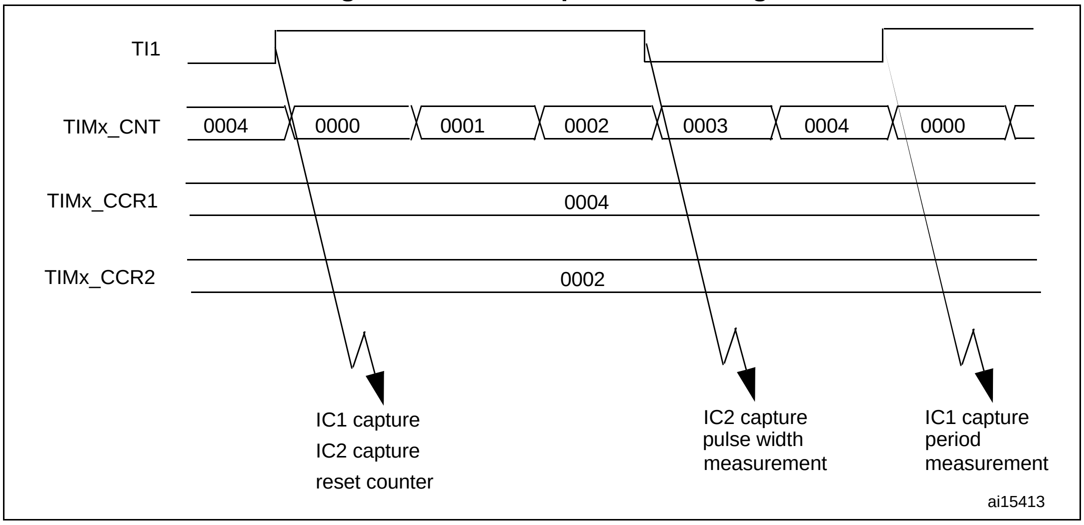
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP=’10’ (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 108. PWM input mode timing



20.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

20.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

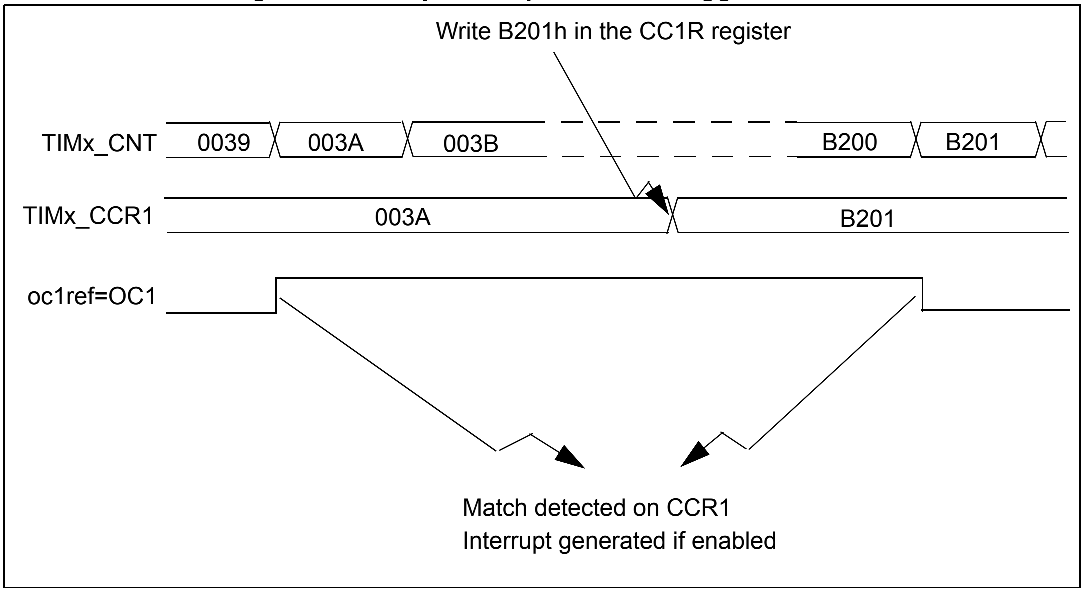
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   – Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
   – Write OCxPE = 0 to disable preload register
   – Write CCxP = 0 to select active high polarity
   – Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 109*.

Figure 109. Output compare mode, toggle on OC1



20.3.11 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤TIMx_CNT or TIMx_CNT ≤TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

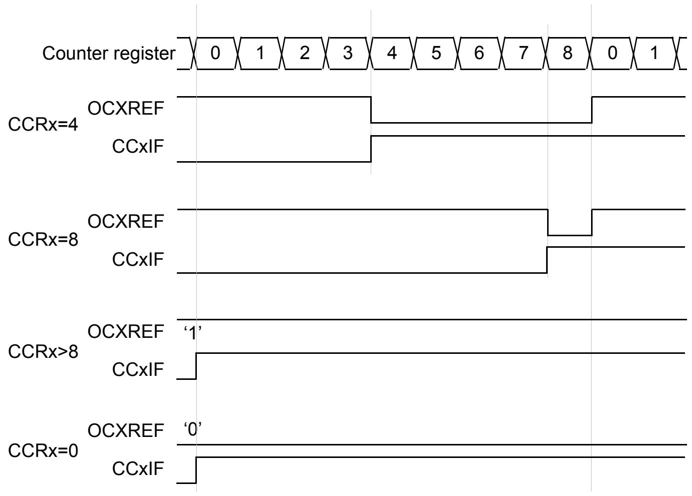
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the Up *Section 20.3.2: Counter modes*.

  In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. *Figure 110* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 110. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode on page 399*

  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

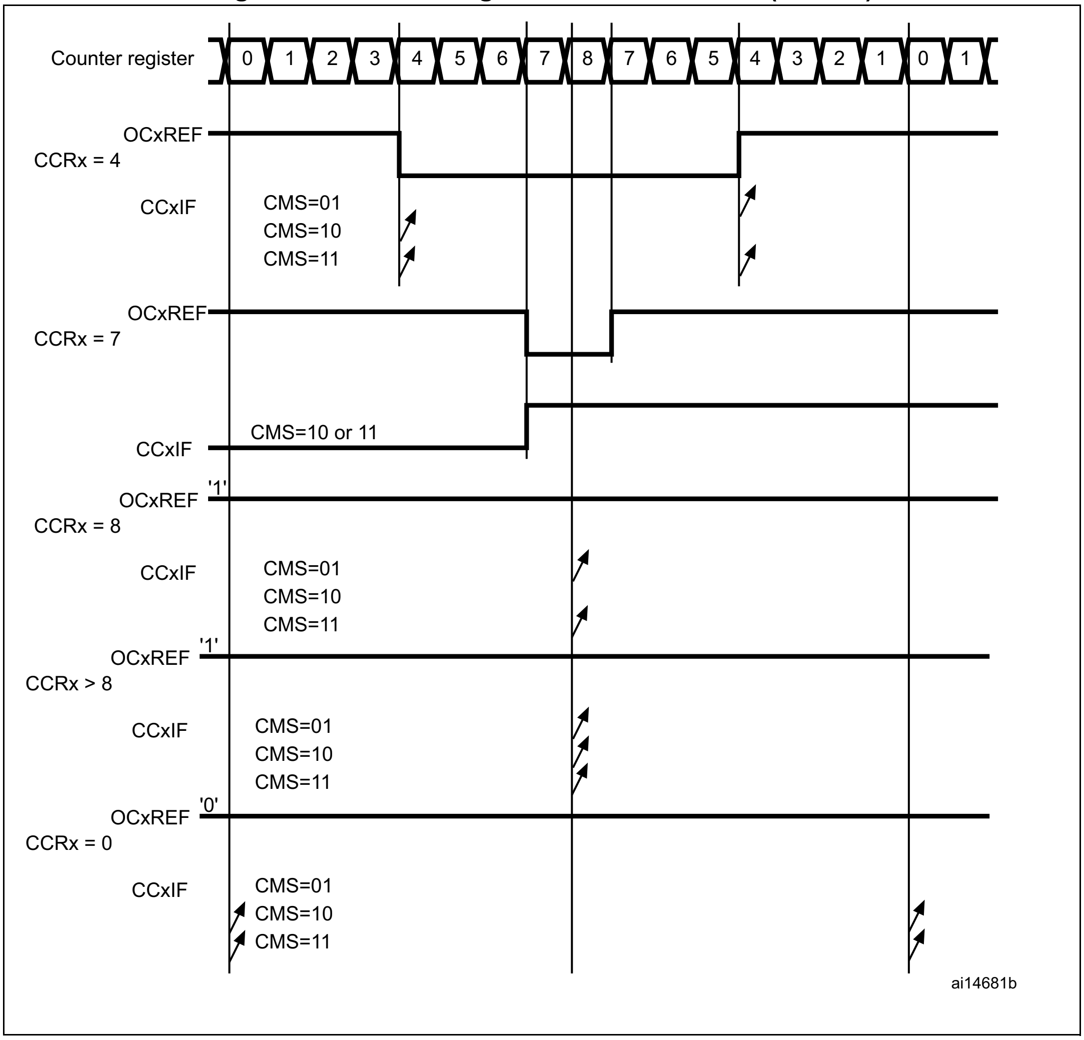
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the *Center-aligned mode (up/down counting) on page 401*.

Figure 111 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 111. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if you write 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

20.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

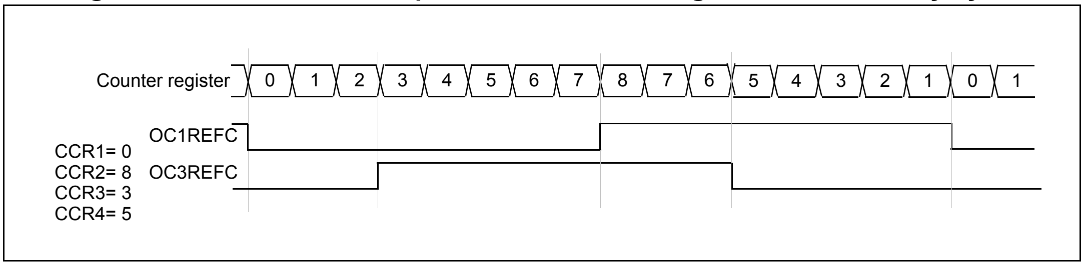
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

Figure 112 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 112. Generation of 2 phase-shifted PWM signals with 50% duty cycle



20.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

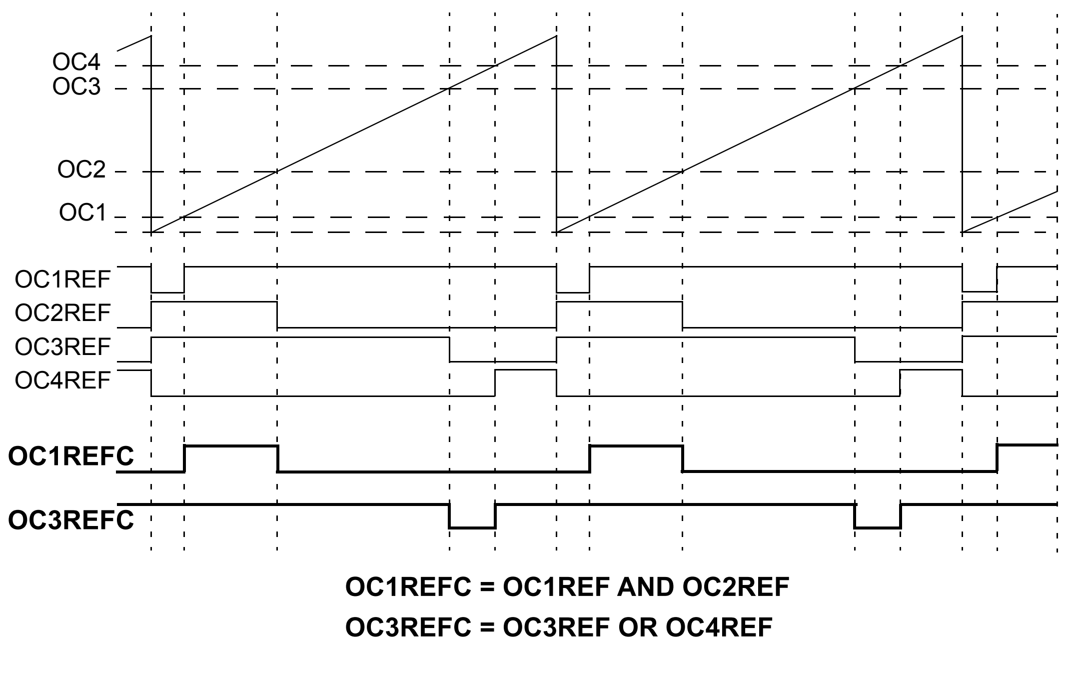
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 113 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2
- Channel 2 is configured in PWM mode 1
- Channel 3 is configured in Combined PWM mode 2
- Channel 4 is configured in PWM mode 1

Figure 113. Combined PWM mode on channel 1 and 3



20.3.14 Clearing the OCxREF signal on an external event

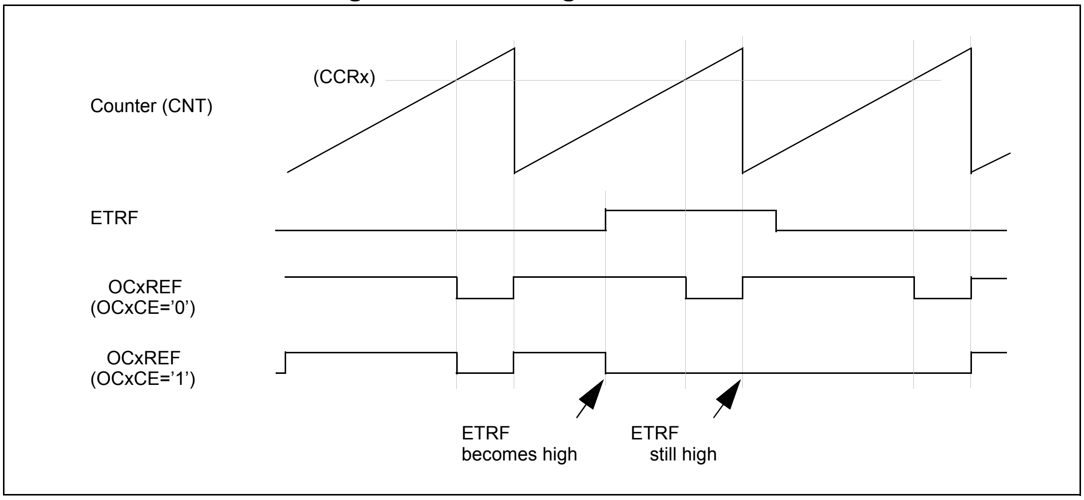
The OCxREF signal of a given channel can be cleared when a high level is applied on the ETRF input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1) if TIMx_SMCR.OCCS bit is set to 1. The OCxREF signal remains low until the next update event (UEV) occurs.

This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

Figure 114 shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 114. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.

20.3.15 One-pulse mode

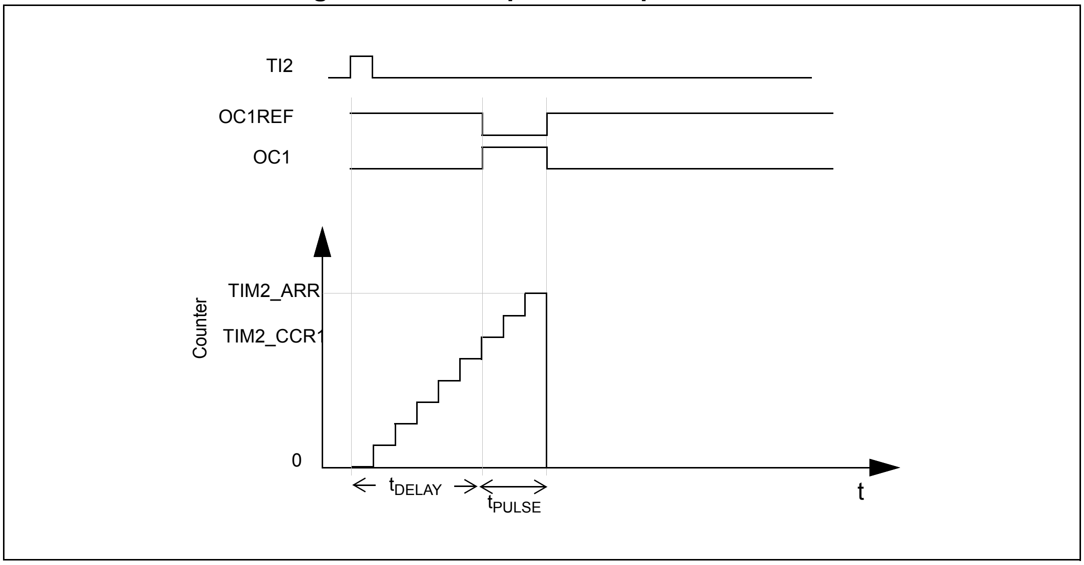
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 115. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say you want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

20.3.16 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in Section 20.3.15:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

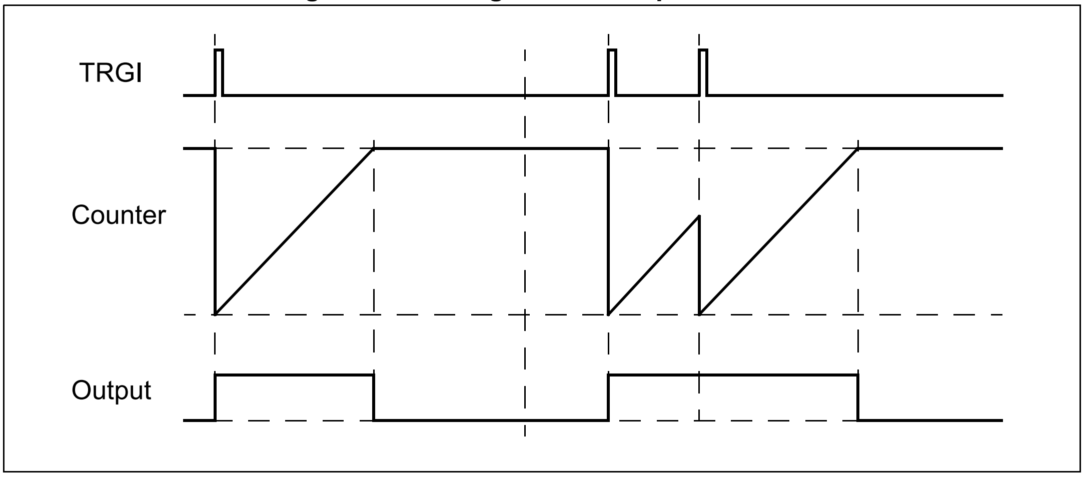
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, the ARR must be set to 0 (the CCRx register sets the pulse length).

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

In Retriggerable one pulse mode, the CCxIF flags are not significant.

Figure 116. Retrigerrable one pulse mode



20.3.17 Encoder interface mode

To select Encoder Interface mode write SMS=‘001’ in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=’010’ if it is counting on TI1 edges only and SMS=’011’ if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, you can program the input filter as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an quadrature encoder. Refer to *Table 61*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1’). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So you must configure TIMx_ARR before starting. in the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Note: The prescaler must be set to zero when encoder mode is enabled.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don’t switch at the same time.

Table 61. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic.

However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The *Figure 117* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2)
- CC1P='0' and CC1NP='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1)
- CC2P='0' and CC2NP='0' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2)
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 117. Example of counter operation in encoder interface mode.

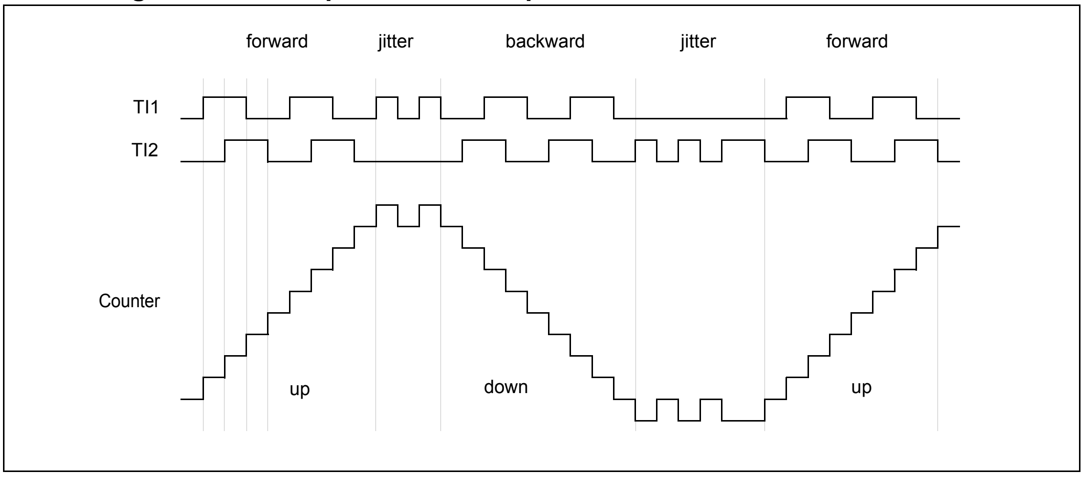
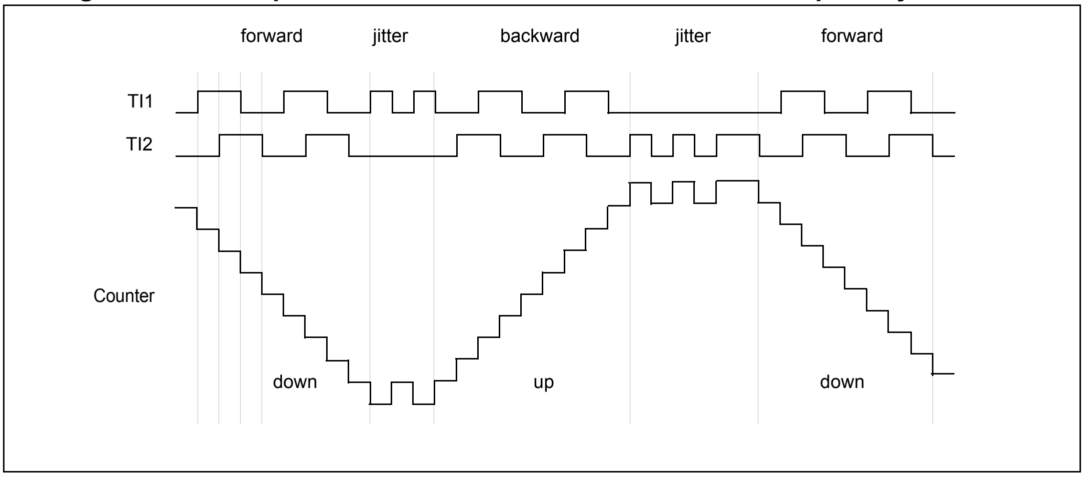


Figure 118 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 118. Example of encoder interface mode with TI1FP1 polarity inverted.



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. You can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer).

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

20.3.18 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

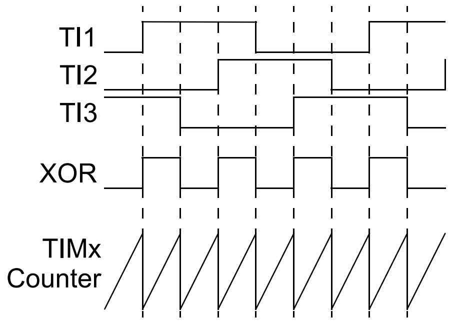
There is no latency between the UIF and UIFCPY flags assertion.

20.3.19 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 119* below.

Figure 119. Measuring time interval between edges on 3 signals



20.3.20 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

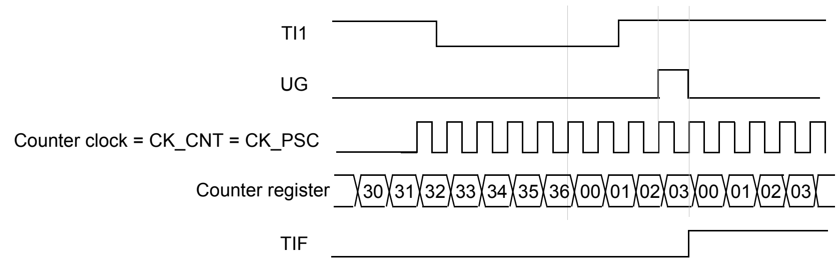
The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated. In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 120. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

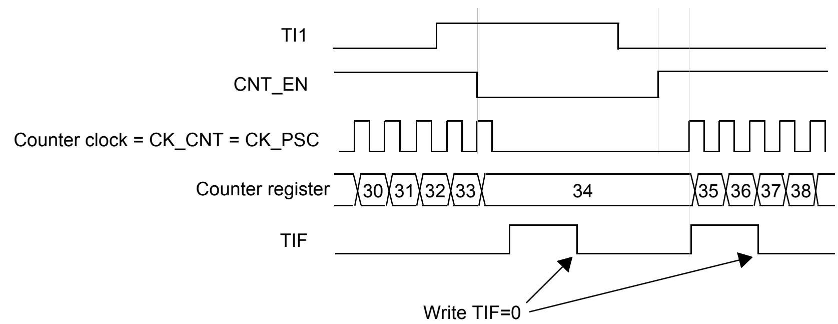
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 121. Control circuit in gated mode



1. The configuration “CCxP=CCxNP=1” (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Note: The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

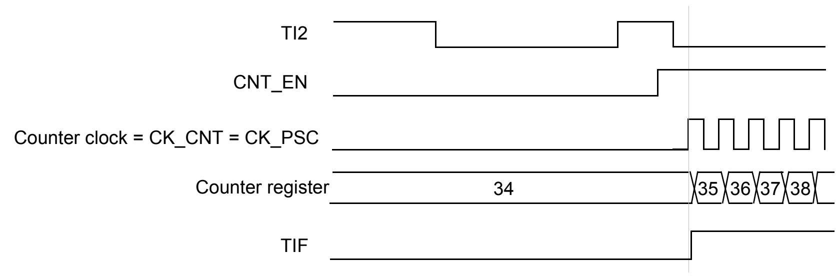
Slave mode: Trigger mode

The counter can start in response to an event on a selected input.
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.
The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 122. Control circuit in trigger mode



Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1.Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:

–ETF = 0000: no filter

–ETPS=00: prescaler disabled

–ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:

–IC1F=0000: no filter.

–The capture prescaler is not used for triggering and does not need to be configured.

–CC1S=01in TIMx_CCMR1 register to select only the input capture source

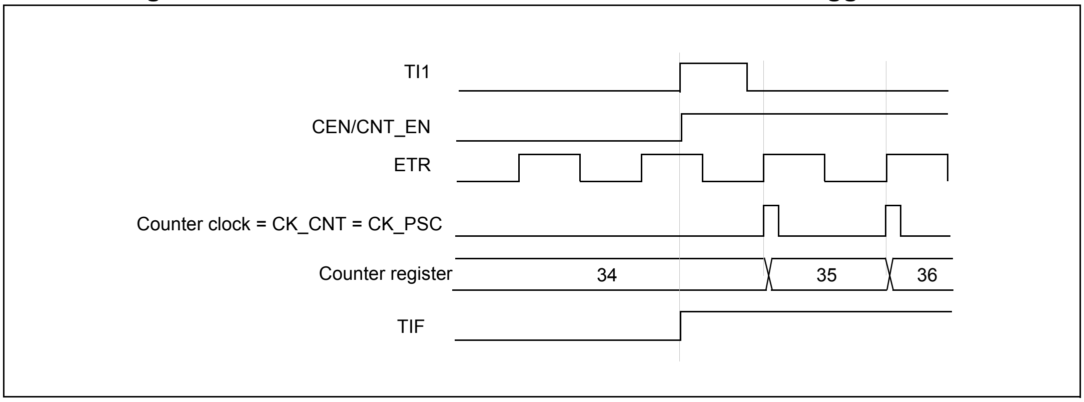
–CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 123. Control circuit in external clock mode 2 + trigger mode



Slave mode – combined reset + trigger mode

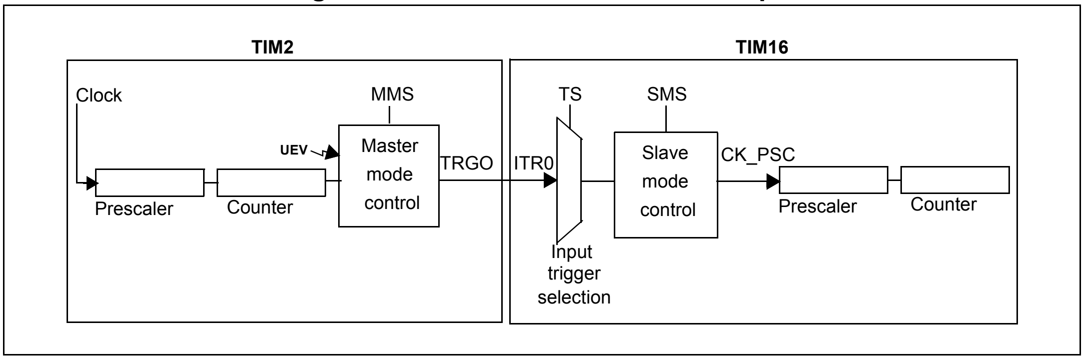
In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter. This mode is used for one-pulse mode.

20.3.21 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 124 presents an overview of the trigger selection and the master mode selection blocks.

Figure 124. Master/slave timers example



Using one timer as prescaler for another timer

For example, TIM2 can be configured to act as a prescaler for TIM16. Refer to *Figure 124*. To do this:

1. Configure TIM2 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM2_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM2 to TIM16, TIM16 must be configured in slave mode using ITR0 as internal trigger. This is selected through the TS bits in the TIM16_SMCR register (writing TS=00000).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM16_SMCR register). This causes TIM16 to be clocked by the rising edge of the periodic TIM2 trigger signal (which correspond to the TIM2 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM2 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM16.

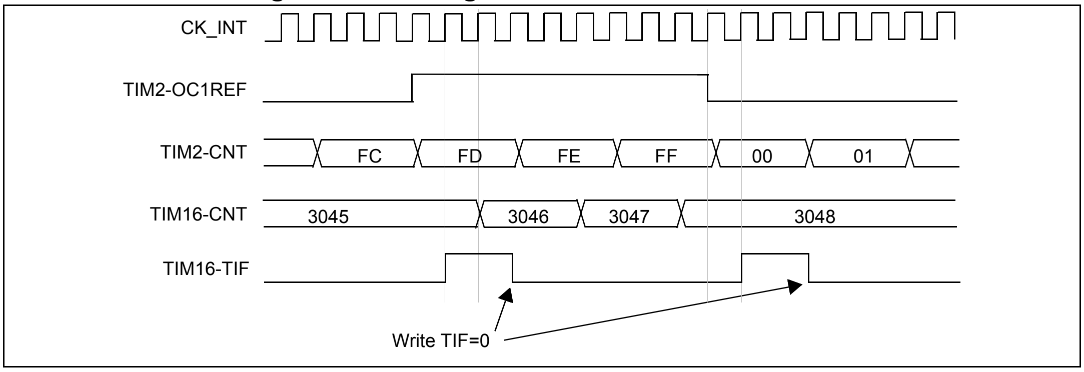
Using one timer to enable another timer

In this example, we control the enable of TIM16 with the output compare 1 of Timer 2. Refer to *Figure 124* for connections. TIM16 counts on the divided internal clock only when OC1REF of TIM2 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

1. Configure TIM2 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM2_CR2 register).
2. Configure the TIM2 OC1REF waveform (TIM2_CCMR1 register).
3. Configure TIM16 to get the input trigger from TIM2 (TS=00000 in the TIM16_SMCR register).
4. Configure TIM16 in gated mode (SMS=101 in TIM16_SMCR register).
5. Enable TIM16 by writing ‘1 in the CEN bit (TIM16_CR1 register).
6. Start TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM16 counter enable signal.

Figure 125. Gating TIM16 with OC1REF of TIM2

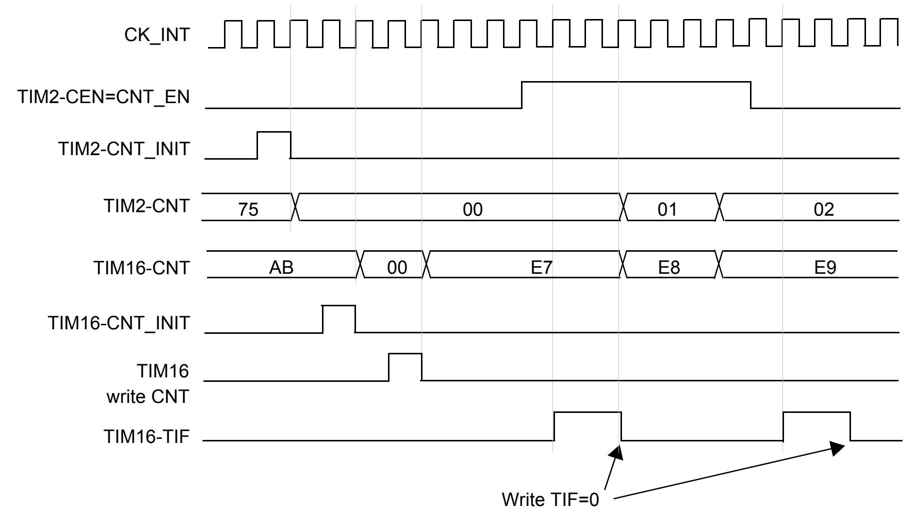


In the example in *Figure 125* , the TIM16 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM2. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 126*), we synchronize TIM2 and TIM16. TIM2 is the master and starts from 0. TIM16 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM16 stops when TIM2 is disabled by writing ‘0 to the CEN bit in the TIM2_CR1 register:

1. Configure TIM2 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM2_CR2 register).
2. Configure the TIM2 OC1REF waveform (TIM2_CCMR1 register).
3. Configure TIM16 to get the input trigger from TIM2 (TS=00000 in the TIM16_SMCR register).
4. Configure TIM16 in gated mode (SMS=101 in TIM16_SMCR register).
5. Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
6. Reset TIM16 by writing ‘1 in UG bit (TIM16_EGR register).
7. Initialize TIM16 to 0xE7 by writing ‘0xE7’ in the TIM16 counter (TIM16_CNTL).
8. Enable TIM16 by writing ‘1 in the CEN bit (TIM16_CR1 register).
9. Start TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
10. Stop TIM2 by writing ‘0 in the CEN bit (TIM2_CR1 register).

Figure 126. Gating TIM16 with Enable of TIM2

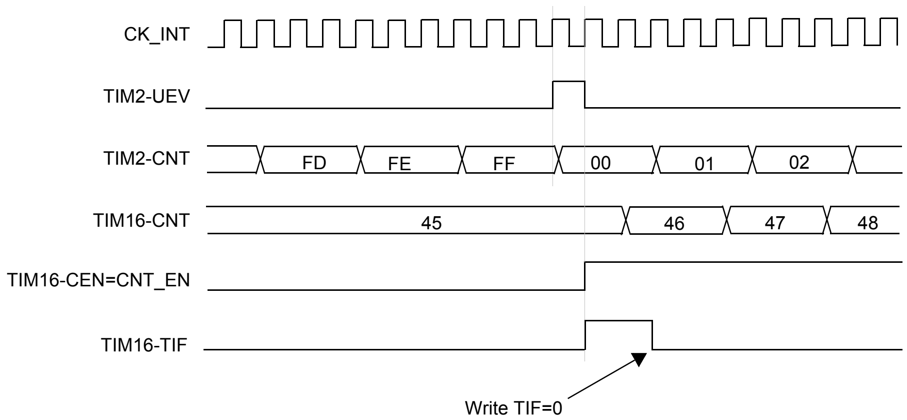


Using one timer to start another timer

In this example, we set the enable of Timer 16 with the update event of Timer 2. Refer to *Figure 124* for connections. Timer 16starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 2. When Timer 16 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM16_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

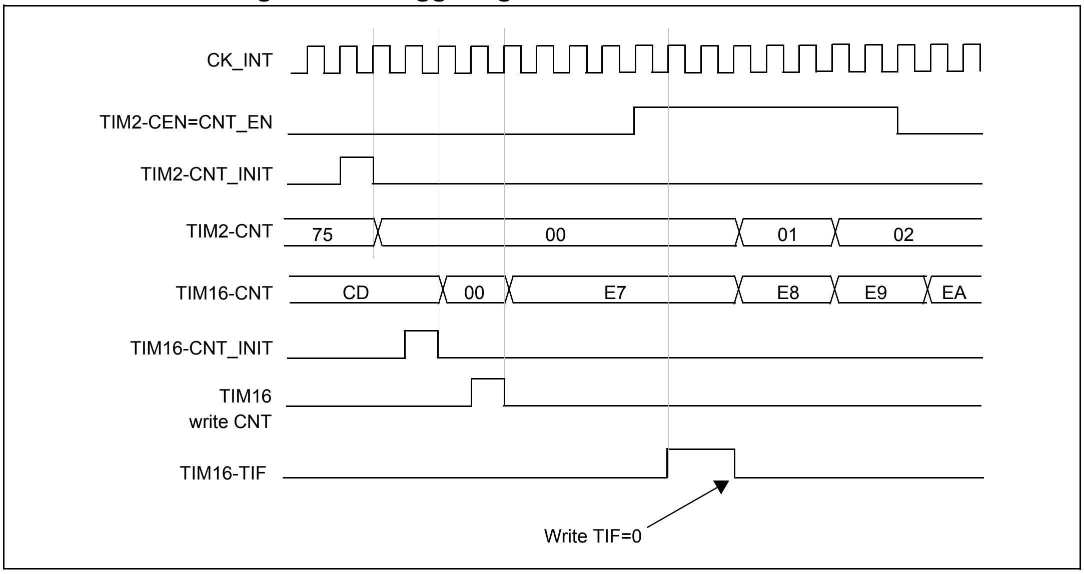
1. Configure TIM2 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM2_CR2 register).
2. Configure the TIM2 period (TIM2_ARR registers).
3. Configure TIM16 to get the input trigger from TIM2 (TS=00000 in the TIM16_SMCR register).
4. Configure TIM16 in trigger mode (SMS=110 in TIM16_SMCR register).
5. Start TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).

Figure 127. Triggering TIM16 with update of TIM2



As in the previous example, both counters can be initialized before starting counting. *Figure 128* shows the behavior with the same configuration as in *Figure 127* but in trigger mode instead of gated mode (SMS=110 in the TIM16_SMCR register).

Figure 128. Triggering TIM16 with Enable of TIM2



Note: The clock of the slave peripherals (timer, ...) receiving the TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

20.3.22 DMA burst mode

The TIMx timer has the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers. The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:

– DMA channel peripheral address is the DMAR register address

– DMA channel memory address is the address of the buffer in the RAM containing

the data to be transferred by DMA into CCRx registers.

– Number of data to transfer = 3 (See note below).

– Circular mode disabled.

2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:

DBL = 3 transfers, DBA = 0xE.

3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).

4. Enable TIMx

5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

20.3.23 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIM2_STOP configuration bit in DBGMCU module. For more details, refer to *Section 30.3.2: DBG APB0 freeze register (DBG_APB0_FZ)*.

20.4 TIM2 registers

Refer to for a list of abbreviations used in register descriptions.

20.4.1 TIM2 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, always read as 0

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10 Reserved, always read as 0

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx),
00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

20.4.2 TIM2 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw				

Bits 31:8 Reserved, always read as 0

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 Reserved, always read as 0

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)
100: **Compare** - OC1REF signal is used as trigger output (TRGO)
101: **Compare** - OC2REF signal is used as trigger output (TRGO)
110: **Compare** - OC3REF signal is used as trigger output (TRGO)
111: **Compare** - OC4REF signal is used as trigger output (TRGO)

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2:0 Reserved, always read as 0

20.4.3 TIM2 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

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

Bits 31:22 Reserved, always read as 0

Bits 19:17 Reserved, always read as 0

Bit 16 **SMS[3]**: Slave mode selection - bit 3
Refer to SMS description - bits2:0

Bit 15 **ETP**: External trigger polarity
This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

Note: ***1:*** *Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).*

2: *It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).*

3: *If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler
External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.
00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/Slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21:20, Bits 6:4 **TS[4:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (ITR0)

00001: Internal Trigger 1 (ITR1)

00010: Internal Trigger 2 (ITR2)

00011: Internal Trigger 3 (ITR3)

00100: TI1 Edge Detector (TI1F_ED)

00101: Filtered Timer Input 1 (TI1FP1)

00110: Filtered Timer Input 2 (TI2FP2)

00111: External Trigger input (ETRF)

Others: Reserved

See *TIM2 internal trigger connection* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.
0: OCREF_CLR_INT is connected to the OCREF_CLR input (stuck at 0 so no effect)
1: OCREF_CLR_INT is connected to ETRF

Bit 16, Bits 2:0 **SMS:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Others: Reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=’100’). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.The clock of the slave peripherals (timer2) receiving the TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: TIM2 internal trigger connection

Slave TIM	ITR0	ITR1	ITR2 - ITR8
TIM2	TIM16	-	-

20.4.4 TIM2 DMA/interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
			rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 15, 13 Reserved, always read as 0.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 Reserved, always read as 0.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 Reserved, always read as 0.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

20.4.5 TIM2 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:13 Reserved, always read as 0.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8:7 Reserved, always read as 0.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, always read as 0.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register.

When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by a trigger event (refer to *Section 20.4.3: TIM2 slave mode control register (TIMx_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

20.4.6 TIM2 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, always read as 0.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt can occur if enabled.

Bits 5 Reserved, always read as 0.

Bit 4 **CC4G**: Capture/Compare 4 generation

refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation

refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

20.4.7 TIM2 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							Res.								Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bits16 **OC1M[3]**: Output Compare 1 mode - bit 3
Refer to OC1M description on bits 6:4

Bit 15 **OC2CE:** Output Compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = ‘0’ in TIMx_CCER).

Bit 7 **OC1CE:** Output Compare 1 clear enable
OC1CE: Output Compare 1 Clear Enable
0: OC1Ref is not affected by the ETRF Input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.
0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
0100: Force inactive level - OC1REF is forced low.
0101: Force active level - OC1REF is forced high.
0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0’) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=’1’).
0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.
1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.
1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.
1010: Reserved,
1011: Reserved,
1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.
1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.
1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.
1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: ***1:*** *These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).*

2: *In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

20.4.8 TIM2 capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							Res.								Res.
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC4M[3]**: Output Compare 4 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bit 16 **OC3M[3]**: Output Compare 3 mode - bit 3

Bit 15 OC4CE: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = ‘0’ in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = ‘0’ in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = ‘0’ in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = ‘0’ in TIMx_CCER).

20.4.9 TIM2 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC4P	CC4E	Res.	Res.	CC3P	CC3E	Res.	Res.	CC2P	CC2E	Res.	Res.	CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

Bits 31:14 Reserved, always read as 0.

Bit 13 **CC4P**: Capture/Compare 4 output polarity
refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable
refer to CC1E description

Bit 11:10 Reserved, always read as 0.

Bit 9 **CC3P**: Capture/Compare 3 output polarity
refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
refer to CC1E description

Bit 7:6 Reserved, always read as 0.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
refer to CC1E description

Bit 3:2 Reserved, always read as 0.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:
0: OC1 active high
1: OC1 active low

CC1 channel configured as input:
CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

01: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

10: reserved, do not use this configuration.

11: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:
0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 62. Output control bits for OCx channels

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z) OCx=0
1	OCxREF + Polarity OCx=OCxREF xor CCxP

20.4.10 TIM2 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, always read as 0.

Bits 15:0 **CNT[15:0]**: Counter value

20.4.11 TIM2 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

20.4.12 TIM2 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 20.3.1: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

20.4.13 TIM2 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

the number of PWM periods in edge-aligned mode

the number of half PWM period in center-aligned mode.

20.4.14 TIM2 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

20.4.15 TIM2 capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

20.4.16 TIM2 capture/compare register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (IC3).

20.4.17 TIM2 capture/compare register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC4 output.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (IC4).

20.4.18 TIM2 DMA control register (TIM2_DCR)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,

00001: 2 transfers,

00010: 3 transfers,

...

10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: Reserved,

...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

20.4.19 TIM2 DMA address for full transfer (TIM2_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

20.4.20 TIM2 option register 1 (TIM2_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI4_RMP	Res.	ETR_RMP
													rw		rw

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

Bit 2 **TI4_RMP**: Input capture 4 remap

0: TIM2 input capture 4 is connected to I/O

1: TIM2 input capture 4 is connected to COMP1_OUT (STM32WL33xx only)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ETR_RMP**: ETR remapping capability

0: TIMx_ETR is not connected to ADC[1] AWD (must be selected when the ETR comes from the ETR input pin)

1: TIMx_ETR is connected to ADC[1] AWD

Note: ADC AWD source is 'ORed' with the TIMx_ETR input signals. When ADC AWD is used, it is necessary to make sure that the corresponding TIMx_ETR input pin is not enabled in the alternate function controller.

1. STM32WL33xx/31xx only

20.4.21 TIM2 alternate function option register 1 (TIM2_AF1)

Address offset: 0x60

Reset value: 0x0000

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

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

Bits 16:14 **ETRSEL[2:0]**: External trigger source selection

000: TIMx External trigger legacy mode

001: TIMx External trigger source select COMP1_OUT (STM32WL33xx only)

Other: Reserved

Note: These bits can't be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

Bits 13:0 Reserved, must be kept at reset value.

20.4.22 TIM2 input selection register (TIM2_TISEL)

Address offset: 0x68

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

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

Bits 27:24 **TI4SEL[3:0]**: selects TI4[0] to TI4[15] input
0000: TIMx_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: selects TI3[0] to TI3[15] input
0000: TIMx_CH3 input
Others: Reserved

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

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input
0000: TIMx_CH2 input
Others: Reserved

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIMx_CH1 input
Others: Reserved

20.5 TIM2 register map

TIM2 registers are mapped as 16-bit addressable registers as described in the table below:

Table 63. TIM2 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res		TI1S	MMS[2:0]			CCDS	Res	Res	Res
	Reset value																									0	0	0	0	0			
0x08	TIMx_SMCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TS[4:3]		Res	Res	Res	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value											0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	TIMx_DIER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TDE	Res	CC1DE	CC2DE	CC3DE	CC4DE	UDE	Res	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0		0	0	0	0	0		0		0	0	0	0	0
0x10	TIMx_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC4OF	CC3OF	CC2OF	CC1OF	Res	Res	TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	TIMx_EGR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TG	Res	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	TIMx_CCMR1 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC2M[3]	Res	Res	Res	Res	Res	Res	Res	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIMx_CCMR2 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC4M[3]	Res	Res	Res	Res	Res	Res	Res	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC4P	CC4E	Res	Res	CC3P	CC3E	Res	Res	CC2P	CC2E	Res	Res	CC1P	CC1E
	Reset value																			0	0			0	0			0	0			0	0
0x24	TIMx_CNT	UIFCPY	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63. TIM2 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TIMx_ARR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIMx_RCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBL[4:0]					Res	Res	Res	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIMx_DMAR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TIMx_OR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI4_RMP	Res	ETR_RMP
	Reset value																														0	Res	0
0x60	TIMx_AF1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ETRSEL[2:0]			Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																0	0	0														
0x68	TIMx_TISEL	Res	Res	Res	Res	TI4SEL[3:0]				Res	Res	Res	Res	Res	TI3SEL[3:0]				Res	Res	Res	TI2SEL[3:0]				Res	Res	Res	Res	TI1SEL[3:0]			
	Reset value					0	0	0	0						0	0	0	0				0	0	0	0					0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

21 General-purpose timers (TIM16)

Note: 1 In this section, “TIMx” should be understood as “TIM16” since there is only one instance of this timer in STM32WL30xx/31xx/33xx devices.

2 The DAC is available only on STM32WL33xx devices. Any reference to the DAC IP in this chapter is valid only for STM32WL33xx devices.

21.1 TIM16 introduction

The TIM16 timer consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion), but it is also specifically used to drive the digital-to-analog converter (DAC). In fact, the timer is internally connected to the DAC and is able to drive it through its trigger output.

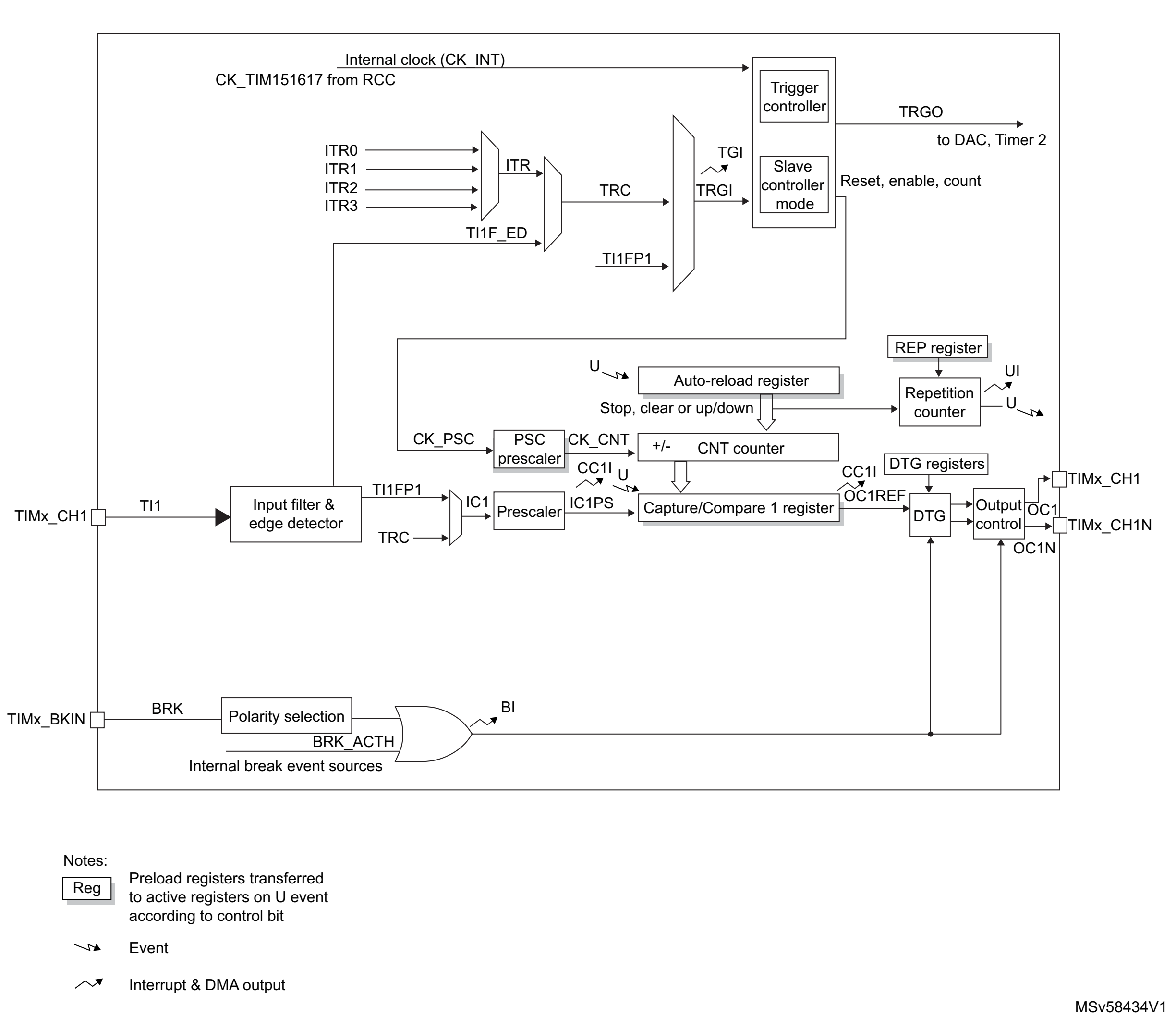
Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

21.2 TIM16 main features

The TIM16 timer includes the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary output with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
- Synchronization circuit to trigger the DAC

Figure 129. TIM16 block diagram



21.3 TIM16 functional description

21.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 130 and *Figure 131* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 130. Counter timing diagram with prescaler division change from 1 to 2

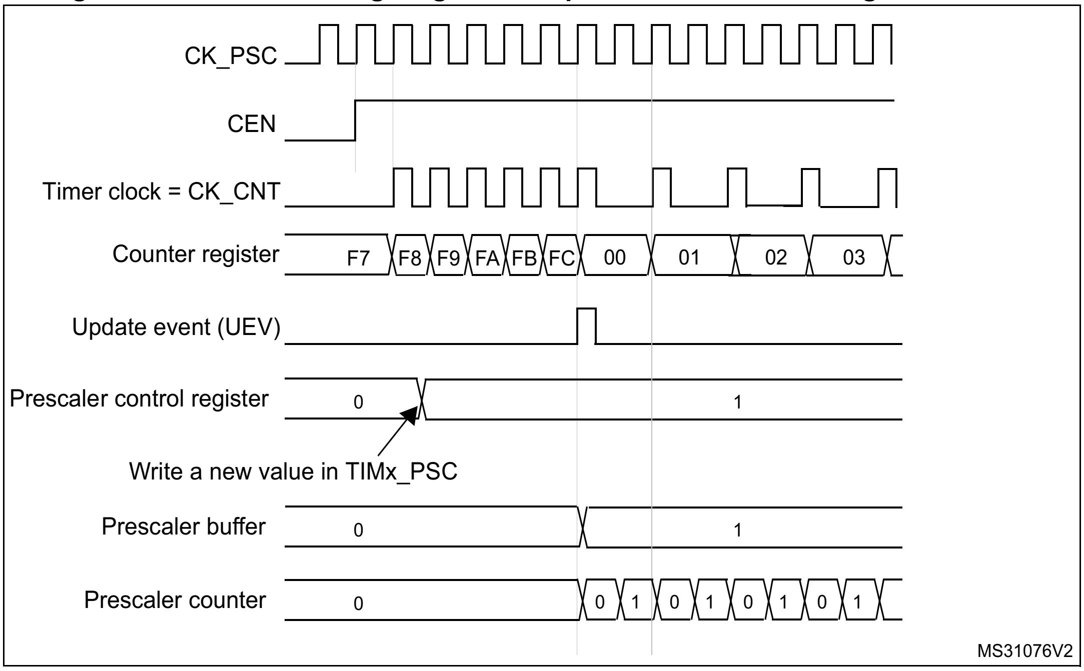
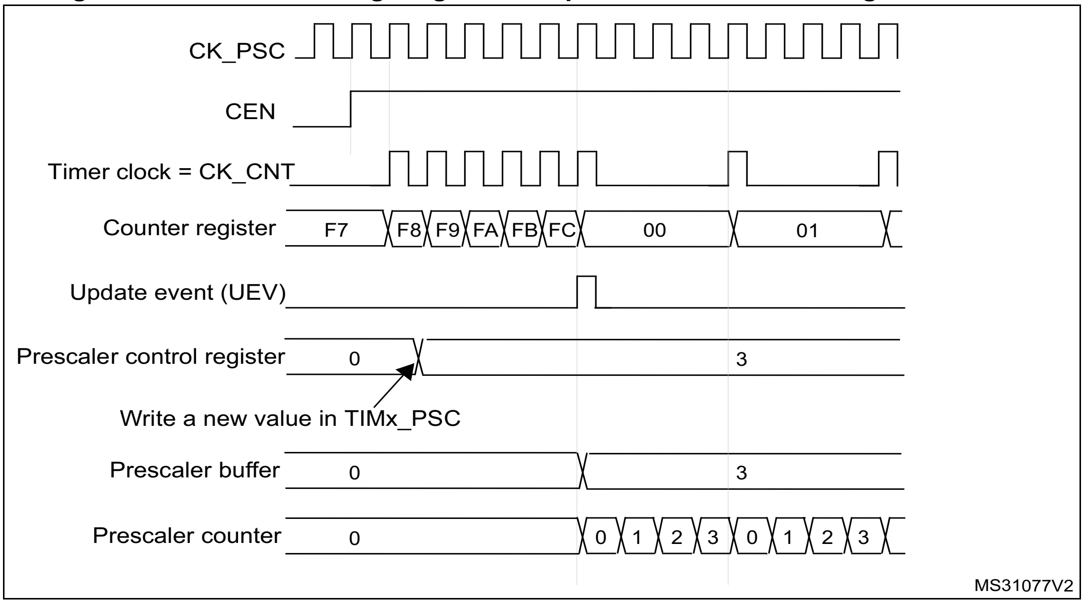


Figure 131. Counter timing diagram with prescaler division change from 1 to 4



21.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 132. Counter timing diagram, internal clock divided by 1

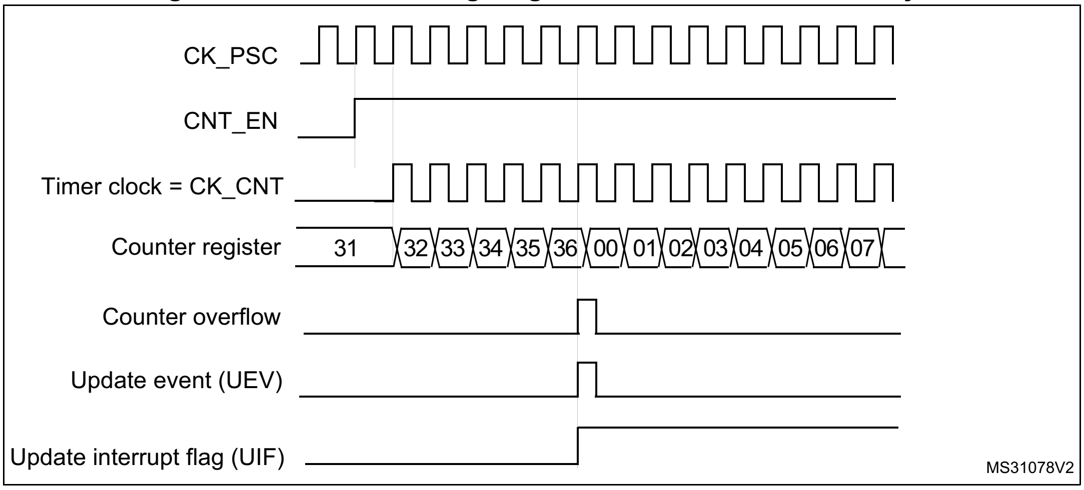


Figure 133. Counter timing diagram, internal clock divided by 2

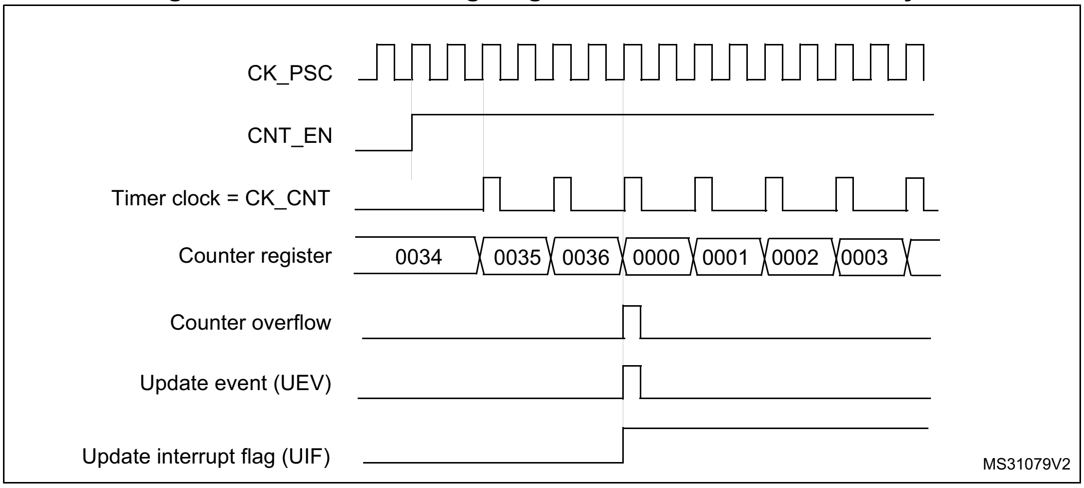


Figure 134. Counter timing diagram, internal clock divided by 4

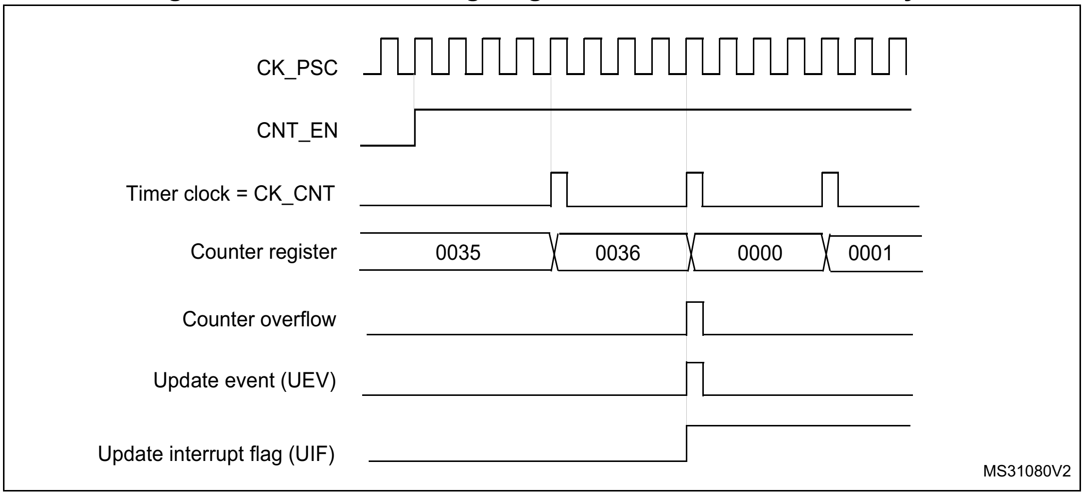


Figure 135. Counter timing diagram, internal clock divided by N

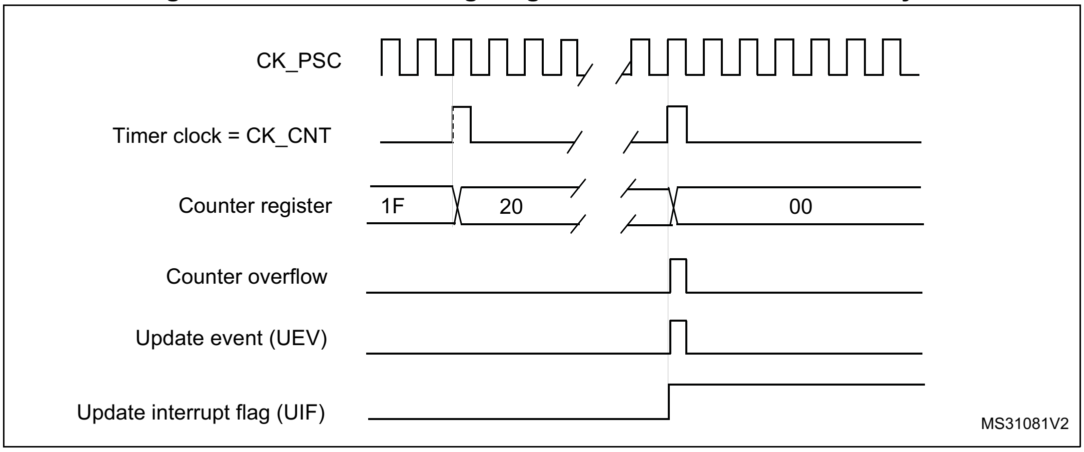


Figure 136. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

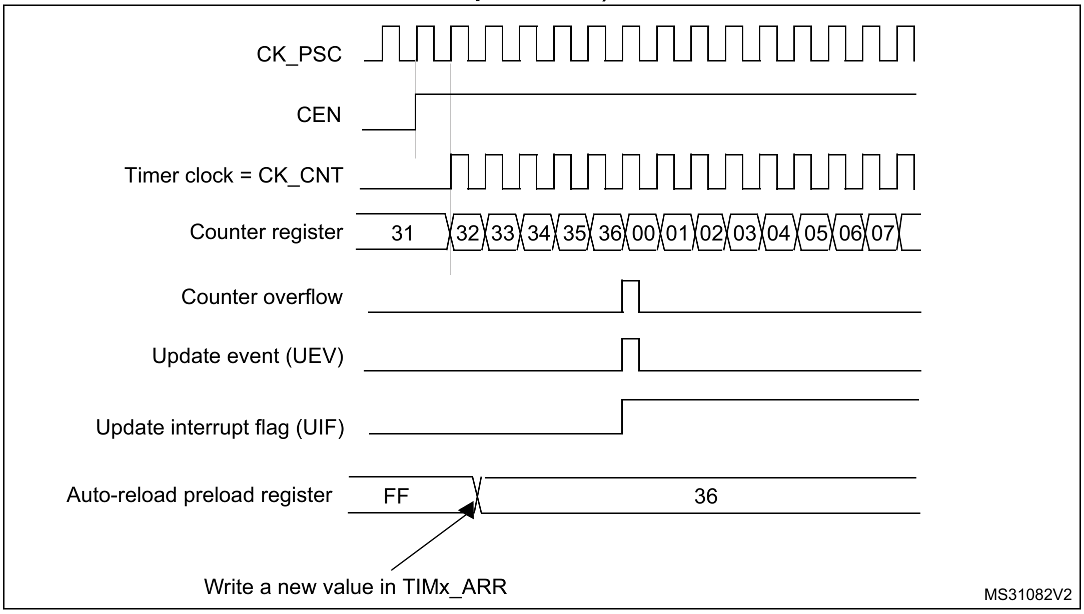
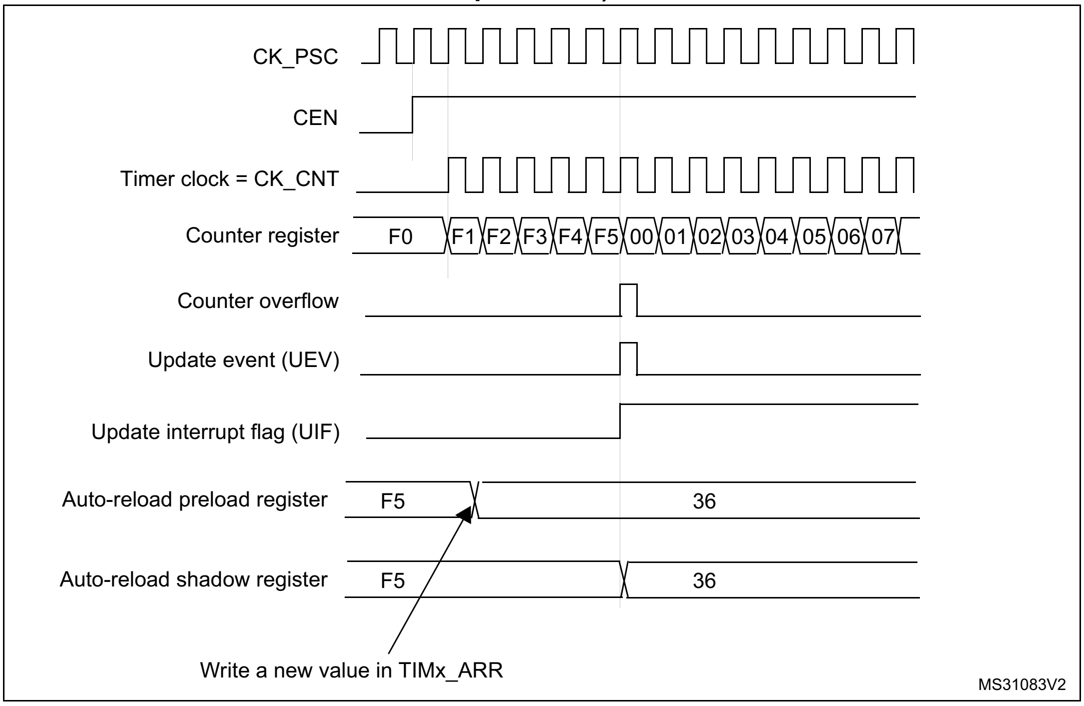


Figure 137. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



21.3.3 Repetition counter

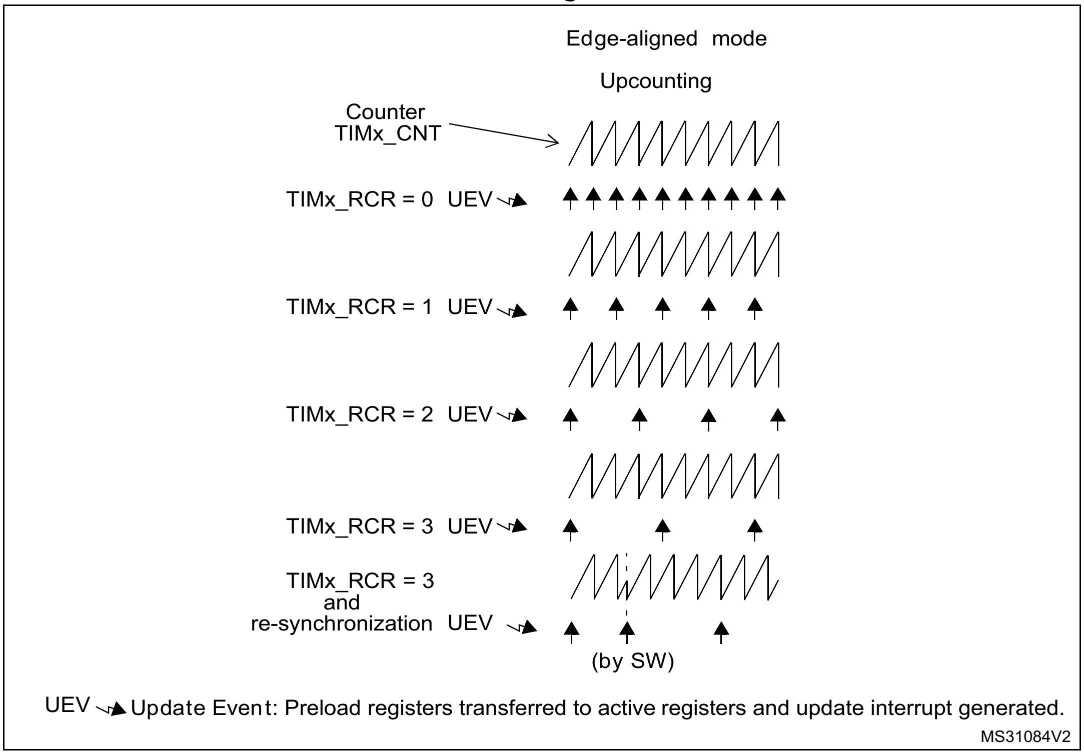
Section 21.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 138*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 138. Update rate examples depending on mode and TIMx_RCR register settings



21.3.4 Clock selection

The counter clock can be provided by the following clock sources:

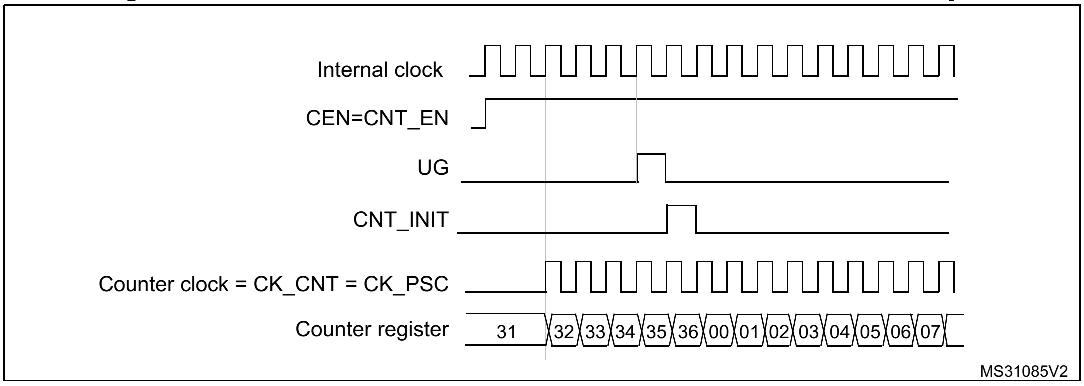
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)

Internal clock source (CK_INT)

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 139 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

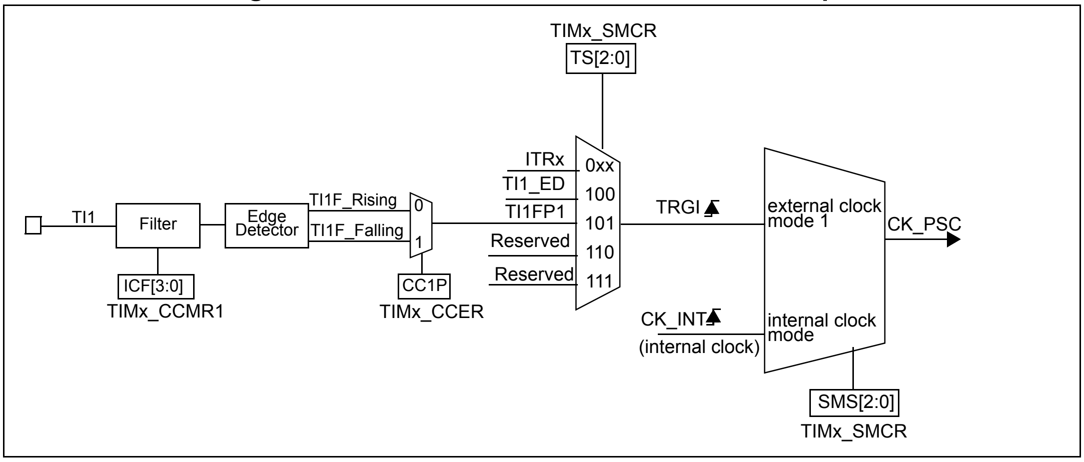
Figure 139. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 140. TI1 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI1 input, use the following procedure:

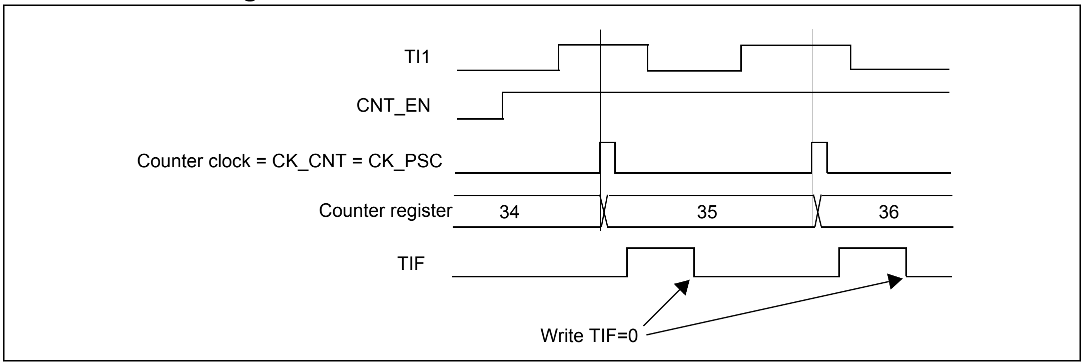
1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 1to detect rising edges on the TI1 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC1F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC1F=0000).
4. Select rising edge polarity by writing CC1P=0 and CC1NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI1 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don’t need to configure it.

When a rising edge occurs on TI1, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI1 and the actual clock of the counter is due to the resynchronization circuit on TI1 input.

Figure 141. Control circuit in external clock mode 1



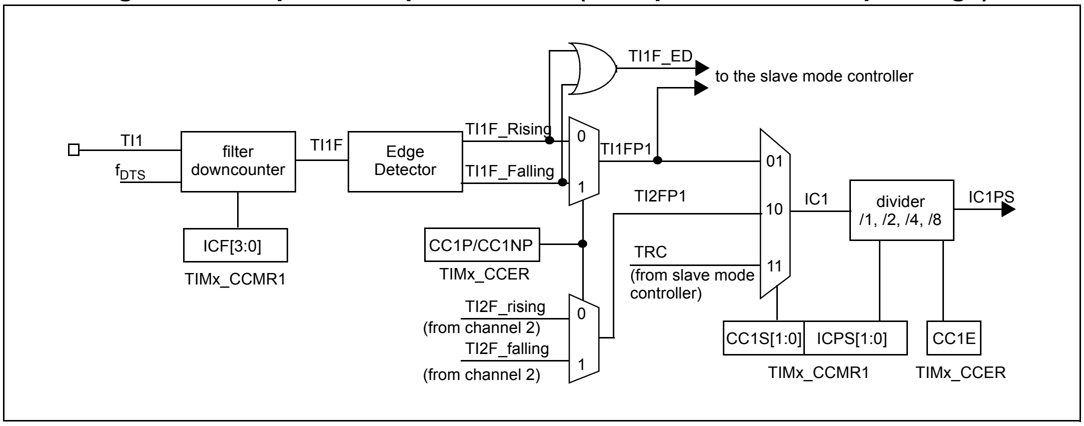
21.3.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 142 to *Figure 144* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 142. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 143. Capture/compare channel 1 main circuit

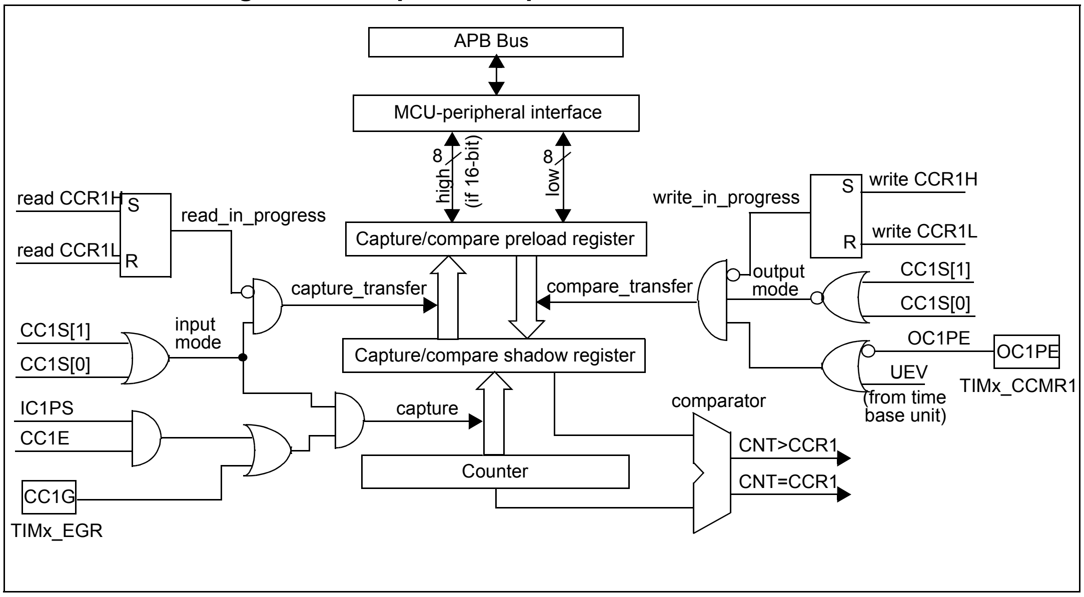
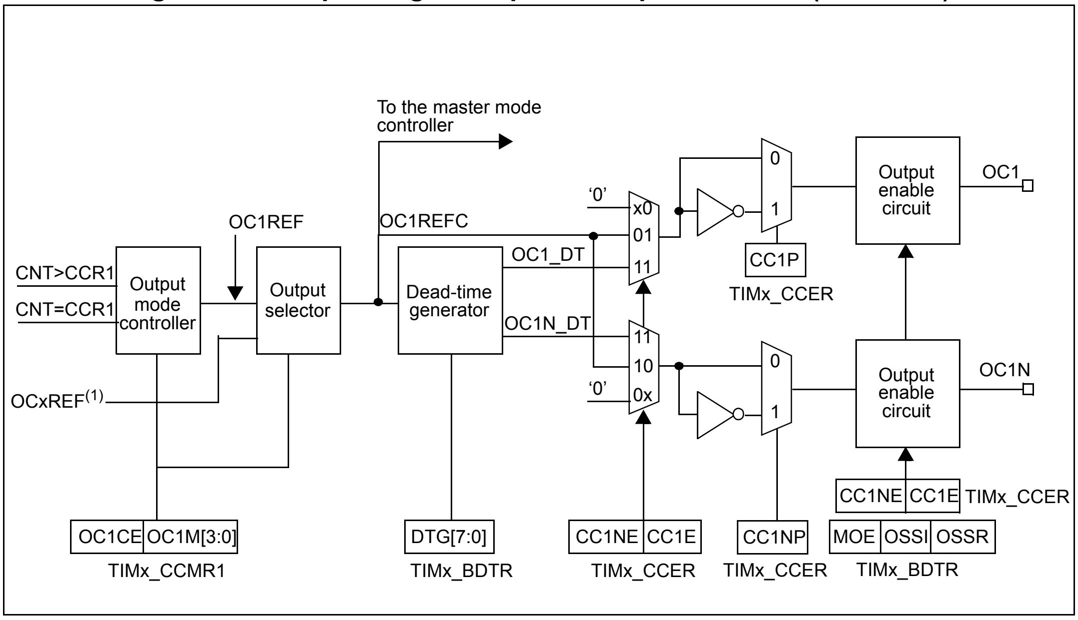


Figure 144. Output stage of capture/compare channel (channel 1)



1. OCxREF, where x is the rank of the complementary channel

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

21.3.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

3. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

21.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

21.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

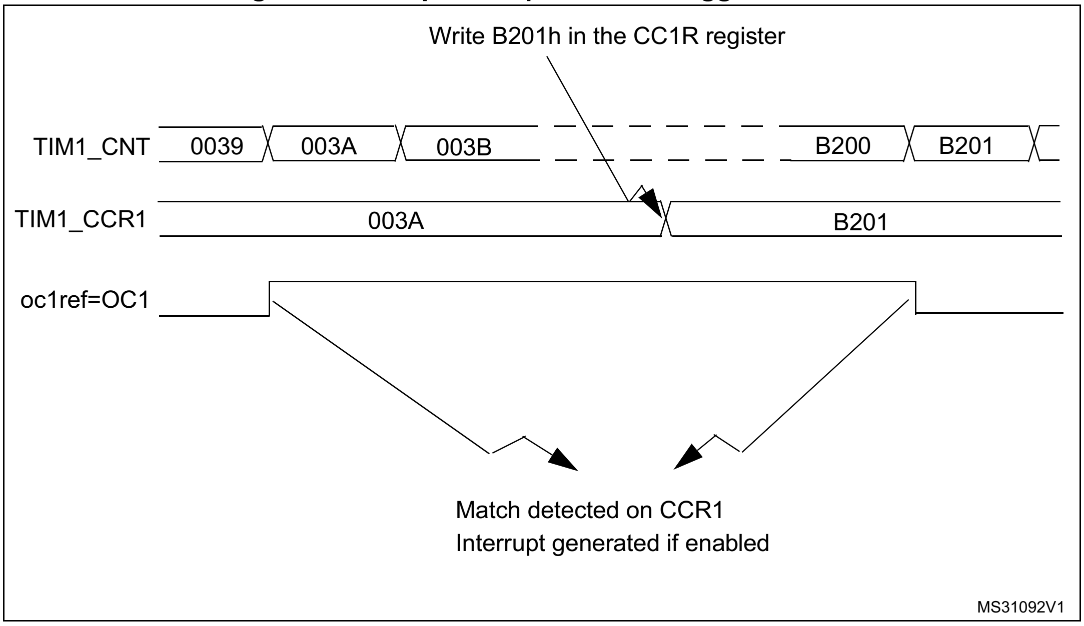
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   – Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   – Write OCxPE = 0 to disable preload register
   – Write CCxP = 0 to select active high polarity
   – Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 145*.

Figure 145. Output compare mode, toggle on OC1



21.3.9 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

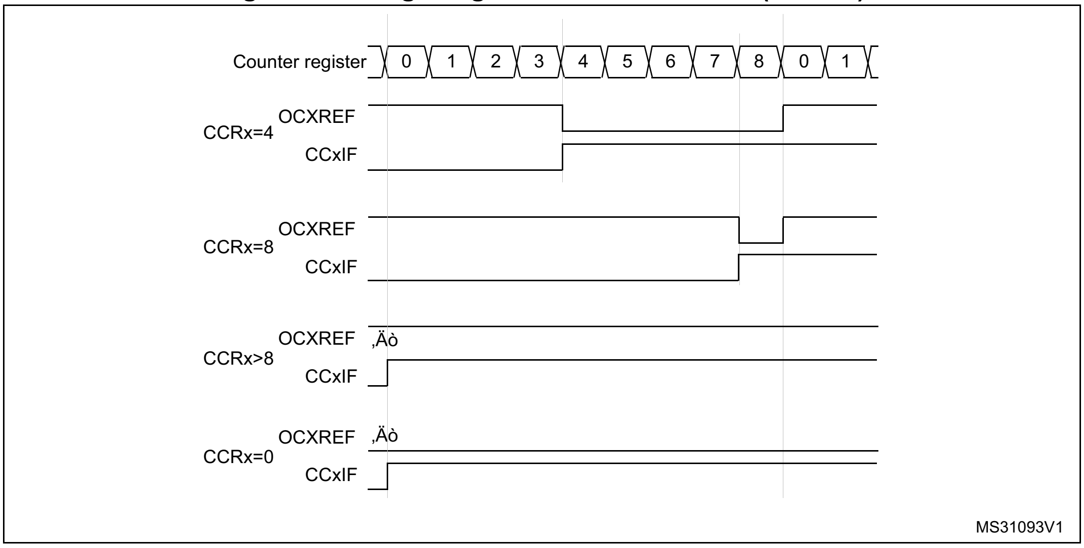
OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤TIMx_CNT or TIMx_CNT ≤TIMx_CCRx (depending on the direction of the counter).

The TIM16 is capable of upcounting only. Refer to *Upcounting mode*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at '1'. If the compare value is 0 then OCxRef is held at '0'. *Figure 146* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 146. Edge-aligned PWM waveforms (ARR=8)



21.3.10 Complementary outputs and dead-time insertion

The TIM16 general-purpose timer can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 66* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 147. Complementary output with dead-time insertion.

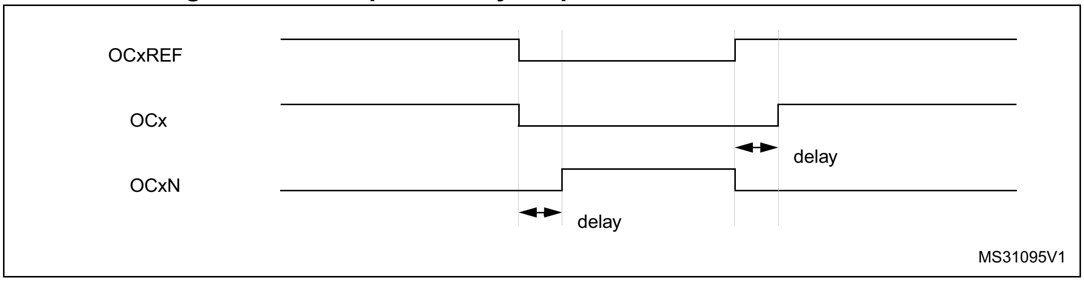


Figure 148. Dead-time waveforms with delay greater than the negative pulse.

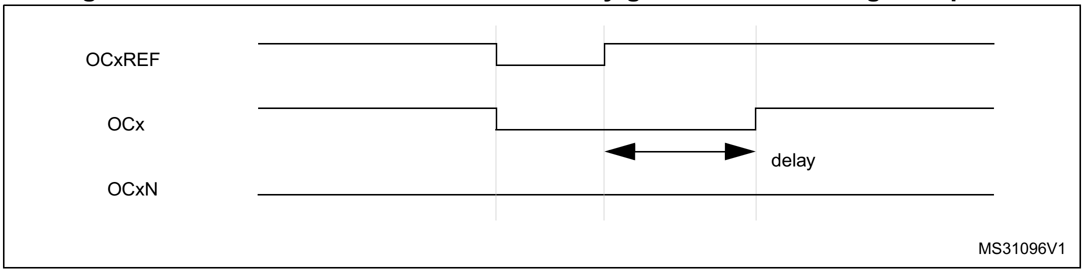
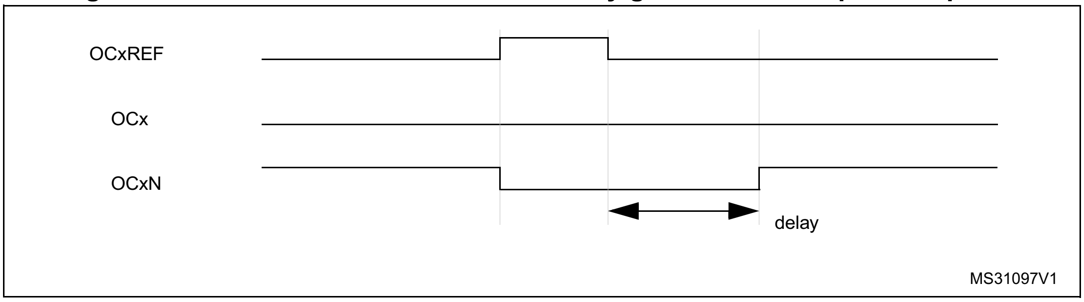


Figure 149. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 21.4.14: TIM16 break and dead-time register (TIMx_BDTR)* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

21.3.11 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM16 timer. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to *Table 66* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break function by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

The break is generated by the BRK inputs which has:

- Programmable polarity (BKP bit in the TIMx_BDTR register)
- Programmable enable bit (BKE bit in the TIMx_BDTR register)
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the AFIO controller (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the AFIO controller) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their

active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).

– If OSSI=0 then the timer releases the enable outputs (taken over by the AFIO controller which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until you write it to ‘1’ again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

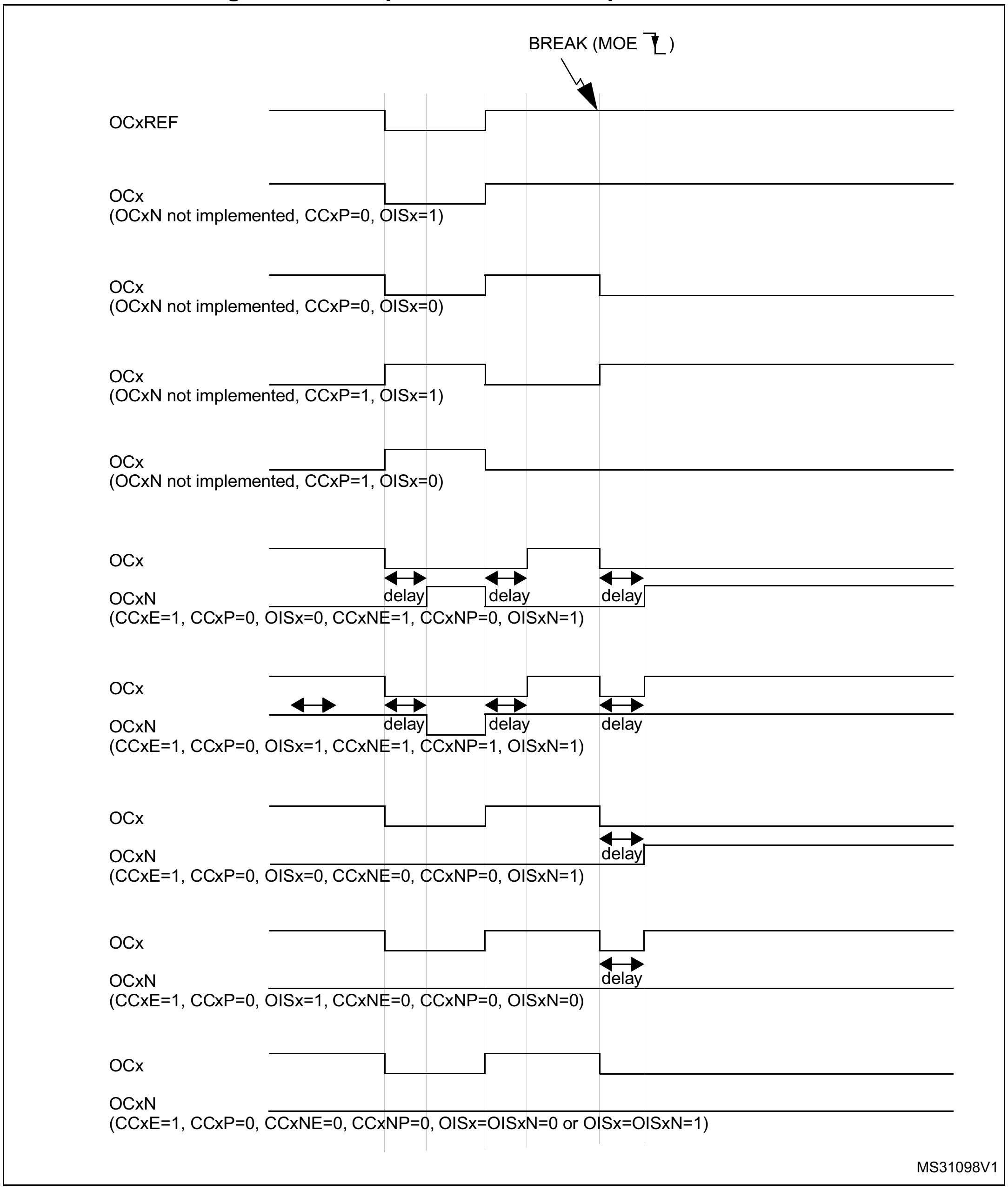
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows you to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). You can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *TIM16 break and dead-time register (TIMx_BDTR)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 150* shows an example of behavior of the outputs in response to a break.

Figure 150. Output behavior in response to a break



21.3.12 Bidirectional break inputs

The TIM16 is featuring bidirectional break I/Os, as represented on *Figure 151*.
They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The break input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (BG) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 64*)

Table 64. Break protection disarming conditions

MODE	BKDIR	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

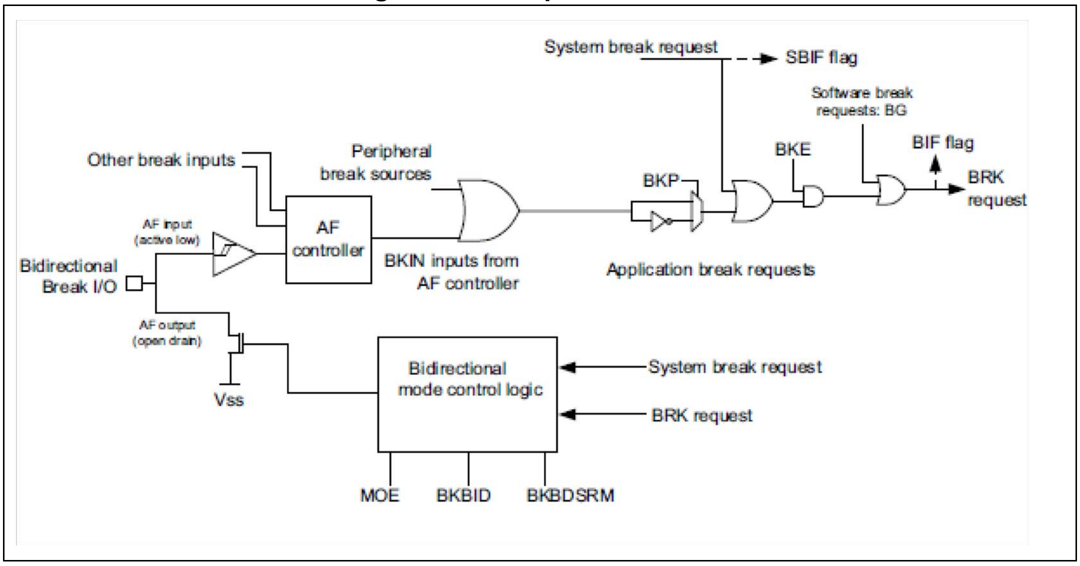
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 151. Output redirection



21.3.13 One-pulse mode

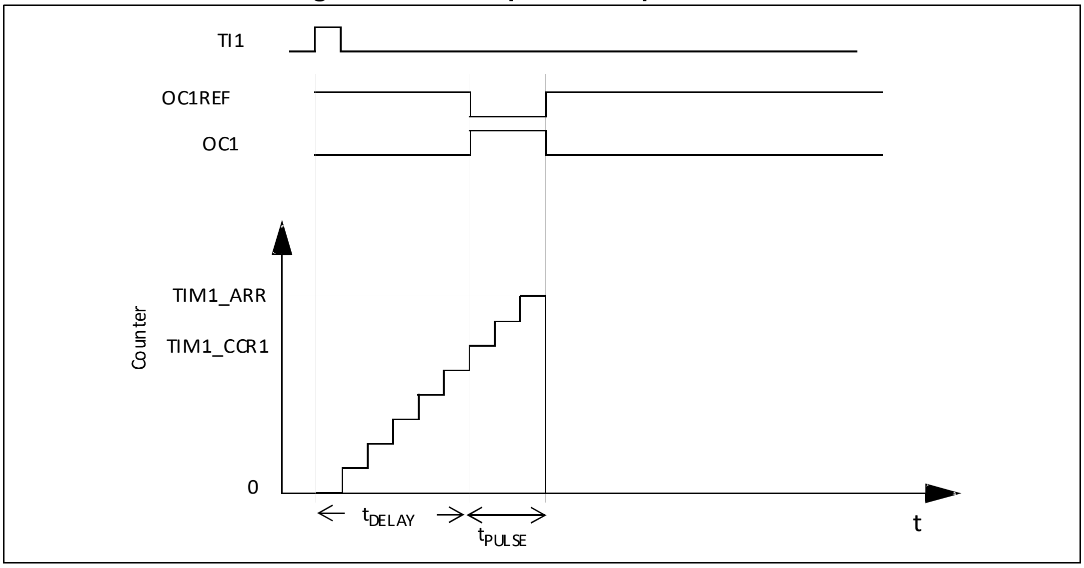
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting, the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 152. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY}.

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say you want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI1. CC1P is written to ‘0’ in this example.

You only want 1 pulse, so you write ‘1’ in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

21.3.14 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 21.3.13: One-pulse mode*:

– The pulse starts as soon as the trigger occurs (no programmable delay)

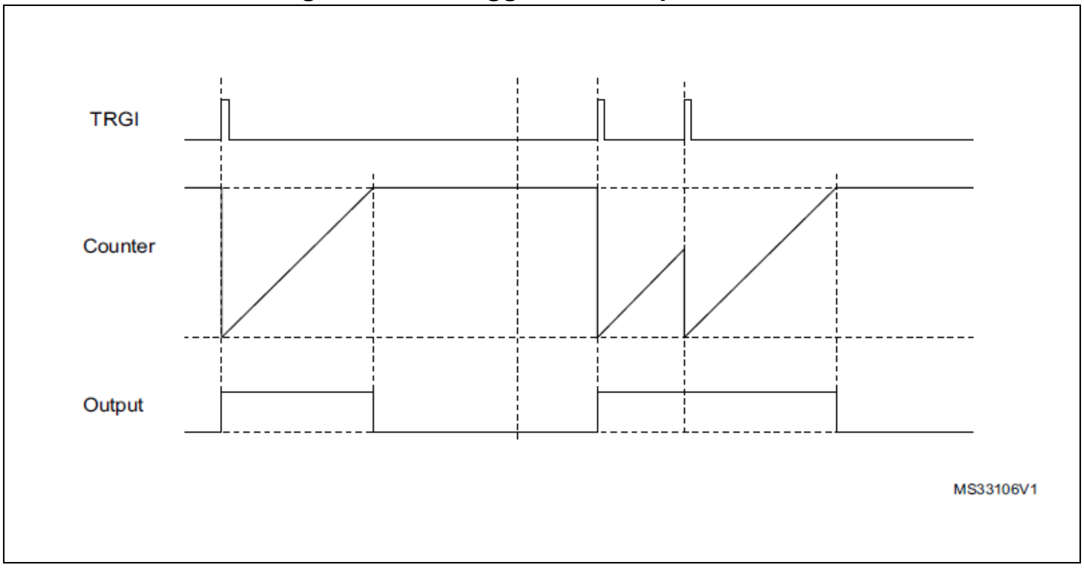
– The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: In retriggerable one pulse mode, the CCxIF flag is not significant. The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones. This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1

Figure 153. Retriggerable one pulse mode



21.3.15 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

21.3.16 Timer16 and external trigger synchronization

The TIM16 Timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

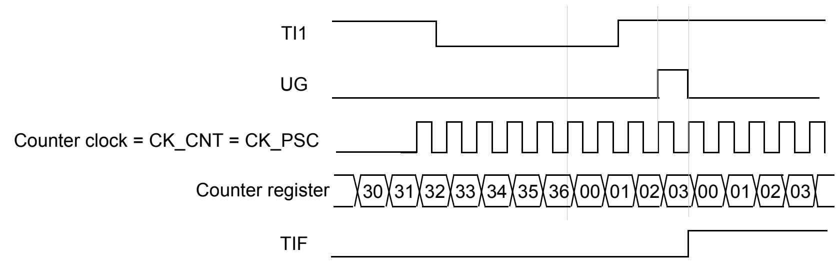
The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated. In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 154. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

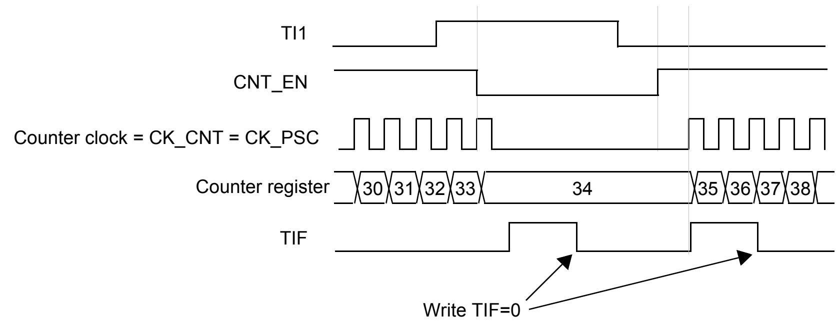
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 155. Control circuit in gated mode



1. The configuration “CCxP=CCxNP=1” (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Note: The configuration “CCxP=CCxNP=1” (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

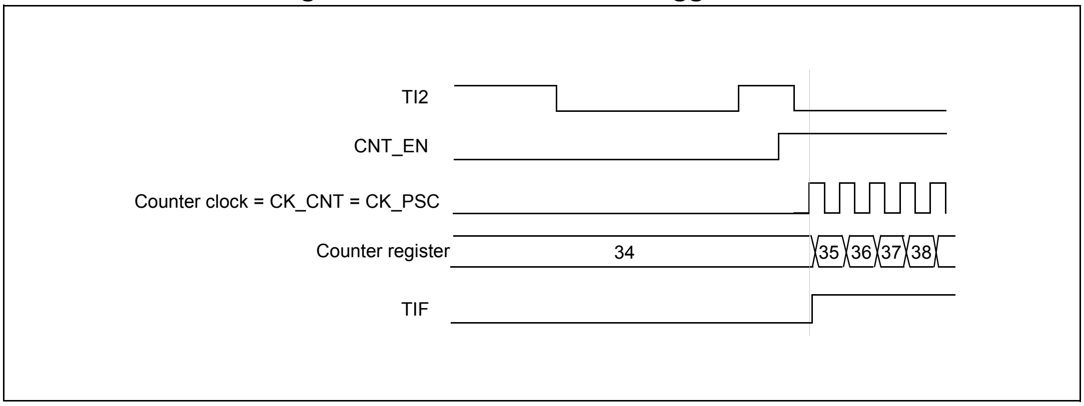
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 156. Control circuit in trigger mode



Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter. This mode is used for one-pulse mode.

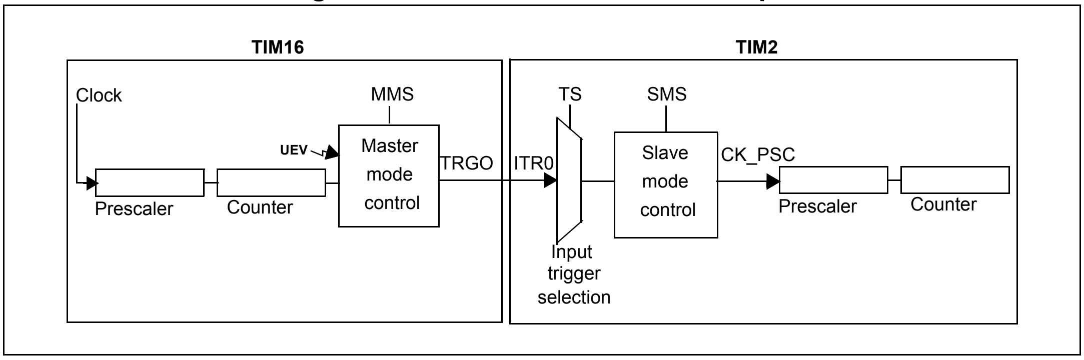
21.3.17 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 157 presents an overview of the trigger selection and

the master mode selection blocks.

Figure 157. Master/slave timers example



Using one timer as prescaler for another timer

For example, TIM16 can be configured to act as a prescaler for TIM2. Refer to *Figure 157*. To do this:

1. Configure TIM16 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM16_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM16 to TIM2, TIM2 must be configured in slave mode using ITR0 as internal trigger. This is selected through the TS bits in the TIM2_SMCR register (writing TS=00000).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM16 trigger signal (which correspond to the TIM16 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM16 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

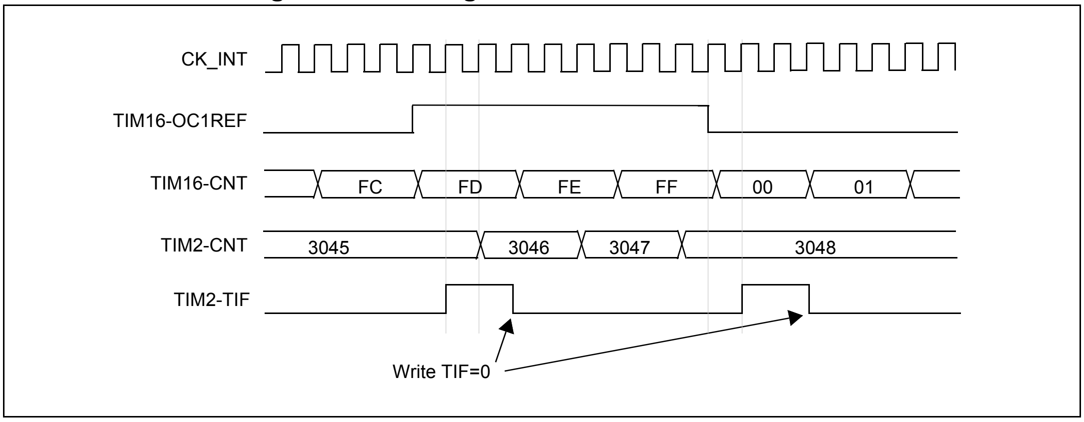
Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 2. Refer to *Figure 157* for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM16 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

1. Configure TIM16 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM16_CR2 register).
2. Configure the TIM16 OC1REF waveform (TIM16_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM16 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
6. Start TIM16 by writing ‘1 in the CEN bit (TIM16_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM16 counter enable signal.

Figure 158. Gating TIM2 with OC1REF of TIM16

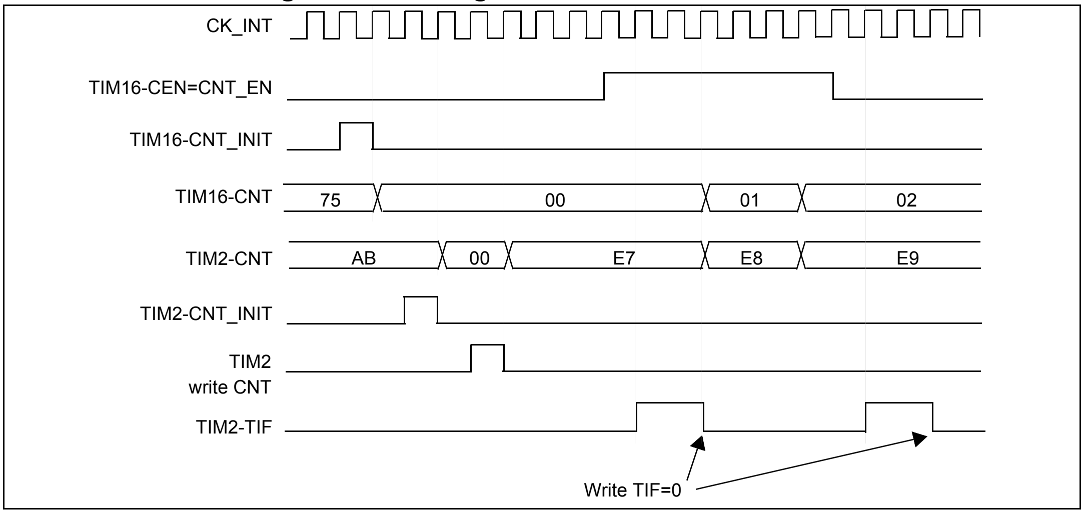


In the example in *Figure 158* , the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM16. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 159*), we synchronize TIM16 and TIM2. TIM16 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM16 is disabled by writing ‘0 to the CEN bit in the TIM16_CR1 register:

1. Configure TIM16 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM16_CR2 register).
2. Configure the TIM16 OC1REF waveform (TIM16_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM16 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM16 by writing ‘1 in UG bit (TIM16_EGR register).
6. Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
7. Initialize TIM2 to 0xE7 by writing ‘0xE7’ in the TIM2 counter (TIM2_CNTL).
8. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
9. Start TIM16 by writing ‘1 in the CEN bit (TIM16_CR1 register).
10. Stop TIM16 by writing ‘0 in the CEN bit (TIM16_CR1 register).

Figure 159. Gating TIM2 with Enable of TIM16



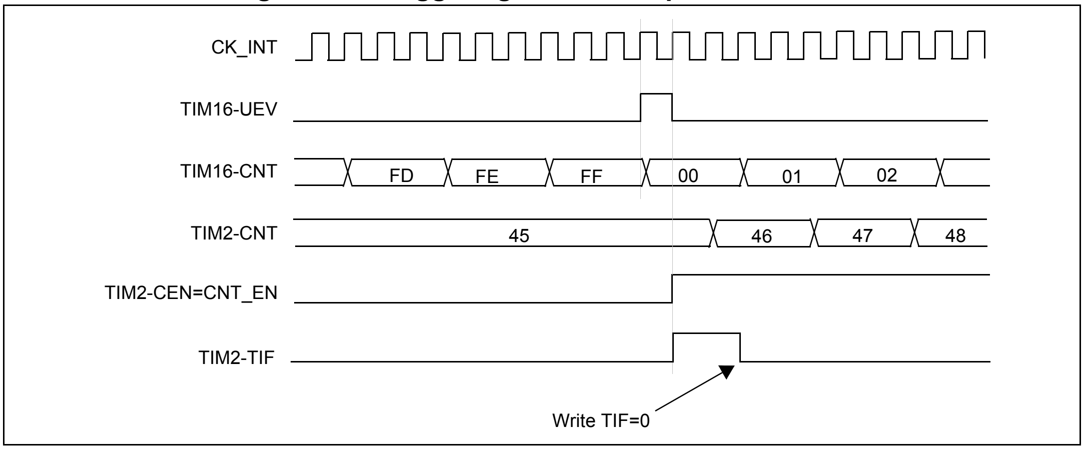
Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 16. Refer to *Figure 157* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 16.

When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write '0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

1. Configure TIM16 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM16_CR2 register).
2. Configure the TIM16 period (TIM16_ARR registers).
3. Configure TIM2 to get the input trigger from TIM16 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM16 by writing '1 in the CEN bit (TIM16_CR1 register).

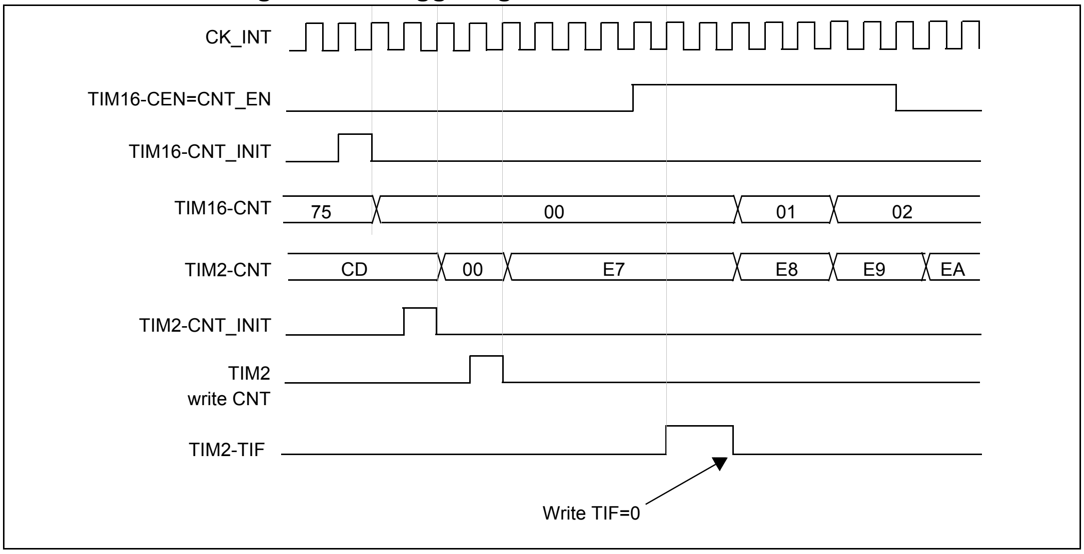
Figure 160. Triggering TIM2 with update of TIM16



As in the previous example, both counters can be initialized before starting counting.

Figure 161 shows the behavior with the same configuration as in *Figure 160* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 161. Triggering TIM2 with Enable of TIM16



Note: The clock of the slave peripherals (timer, DAC, and so on) receiving the TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

21.3.18 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

21.3.19 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIM16_STOP configuration bit in DBGMCU module. For more details, refer to *Section 30.3.2: DBG APB0 freeze register (DBG_APB0_FZ)*.

21.4 TIM16 registers

Refer to *Section 1.1: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

21.4.1 TIM16 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	UIF REM-AP	Res	CKD[1:0]		ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (TIx),
00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

21.4.2 TIM16 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res	CCPC
						rw	rw	rw	rw	rw	rw	rw	rw		rw

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

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input

1: Reserved

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).

100: **Compare** - OC1REF signal is used as trigger output (TRGO).

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

Note: This bit acts only on channels that have a complementary output.

21.4.3 TIM16 slave mode control register (TIM16_SMCR)

Address offset: 0x08

Reset value: 0x0000

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

Bit 31:22 Reserved, must be kept at reset value.

Bit 19:18 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21,20, 6:4 **TS[4:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.
00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010: Internal Trigger 2 (ITR2)
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
Other codes: Reserved
Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.
See *Table 67: TIM16 register map and reset values* for more details on ITRx meaning for each Timer.

Bit 3 Reserved, must be kept at reset value.

Bits16, 2:0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.
0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.
0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.
Other codes: Reserved

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=’00100’). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer16, DAC) receiving the TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 65. TIM16 internal trigger connection

Slave TIM	ITR0	ITR1	ITR2 - ITR8
TIM16	TIM2	-	-

21.4.4 TIM16 DMA/interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	TDE	Res	Res	Res	Res	CC1DE	UDE	BIE	TIE	COMIE	Res	Res	Res	CC1IE	UIE
	rw					rw	rw	rw	rw	rw				rw	rw

Bit 15:13 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

21.4.5 TIM16 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CC1OF	Res	BIF	TIF	COMIF	Res	Res	Res	CC1IF	UIF
						rc_w0		rc_w0	rc_w0	rc_w0				rc_w0	rc_w0

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

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software.

0: No trigger event occurred

1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register.

When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

21.4.6 TIM16 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	BG	TG	COMG	Res	Res	Res	CC1G	UG
								w	w	w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

21.4.7 TIM16 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OC1M[3]
															Res.
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, always read as 0

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode (bits 3 to 0)

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

00000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

00001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

00010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

00011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

00100: Force inactive level - OC1REF is forced low.

00101: Force active level - OC1REF is forced high.

00110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

00111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

All other values: Reserved

Note: ***1:*** *These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

2: *In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1:
1x: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
1x: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

21.4.8 TIM16 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

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

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high

1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1. Refer to the description of CC1P.

*Note: **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).*

***2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.*

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high

1: OC1 active low

CC1 channel configured as input:

The CC1NP/CC1P bits select the polarity of TI1FP1 for trigger or capture operations.

00: Non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

01: Inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode.

10: Reserved, do not use this configuration.

1: Non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

*Note: **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

***2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:
0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 66. Output control bits for complementary OCx and OCxN channels with break feature

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states[1]</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>OCx output state</th><th>OCxN output state</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
OCx=0
OCxN=0, OCxN_EN=0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCx=0</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF XOR CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCxN=0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
OCx=CCxP</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF XOR CCxP, OCx_EN=1</td><td>Off-State (output enabled with inactive state)
OCxN=CCxNP, OCxN_EN=1</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output Disabled (not driven by the timer: Hi-Z)
OCx=CCxP, OCxN=CCxNP</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: OCx=CCxP, OCxN=CCxNP
Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

21.4.9 TIM16 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

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

Bits 15:0 **CNT[15:0]**: Counter value

21.4.10 TIM16 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

21.4.11 TIM16 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Prescaler value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 21.3.1: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

21.4.12 TIM16 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

21.4.13 TIM16 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

21.4.14 TIM16 break and dead-time register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	BKBID	Res	BKDSRM	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
			rw		rw										
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, AOE, BKP, BKE, OSSI, OSSR, DTG[7:0] bits and all used bits of TIMx_AF1 register may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional
0: Break input BRK in input mode
1: Break input BRK in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm
0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (opendrain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:16 Reserved, must be kept at reset value.

Bit 15 **MOE**: Main output enable
This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)
See OC/OCN enable description for more details (*Section 21.4.8: TIM16 capture/compare enable register (TIMx_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note: **1.** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

__2.__ Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK) disabled

1; Break inputs (BRK) enabled

*Note: **1.** This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

__2.__ Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 21.4.8: TIM16 capture/compare enable register (TIMx_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.
See OC/OCN enable description for more details (*Section 21.4.8: TIM16 capture/compare enable register (TIMx_CCER)*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register, BKE/BKP/AOE/BKBID/BKDSRM bits in TIMx_BDTR register and all used bits in TIMx_AF1 register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$
DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$
Example if T_{DTS}=125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

21.4.15 TIM16 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: Reserved,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

21.4.16 TIM16 DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

21.4.17 TIM16 option register 1 (TIM16_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP[1:0]		Res.
													rw	rw	

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

Bits 2:1 **TI1_RMP[1:0]**: Timer 16 input 1 connection
This bit is set and cleared by software.
00: TIM16 TI1 is connected to GPIO
01: TIM16 TI1 is connected to LCO
10: TIM16 TI1 is connected to COMP_OUT (STM32WL33xx only)
11: TIM16 TI1 is connected to MCO

Bit 0 Reserved, must be kept at reset value.

21.4.18 TIM16 alternate function option register 2 (TIMx_AF1)

Address offset: 0x60

Reset value: 0x0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	BKCMP1P	BKINP	Res	Res	Res	Res	Res	Res	Res	BKCMP1E	BKINE
					rw	rw								rw	rw

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

Bit 10 **BKCMP1P**: BRK COMP1 input polarity (STM32WL33xx only) .

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low.

1: COMP1 input is active high.

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

Bit 9 **BKINP**: BRK BKIN input polarity.

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low.

1: BKIN input is active high.

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **BKCMP1E**: BRK COMP1 enable (STM32WL33xx only).

This bit enables the COMP1 for the timer’s BRK input. COMP1 output is ORed with the other enabled BRK sources.

0: COMP1 input disabled.

1: COMP1 input enabled.

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

Bit 0 **BKINE**: BRK BKIN enable.

This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ORed with the other enabled BRK sources.

0: BKIN input disabled.

1: BKIN input enabled.

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

21.4.19 TIM16 input selection register (TIM16_TISEL)

Address offset: 0x68

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIMx_CH1 input
Others: Reserved

21.4.20 TIM16 register map

TIM16 registers are mapped as 16-bit addressable registers as described in the table below:

Table 67. TIM16 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIFREMA	Res	CKD [1:0]		ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIMx_CR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OIS1N	OIS1	Res	MMS			CCDS	CCUS	Res.	CCPC
	Reset value																							0	0		0	0	0	0	0		0
0x08	TIMx_SMCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TS[4:3]		Res	Res	Res	SMS[3]	Res	Res	Res	Res	Res	Res	Res	Res	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x0C	TIMx_DIER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TDE	Res	Res	Res	Res	CC1DE	UDE	OIS1	TIE	COMIE	Res	Res	Res	CC1IE	UIE
	Reset value																		0					0	0	0	0	0				0	0
0x10	TIMx_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC1OF	Res	BIF	TIF	COMIF	Res	Res	Res	CC1IF	UIF
	Reset value																							0		0	0	0				0	0
0x14	TIMx_EGR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BG	TG	COMG	Res	Res	Res	CC1G	UG
	Reset value																									0	0	0				0	0
0x18	TIMx_CCMR1 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OC1M[3]	Res	Res	Res	Res	Res	Res	Res	Res	Res	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0										0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	TIMx_CNT	UIFCPY or Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 67. TIM16 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	TIMx_RCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIMx_BDTR	Res	Res	Res	BKBID	Res	BKDSRM	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBL[4:0]					Res	Res	Res	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIMx_DMAR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	TIMx_OR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1_RMP [1:0]		Res
	Reset value																														0	0	
0x60	TIMx_AF1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BKCMP1P	BKINP	Res	Res	Res	Res	Res	Res	Res	Res	BKCMP1E	BKINE
	Reset value																					0	0									0	1
0x68	TIMx_TISEL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1SEL[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

22 Real-time clock (RTC)

22.1 Introduction

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupt.

The RTC includes also a periodic programmable wakeup flag with interrupt capability.

The RTC provides an automatic wakeup to manage all low power modes.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After power-on reset, all RTC registers are protected against possible parasitic write accesses.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low power mode or under system reset).

22.2 RTC main features

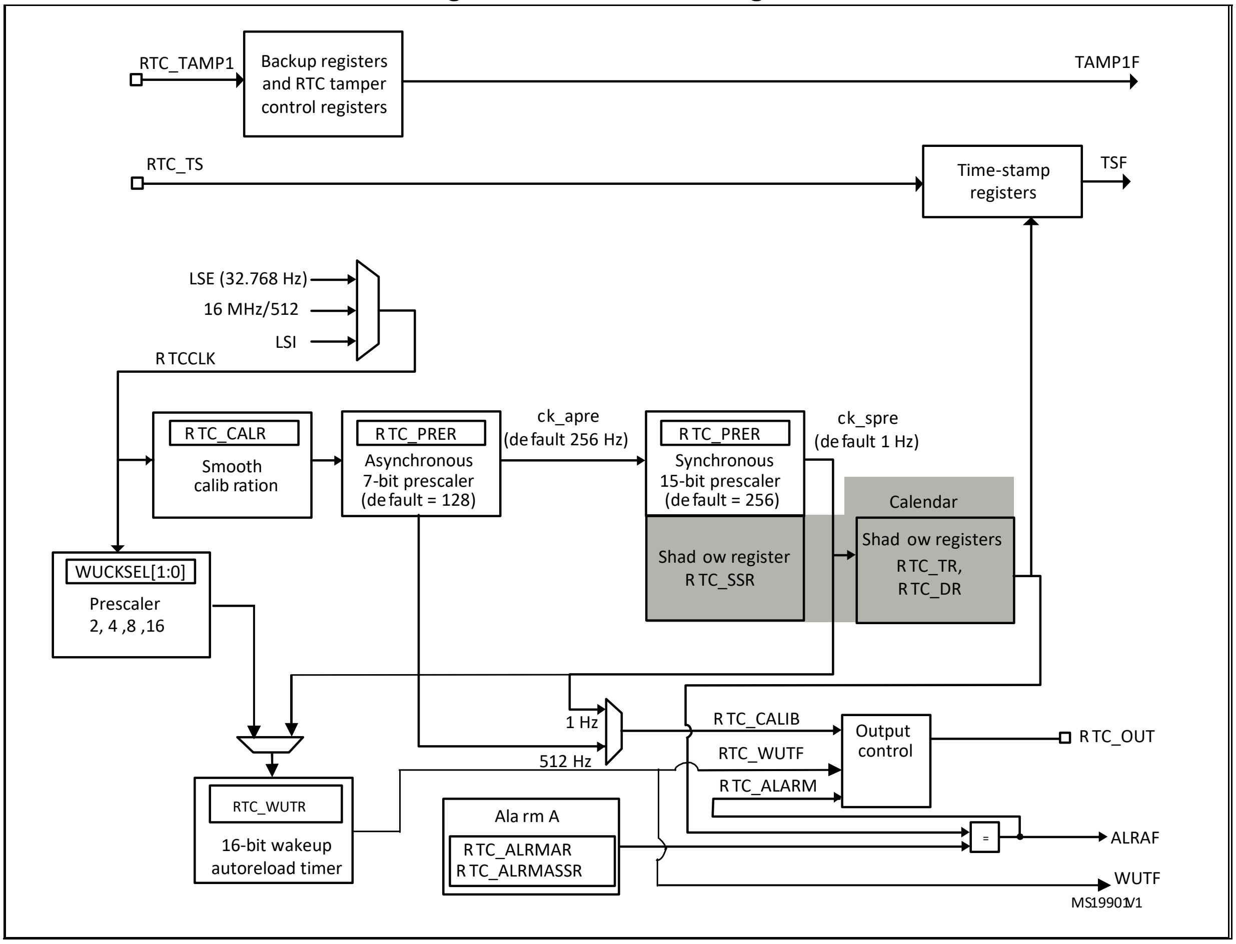
The RTC unit main features are the following (see *Figure 162: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Time-stamp function for event saving
- Tamper detection event with configurable filter and internal pull-up
- Maskable interrupts/events:
  - Alarm A
  - Wakeup interrupt
  - Time-stamp
  - Tamper detection
- Backup registers: the backup registers are reset when a tamper detection event occurs.

22.3 RTC functional description

22.3.1 RTC block diagram

Figure 162. RTC block diagram



22.3.2 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the CLK_16MHz/512 clock (see *Figure 13: System clock details* and *Figure 162: RTC block diagram*). For more information on the RTC clock source configuration, refer to *Section 6: Reset and clock controller (RCC)*.

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 162: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 22.3.5: Periodic auto-wakeup* for details).

22.3.3 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every two RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ISR register is set (see *Section 22.4: RTC low power modes*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 2 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD=0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

22.3.4 Programmable alarm

The RTC unit provides programmable alarm: Alarm A.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register. The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register. The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

Caution: If the seconds field is selected (MSK0 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A (if enabled by bits OSEL[0:1] in RTC_CR register) can be routed to the RTC_ALARM output. RTC_ALARM output polarity can be configured through bit POL the RTC_CR register.

22.3.5 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE(32.768 kHz), this allows to configure the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock)

  When ck_spre frequency is 1Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value.When the initialization sequence is complete (see *Programming the wakeup timer*), the timer starts counting down.When the wakeup function is enabled, the down-counting remains active in low power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_ISR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR2 register, it can exit the device from low power modes.

The periodic wakeup flag can be routed to the RTC_ALARM output provided it has been enabled through bits OSEL[0:1] of RTC_CR register. RTC_ALARM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low power mode (DEEPSTOP) have no influence on the wakeup timer.

22.3.6 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

RTC register write protection

After power-on reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on all the RTC registers except for RTC_ISR[13:8], RTC_TAMPCR, RTC_OR and RTC_BKPxR.

1. Write ‘0xCA’ into the RTC_WPR register.
2. Write ‘0x53’ into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

Note: 1 After a system reset, the application can read the INITS flag in the RTC_ISR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its power-on reset default value (0x00).

2 To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ISR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms.

1. Clear ALRAE in RTC_CR to disable Alarm A.
2. Program the Alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).
3. Set ALRAE in the RTC_CR register to enable Alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ISR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR).Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting.

22.3.7 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register:

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK}) must be equal to or greater than seven times the f_{RTCCLK} RTC clock frequency. This ensures a secure behavior of the synchronization mechanism.

If the APB0 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB0 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ISR register each time the calendar registers are copied into the RTC_TR and RTC_DR shadow registers. The copy is performed every two RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 2 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low power mode (Deepstop), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 22.3.9: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers):

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low power modes (STOP or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD=1, instructions which read the calendar registers require one extra APB cycle to complete.

22.3.8 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and the RTC status register (RTC_ISR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a power-on reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the RTC tamper and alternate-function configuration register (RTC_TAMPCR), the RTC backup registers (RTC_BKPxR), the wakeup timer register (RTC_WUTR), the Alarm A registers (RTC_ALRMASSR/RTC_ALRMAR), and the Option register (RTC_OR).

Also, the RTC keeps on running under system reset if the reset source is different from the power-on reset source. When a power-on reset occurs, the RTC is stopped, and all the RTC registers are set to their reset values.

22.3.9 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR .

RTC_SSR contains the value of the synchronous prescaler’s counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]. The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler's output at 1 Hz. In this way, the frequency of the asynchronous prescaler's output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (PREDIV_S + 1) seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock.

Caution: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow occurs.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

22.3.10 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjusments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, calib_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-

second cycle.

- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting SMC[2] to 1 causes four additional cycles to be masked
- and so on up to SMC[8] set to 1 which causes 256 clocks to be masked.

Note: *CALM[8:0] (RTC_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to '1' causes exactly one pulse to be masked during the 32-second cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1]=1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); SMC[2]=1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to SMC[8]=1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).*

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm ($511/(2^{20}+511)$) with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm ($512/(2^{20}-512)$). Setting CALP to '1' effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [\ 1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)\ ]$$

Calibration when PREDIV_A<3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, either 255 or 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the

calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [\ 1 + (256 - CALM) / (2^{20} + CALM - 256)\ ]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Note: The case PREDIV_A=2 (asynchronous prescaler divides by 3) seems unlikely to be useful unless the nominal input frequency is a multiple of 3. For example, if RTCCLK is nominally 98304Hz (32768Hz x 3), setting PREDIV_S to 32759 rather than 32767 (8 less) would render the above formula valid.

Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8- second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ISR/INITF=0, by using the follow process:

1. Poll the RTC_ISR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR,if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

Note: RECALPF then becomes ‘0’ automatically. Note that RECALPF can stay at ‘1’ for as long as 4 ck_apre cycles plus 2 system clock cycles after writing to RTC_CALR. During initialization mode (RTC_ISR/INIT=1), RECALPF can stay at ‘1’ indefinitely.

22.3.11 Time-stamp function

Time-stamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set: the calendar is saved in the time-stamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a time-stamp event is detected on the RTC_TS pin.

When ITSE is set : the calendar is saved in the time-stamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal time-stamp event is detected. The internal timestamp event is generated by the switch to the VDD12o supply.

When a time-stamp event occurs, due to internal or external event, the time-stamp flag bit (TSF) in RTC_ISR register is set. In case the event is internal, the ITSF flag is also set in RTC_ISR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a time-stamp event occurs.

If a new time-stamp event is detected while the time-stamp flag (TSF) is already set, the time-stamp overflow flag (TSOVF) flag is set and the time-stamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set 2 ck_apre cycles after the time-stamp event occurs due to synchronization process.

Note: There is no delay in the setting of TSOVF. This means that if two time-stamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a time-stamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. This anomaly occurs only if the time-stamp event occurs during a relatively narrow time window of one APB clock cycle. Note that this time To avoid masking a time-stamp event occurring at the same moment, the application must not write '0' into TSF bit unless it has already read it to '1'.

Optionally, a tamper event can cause a time-stamp to be recorded. See the description of the TAMPTS control bit in *Section 22.6.13: RTC time-stamp sub second register (RTC_TSSSR)*.

Caution: corresponds exactly to the period between TSOVF being set and the TSF bit being set. To avoid masking a time-stamp event occurring at the same moment, the application must not write '0' into TSF bit unless it has already read it to '1'.If the TSF bit is already cleared and the application writes a zero to this bit, then TSOVF may be set if a time-stamp occurs at the same moment. This anomaly occurs only if the time-stamp event occurs during a relatively narrow time window of one APB clock cycle. As a consequence, it is recommended not to clear the TSF bit unless it has already been read to 1 to avoid masking a time-stamp event. Optionally, a tamper event can cause a time-stamp to be recorded. See the description of the TAMPTS control bit in Section 34.6.14: RTC time-stamp sub second register (RTC_TSSSR).

22.3.12 Tamper detection

The RTC_TAMPx input events can be configured either for edge detection, or for level detection with filtering.

The tamper detection can be configured for these three purposes :

- erase the RTC backup registers (default configuration)
- generate an interrupt, capable to wakeup from Deepstop mode.
- generate a hardware trigger for the low power timers

RTC backup registers

The backup registers (RTC_BKPxR) are implemented in the VDD12o domain that remains powered-on by VDD12o when the VDD12i power is switched off. They are not reset by system reset, or when the device wakes up from DEESPTOP mode. They are reset by a power-on reset.

By default, the backup registers are reset when a tamper detection event occurs(see *Section 22.6.21: RTC backup registers (RTC_BKPxR)* and *Tamper detection initialization*. If TAMPxNOERASE bit is set, or if TAMPxMF is set in the RTC_TAMPCR register, the backup registers are not erased when the corresponding RTC_TAMPx input event occurs.

Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the RTC_TAMPCR register.

Each RTC_TAMPx tamper detection input is associated with a flag TAMPxF in the RTC_ISR register.

When TAMPxMF is cleared, the TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck_apre cycles when TAMPFLT differs from 0x0 (Level detection with filtering)
- 3 ck_apre cycles when TAMPTS=1 (Timestamp on tamper event)
- No latency when TAMPFLT=0x0 (Edge detection) and TAMPTS=0

A new tamper occuring on the same pin during this period and as long as TAMPxF is set cannot be detected.

In case TAMPxMF is set, a new tamper occuring on the same pin cannot be detected during the latency described above and 2.5 ck_rtc additional cycles.

By setting the TAMPIE bit in the RTC_TAMPCR register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set).

When TAMPIE is cleared, each tamper pin event interrupt can be individually enabled by setting the corresponding TAMPxIE bit in the RTC_TAMPCR register.

Timestamp on tamper event:

With TAMPTS set to ‘1’ , any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit are set in RTC_ISR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set at the same time that TSF or TSOVF is set.

Edge detection on tamper inputs

If the TAMPFLT bits are “00”, the RTC_TAMPx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the RTC_TAMPx inputs are deactivated when edge detection is selected.

Caution: To avoid losing tamper detection events, the signal used for edge detection is logically ANDed with the corresponding TAMPxE bit in order to detect a tamper detection event in case it occurs before the RTC_TAMPx pin is enabled.

When TAMPxTRG = 0: if the RTC_TAMPx alternate function is already high before tamper detection is enabled (TAMPxE bit set to 1), a tamper event is detected as soon as the RTC_TAMPx input is enabled, even if there was no rising edge on the RTC_TAMPx input after TAMPxE was set.

When TAMPxTRG = 1: if the RTC_TAMPx alternate function is already low before tamper detection is enabled, a tamper event is detected as soon as the RTC_TAMPx input is enabled (even if there was no falling edge on the RTC_TAMPx input after TAMPxE was set.

After a tamper event has been detected and cleared, the RTC_TAMPx alternate function should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (RTC_BKPxR). This prevents the application from writing to the backup registers while the RTC_TAMPx input value still indicates a tamper detection. This is equivalent to a level detection on the RTC_TAMPx alternate function input.

Note: Tamper detection is still active when VDD12i power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the RTC_TAMPx alternate function is mapped should be externally tied to the correct level.

Level detection with filtering on RTC_TAMPx inputs

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The RTC_TAMPx inputs are pre-charged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1,The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the RTC_TAMPx inputs.

The tradeoff between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the datasheets for the electrical characteristics of the pull-up resistors.

Level detection with filtering on RTC_TAMPx inputs

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The RTC_TAMPx inputs are pre-charged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1,The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the RTC_TAMPx inputs.

The tradeoff between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the datasheets for the electrical characteristics of the pull-up resistors.

22.3.13 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the RTC_CALIB device output.

Note: This RTC_CALIB information is output on the RTC_OUT I/O signal if the I/O is programmed with the associated AFx mode (see Section 4: I/O operating modes).

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the RTC_CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The RTC_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the RTC_CALIB frequency is fRTCCLK/(256 * (PREDIV_A+1)). This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

22.3.14 Alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm alternate function output RTC_ALARM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_ISR register.

The polarity of the output is determined by the POL control bit in RTC_CR so that the opposite of the selected flag bit is output when POL is set to 1.

Alarm alternate function output

The RTC_ALARM pin can be configured in output open drain or output push-pull using the control bit ALARMOUTTYPE in the RTC_TAMPCR register.

Note: 1 The RTC_ALARM is output on the RTC_OUT I/O signal if the I/O is programmed with the associated AFx mode (see Table 8: GPIO alternate options AF0 - AF3)

22.4 RTC low power modes

The RTC is able to run in Deepstop mode and generate a wakeup event to wake the device through RTC tamper, RTC timestamp, RTC alarm and RTC wakeup root causes.

Note: The software has to clear the RTC_ISR.WUTF or RTC_ISR.TAMP1F or RTC_ISR.ALRAF or RTC_ISR.TSF flag in the RTC after a wakeup otherwise it prevents re entry in low power. The PWRC block only mirrors the RTC wakeup signal in its own wakeup flag register.

22.5 RTC interrupts

All RTC interrupts are combined and connected to the NVIC controller. Refer to *Table 7: Interrupt vectors.*

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the RTC_ALARM IRQ channel in the NVIC.
2. Configure the RTC to generate RTC alarms (Alarm A).

To enable the RTC Tamper interrupt, the following sequence is required:

1. Configure and enable the RTC_ALARM IRQ channel in the NVIC.
2. Configure the RTC to detect the RTC tamper event.

To enable the RTC TimeStamp interrupt, the following sequence is required:

1. Configure and enable the RTC_ALARM IRQ channel in the NVIC.
2. Configure the RTC to detect the RTC time-stamp event.

To enable the Wakeup timer interrupt, the following sequence is required:

1. Configure and Enable the RTC IRQ channel in the NVIC.
2. Configure the RTC to detect the WUT event.

22.6 RTC registers

Refer to *Section 1.1: List of abbreviations for registers* of the reference manual for a list of abbreviations used in register descriptions.

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

22.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration* and *Reading the calendar*.

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x00

Power-on reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0 . Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw

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

Bits 31-23 Reserved, must be kept at reset value

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bit 16:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bit 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bit 3:0 **SU[3:0]**: Second units in BCD format

22.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration* and *Reading the calendar*.

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x04

Power-on reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0 . Not affected when BYPSHAD = 1.

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

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

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

22.6.3 RTC control register (RTC_CR)

Address offset: 0x08

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
							rw	rw	rw	rw	rw	rw	rw	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSIE	WUTIE	Res.	ALRAIE	TSE	WUTE	Res.	ALRAE	Res.	FMT	BYPSHAD	Res.	TSEDGE	WUCKSEL[2:0]		
rw	rw		rw	rw	rw		rw		rw	rw		rw	rw	rw	rw

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

Bit 24 **ITSE**: timestamp on internal event enable
0: internal event timestamp disable
1: internal event timestamp enable

Bit 23 **COE**: Calibration output enable
This bit enables the RTC_CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection
These bits are used to select the flag to be routed to RTC_ALARM output
00: Output disabled
01: Alarm A output enabled
10:
11: Wakeup output enabled

Bit 20 **POL**: Output polarity
This bit is used to configure the polarity of RTC_ALARM output
0: The pin is high when ALRAF/WUTF is asserted (depending on OSEL[1:0])
1: The pin is low when ALRAF/WUTF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL** : Calibration output selection
When COE=1, this bit selects which signal is output on RTC_CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A=127 and PREDIV_S=255). Refer to *Section 22.3.13: Calibration clock output*

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Time-stamp interrupt enable

0: Time-stamp Interrupt disable

1: Time-stamp Interrupt enable

Bit 15 **TSIE**: Time-stamp interrupt enable

0: Time-stamp Interrupt disable

1: Time-stamp Interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable

0: Wakeup timer interrupt disabled

1: Wakeup timer interrupt enabled

Bit 13 Reserved, must be kept at reset value

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled

1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp on RTC_TS input edge enable

0: timestamp on RTC_TS input edge disable

1: timestampon RTC_TS input edge enable

Bit 11 **TSE**: timestamp on RTC_TS input edge enable

0: timestamp on RTC_TS input edge disable

1: timestampon RTC_TS input edge enable

Bit 10 **WUTE**: Wakeup timer enable

0: Wakeup timer disabled

1: Wakeup timer enabled

Bit 9 Reserved, must be kept at reset value

Bit 8 **ALRAE:** Alarm A enable

0: Alarm A disabled

1: Alarm A enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to ‘1’ .

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TSEDGE**: Time-stamp event active edge

0: RTC_TS input rising edge generates a time-stamp event
1: RTC_TS input falling edge generates a time-stamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bit 3 **TSEDGE**: Time-stamp event active edge

0: RTC_TS input rising edge generates a time-stamp event
1: RTC_TS input falling edge generates a time-stamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: Wakeup clock selection

000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected
11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value (see note below)

Note:
1. *Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).*
2. *WUT = Wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].*
3. *Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1.*
4. *It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.*
5. *ADD1H and SUB1H changes are effective in the next second.*
6. *This register is write protected. The write access procedure is described in RTC register write protection.*

22.6.4 RTC initialization and status register (RTC_ISR)

This register is write protected (except for RTC_ISR[-17:8] bits). The write access procedure is described in *RTC register write protection*.

Address offset: 0x0C

Reset value: 0x0000 0007

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSF	RECALPF
														rc_w0	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	TAMP1F	TSOVF	TSF	WUTF	Res.	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	ALRAWF
		rc_w0	rc_w0	rc_w0	rc_w0		rc_w0	rw	r	rc_w0	r	rc_w0	r		r

Bits 31:18 Reserved, must be kept at reset value

Bit 17 **ITSF**: Internal time-stamp flag

This flag is set by hardware when atime-stamp on the internal event occurs.
This flag is cleared by software by writing 0, and must be cleared together with TSF bit by writing 0 in both bits.

Bit 17 **ITSF**: Internal time-stamp flag

This flag is set by hardware when atime-stamp on the internal event occurs.
This flag is cleared by software by writing 0, and must be cleared together with TSF bit by writing 0 in both bits.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to ‘1’ when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to ‘0’. Refer to *Re-calibration on-the-fly*.

Bit 15 Reserved, must be kept at reset value

Bit 14 Reserved, must be kept at reset value

Bit 13 **TAMP1F**: RTC_TAMP1 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP1 input.
It is cleared by software writing 0

Bit 13 **TAMP1F**: RTC_TAMP1 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP1 input.
It is cleared by software writing 0

Bit 12 **TSOVF**: Time-stamp overflow flag

This flag is set by hardware when a time-stamp event occurs while TSF is already set.
This flag is cleared by software by writing 0. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a time-stamp event occurs immediately before the TSF bit is cleared.

Bit 12 **TSOVF**: Time-stamp overflow flag

This flag is set by hardware when a time-stamp event occurs while TSF is already set.

This flag is cleared by software by writing 0. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a time-stamp event occurs immediately before the TSF bit is cleared.

Bit 11 **TSF**: Time-stamp flag

This flag is set by hardware when a time-stamp event occurs.

This flag is cleared by software by writing 0. If ITSF flag is set, TSF must be cleared together with ITSF by writing 0 in both bits.

Bit 11 **TSF**: Time-stamp flag

This flag is set by hardware when a time-stamp event occurs.

This flag is cleared by software by writing 0. If ITSF flag is set, TSF must be cleared together with ITSF by writing 0 in both bits.

Bit 10 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.

This flag is cleared by software by writing 0.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm A register (RTC_ALRMAR).

This flag is cleared by software by writing 0.

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed.

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow regsiter mode (BYPSHAD=1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (power-on reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**:Shift operation pending

0: No shift operation is pending

1: A shift operation is pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware when the wakeup timer values can be changed, after the WUTE bit has been set to 0 in RTC_CR.

0: Wakeup timer configuration update not allowed

1: Wakeup timer configuration update allowed.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ALRAWF**: Alarm A write flag

This bit is set by hardware when Alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm A update not allowed

1: Alarm A update allowed

Note: 1 The bits ALRAF, WUTF and TSF are cleared 2 APB clock cycles after programming them to 0.

22.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration*

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x10

Power-on reset value: 0x007F 00FF

System reset: not affected

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

Bits 31:23 Reserved, must be kept at reset value

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor
This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor
This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

22.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ISR.

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x14

Power-on reset value: 0x0000 FFFF

System reset: not affected

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

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

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

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs (WUT+1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] =011 (RTCCLK/2) is forbidden.

22.6.7 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in ().

Address offset: 0x1C

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don’t care in Alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format.

Bits 27:24 **DU[3:0]**: Date units or day in BCD format.

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don’t care in Alarm A comparison

Bit 22 **PM:** AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don’t care in Alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don’t care in Alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

22.6.8 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

22.6.9 RTC sub second register (RTC_SSR)

Address offset: 0x28

Power-on reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0 . Not affected when BYPSHAD = 1.

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

Bits31:16 Reserved, must be kept at reset value

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value in the synchronous prescaler's counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

22.6.10 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

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

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 31:15 Reserved, must be kept at reset value

Bits 14:0 **SUBFS**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

The value which is written to SUBFS is added to the synchronous prescaler's counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by :

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))) .

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.

Refer to *Section 22.3.9: RTC synchronization*.

22.6.11 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x30
Power-on reset value: 0x0000 0000
System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r

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

Bits 31:23 Reserved, must be kept at reset value

Bit 22 **PM:** AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

22.6.12 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x34

Power-on reset value: 0x0000 0000

System reset: not affected

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

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

Bits 15:13 **WDU[1:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bit 3:0 **DU[3:0]**: Date units in BCD format

22.6.13 RTC time-stamp sub second register (RTC_TSSSR)

The content of this register is valid only when RTC_ISR/TSF is set. It is cleared when the RTC_ISR/TSF bit is reset.

Address offset: 0x38

Power-on reset value: 0x0000 0000

System reset: not affected

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

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

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value of the synchronous prescaler’s counter when the timestamp event occured.

22.6.14 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x30
Power-on reset value: 0x0000 0000
System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r

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

Bits 31:23 Reserved, must be kept at reset value

Bit 22 **PM:** AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

22.6.15 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x34

Power-on reset value: 0x0000 0000

System reset: not affected

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

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

Bits 15:13 **WDU[1:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bit 3:0 **DU[3:0]**: Date units in BCD format

22.6.16 RTC time-stamp sub second register (RTC_TSSSR)

The content of this register is valid only when RTC_ISR/TSF is set. It is cleared when the RTC_ISR/TSF bit is reset.

Address offset: 0x38

Power-on reset value: 0x0000 0000

System reset: not affected

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

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

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value of the synchronous prescaler's counter when the timestamp event occured.

22.6.17 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x3C

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALP	CALW8	CALW 16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:16 Reserved, must be kept at reset value

Bit 15 CALP: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency incresed by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 * CALP) - CALM.

Refer to *Section 22.3.10: RTC smooth digital calibration*.

Bit 14 **CALW8:** Use an 8-second calibration cycle period

When CALW8 is set to ‘1’ , the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stucked at “00” when CALW8=’1’. Refer to Section 22.3.10: RTC smooth digital calibration.

Bit 13 **CALW16:** Use a 16-second calibration cycle period

When CALW16 is set to ‘1’ , the 16-second calibration cycle period is selected.This bit must not be set to ‘1’ if CALW8=1.

Note: CALM[0] is stucked at ‘0’ when CALW16=’1’. Refer to Section 22.3.10: RTC smooth digital calibration.

Bits 12:9 Reserved, must be kept at reset value

Bits 8:0 CALM[8:0]: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 22.3.10: RTC smooth digital calibration*.

22.6.18 RTC_TAMPCR register (RTC_TAMPCR)

Address offset: 0x0040

Reset value: 0x0000 0000

RTC Tamper Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP1MF	TAMP1NOERASE	TAMP1IE
r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	Res.	Res.	Res.	Res.	TAMPIE	TAMP1TRG	TAMP1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **TAMP1MF**: Tamper 1 mask flag

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

- 1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware.The backup registers are not erased.

Bit 17 **TAMP1NOERASE**: Tamper 1 no erase

0: Tamper 1 event erases the backup registers.

1: Tamper 1 event does not erase the backup registers.

Bit 16 **TAMP1IE**: Tamper 1 interrupt enable

0: Tamper 1 interrupt is disabled if TAMPIE = 0.

1: Tamper 1 interrupt enabled.

Bit 15 **TAMPPUDIS**: RTC_TAMPx pull-up disable

This bit determines if each of the RTC_TAMPx pins are pre-charged before each sample.

0: Precharge RTC_TAMPx pins before sampling (enable internal pull-up)

1: Disable precharge of RTC_TAMPx pins.

Bits 14:13 **TAMPPRCH[1:0]**: RTC_TAMPx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the RTC_TAMPx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 12:11 **TAMPFLT[1:0]**: RTC_TAMPx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a Tamper event.

TAMPFLT is valid for each of the RTC_TAMPx inputs.

0x0: Tamper event is activated on edge of RTC_TAMPx input transitions to the active level (no internal pull-up on RTC_TAMPx input).

0x1: Tamper event is activated after 2 consecutive samples at the active level.

0x2: Tamper event is activated after 4 consecutive samples at the active level.

0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 10:8 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the RTC_TAMPx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Bit 7 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a timestamp to be saved

1: Save timestamp on tamper detection event

TAMPTS is valid even if TSE=0 in the RTC_CR register.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **TAMPIE**: Tamper interrupt enable

0: Tamper interrupt disabled

1: Tamper interrupt enabled.

Bit 1 **TAMP1TRG**: Active level for RTC_TAMP1 input

If TAMPFLT != 00

0: RTC_TAMP1 input staying low triggers a tamper detection event.

1: RTC_TAMP1 input staying high triggers a tamper detection event.

if TAMPFLT = 00:

0: RTC_TAMP1 input rising edge triggers a tamper detection event.

\x2d 1: RTC_TAMP1 input falling edge triggers a tamper detection event.

Bit 0 **TAMP1E**: RTC_TAMP1 input detection enable

0: RTC_TAMP1 detection disabled

1: RTC_TAMP1 detection enabled.

Caution: When TAMPFLT = 0, TAMP1E must be reset when TAMP1TRG is changed to avoid spuriously setting TAMP1F.

22.6.19 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection*

Address offset: 0x44

Power-on reset value: 0x0000 0000

System reset: not affected

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

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

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

Bit 27:24 MASKSS[3:0]: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[14:1] are don’t care in Alarm A comparison. Only SS[0] is compared.
2: SS[14:2] are don’t care in Alarm A comparison. Only SS[1:0] are compared.
3: SS[14:3] are don’t care in Alarm A comparison. Only SS[2:0] are compared.
...
12: SS[14:12] are don’t care in Alarm A comparison. SS[11:0] are compared.
13: SS[14:13] are don’t care in Alarm A comparison. SS[12:0] are compared.
14: SS[14] is don’t care in Alarm A comparison. SS[13:0] are compared.
15: All 15 SS bits are compared and must match to activate alarm.
The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bit 23:15 Reserved, must be kept at reset value.

Bit 14:0 SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler’s counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

22.6.20 RTC option register (RTC_OR)

Address offset: 0x4C

Power-on reset value: 0x0000 0000

System reset: not affected

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

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

Bit 1 **RTC_OUT_RMP:** RTC_OUT remap

Setting this bit allows to remap the RTC outputs on PA9 as follows:

– RTC_OUT_RMP = ‘0’ :

If OSEL/= “00” : RTC_ALARM is ouput on PA8

If OSEL= “00” and COE = ‘1’ : RTC_CALIB is output on PA8

– RTC_OUT_RMP = ‘1’:

If OSEL /= “00” and COE = ‘0’ : RTC_ALARM is output on PA9

If OSEL = “00” and COE = ‘1’: RTC_CALIB is output on PA9

If OSEL /= “00” and COE = ‘1’: RTC_CALIB is output on PA9 and RTC_ALARM is output on PA8.

Note: the RTC outputs are functional in Deepstop mode only on PA8.

Bit 0 **ALARMOUTTYPE:** RTC_ALARM on PA8 output type

0: RTC_ALARM, when mapped on PA8, is open-drain output

1: RTC_ALARM, when mapped on PA8, is push-pull output

22.6.21 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0x54

Power-on reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 BKP[31:0]

The application can write or read data to and from these registers.

They are powered-on by VDD12o so they are retained during Deepstop mode.

The application can write or read data to and from these registers. This register is reset on PORESETn or on a tamper detection event, as long as TAMPxF=1, or when the flash readout protection is disabled.

22.7 RTC register map

Table 68. RTC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_TR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**RTC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	**RTC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSE	COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	Res.	ALRAIE	TSE	WUTE	Res.	ALRAE	Res.	FMT	BYPSHAD	Res.	TSEDGE	WUCKSEL[2:0]		
	Reset value								0	0	0	0	0		0	0	0	0	0		0	0	0		0		0	0	0	0	0	0	0
0x0C	**RTC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSF	RECALPF	Res.	Res.	TAMP1F	TSOVF	TSF	WUTF	Res.	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	ALRAWF
	Reset value															0	0			00	00	00	0		0	0	0	0	0	0	1		1
0x10	**RTC_PRER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							Res.	PREDIV_S[14:0]														
	Reset value										1	1	1	1	1	1	1		0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	**RTC_ALRMAR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**RTC_WPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_SSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**RTC_TSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	**RTC_TSDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	**RTC_TSSSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 68. RTC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	**RTC_TSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	**RTC_TSDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	**RTC_TSSSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0
0x40	**RTC_TAMPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP1MF	TAMP1NOERASE	TAMP1IE	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	Res.	Res.	Res.	Res.	TAMPIE	TAMP1TRG	TAMP1E
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0					0	0	0
0x44	RTC_ ALRMASSR	Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**RTC_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_OUT_RMP	ALARMOUTTYPE
	Reset value																															0	0
0x50 to 0x54	**RTC_BKPxR**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

23 Independent watchdog (IWDG)

23.1 Introduction

The devices feature an embedded watchdog peripheral which offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral serves to detect and resolve malfunctions due to software failure, and to trigger system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints.

23.2 IWDG main features

- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional Reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 000h
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

23.3 IWDG functional description

Figure 163 shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0x0000 CCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

23.3.1 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the IWDG_WINR register.

If the reload operation is performed while the counter is greater than the value stored in the window register (IWDG_WINR), then a reset is provided.

The default value of the IWDG_WINR is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the IWDG_RLR value and ease the cycle number calculation to generate the next reload.

Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG_KR register.
2. Enable register access by writing 0x0000 5555 in the IWDG_KR register.
3. Write the IWDG prescaler by programming IWDG_PR from 0 to 7.
4. Write the reload register (IWDG_RLR).
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the window register IWDG_WINR. This automatically refreshes the counter value IWDG_RLR.

Note: Writing the window value allows to refresh the Counter value by the RLR when IWDG_SR to set to 0x0000 0000.

Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable register access by writing 0x0000 5555 in the IWDG_KR register.
2. Write the IWDG prescaler by programming IWDG_PR from 0 to 7.
3. Write the reload register (IWDG_RLR).
4. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
5. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA).
6. Enable the IWDG by writing 0x0000 CCCC in the IWDG_KR.

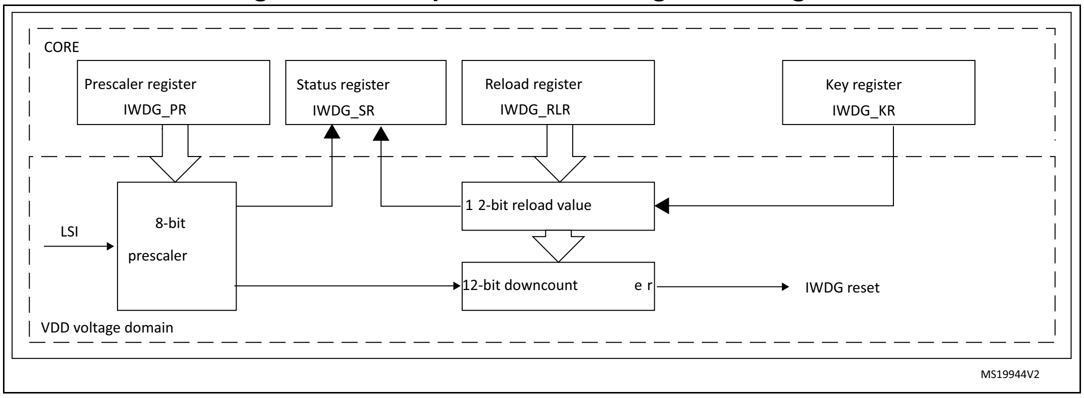
23.3.2 Register access protection

Write access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers is protected. To modify them, you must first write the code 0x0000 5555 in the IWDG_KR register. A write access to this register with a different value breaks the sequence and register access is protected again. This implies that it is the case of the reload operation (writing 0x0000 AAAA).
A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is on going.

23.3.3 Debug mode

No specific debug mode implemented in the STM32WL30xx/31xx/33xx. The timer goes on counting even when the CPU is halted by the debugger.

Figure 163. Independent watchdog block diagram



Note: The watchdog is implemented in the VDD12o power domain that is still functional in Deepstop mode.

23.4 IWDG registers

Refer to *Section 1.3: Acronyms* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

23.4.1 Key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)
These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.
Writing the key value 0x5555 to enables access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers (see *Section 23.3.2*)
Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

23.4.2 Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

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

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see *Section 23.3.2*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider.

000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

Note: Reading this register returns the prescaler value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG_SR register is reset.

23.4.3 Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

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

Bits 31:12 Reserved, must be kept at reset value.

Bits11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Section 23.3.2*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet for the timeout information.

The RVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on this register. For this reason the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset.

23.4.4 Status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset by Standby mode)

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

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

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

23.4.5 Window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF (reset by Standby mode)

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected see *Section 23.3.2*. These bits contain the high limit of the window value to be compared to the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG_SR register is reset.

Note: If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.

23.5 IWDG register map

The following table gives the IWDG register map and reset values.

Table 69. IWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
	Reset value																														0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WVU	RVU	PVU
	Reset value																														0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

24 Inter-integrated circuit (I2C) interface

24.1 Introduction

The availability of the I2C interface depends on the specific STM32WL30xx/31xx/33xx product:

- The STM32WL31xx embeds one I^2C (inter-integrated circuit) interface: I2C1.
- The STM32WL33xx embeds two I^2C (inter-integrated circuit) interfaces: I2C1 and I2C2.
- I^2C is not available on the STM32WL30xx device.

The I^2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I^2C bus. It provides multicontroller capability, and controls all I^2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

24.2 I2C main features

- I^2C bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available depending on the product implementation (see *Section 24.3: I2C implementation*):

- SMBus specification rev 2.0 compatibility:
  - Hardware PEC (Packet Error Checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and Device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.1 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming

Note: For the Fast-Mode Plus mode, it is strongly recommended to use I2C pins mentioned as open-drain capable (embedding a 10 ns/50 ns filter).

24.3 I2C implementation

This manual describes the full set of features implemented in I2C1 (STM32WL31xx/33xx only) and I2C2 (STM32WL33xx only).

Table 70. I2C implementation

I2C features(1)	I2C1	I2C2
7-bit addressing mode	X	X
10-bit addressing mode	X	X
Standard-mode (up to 100 kbit/s)	X	X
Fast-mode (up to 400 kbit/s)	X	X
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	X	X
Independent clock	X	X
SMBus	X	X

1. X = supported.

24.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I^2C bus.

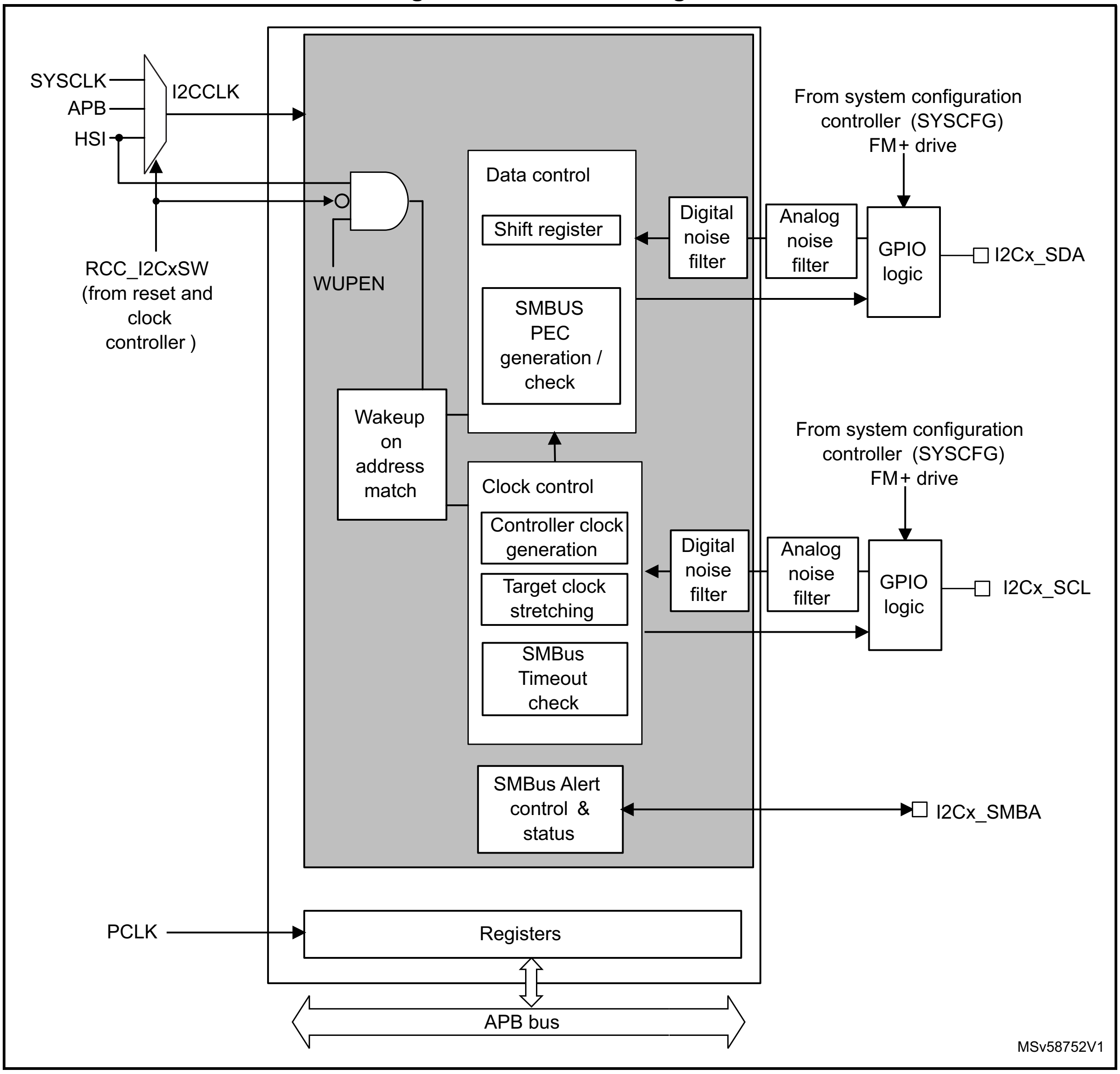
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

24.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 164*.

Figure 164. I2C block diagram



The I2C is clocked by an independent clock source which allows to the I2C to operate independently from the PCLK frequency.

This independent clock source is a fixed 16 MHz clock. Refer to *Section 6: Reset and clock controller (RCC)* for more details.

I2C I/Os support 20 mA output current drive for Fast-mode Plus operation. This is enabled by setting the driving capability control bits for SCL and SDA in *Section 8.2.3: Fast-Mode Plus pin capability control register (I2C_FMP_CTRL)*.

24.4.2 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (tLOW - tfilters) / 4$ and $t_{I2CCLK} < tHIGH$

with:

t_{LOW}: SCL low time and tHIGH : SCL high time

$t_{filters}$: when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is DNF x t_{I2CCLK}.

The PCLK clock period t_{PCLK} must respect the following condition:

$t_{PCLK} < 4/3\ tSCL$

with t_{SCL}: SCL period

24.4.3 Mode selection

The interface can operate in one of the four following modes:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, it operates in target mode. The interface automatically switches from target to controller when it generates a START condition, and from controller to target if an arbitration loss or a STOP generation occurs, allowing multicontroller capability.

Communication flow

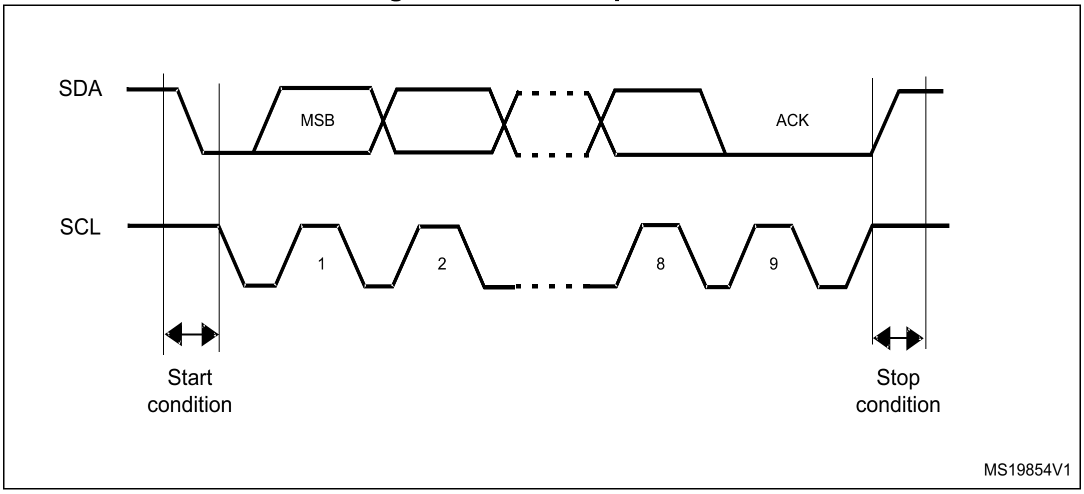
In Controller mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In Target mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Controller mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Figure 165. I²C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

24.4.4 I2C initialization

Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller (refer to *Section 6: Reset and clock controller (RCC)*).

Then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE=0), the I^2C performs a software reset. Refer to *Section 24.4.5: Software reset* for more details.

Noise filters

Before you enable the I2C peripheral by setting the PE bit in I2C_CR1 register, you must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I^2C specification which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. You can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

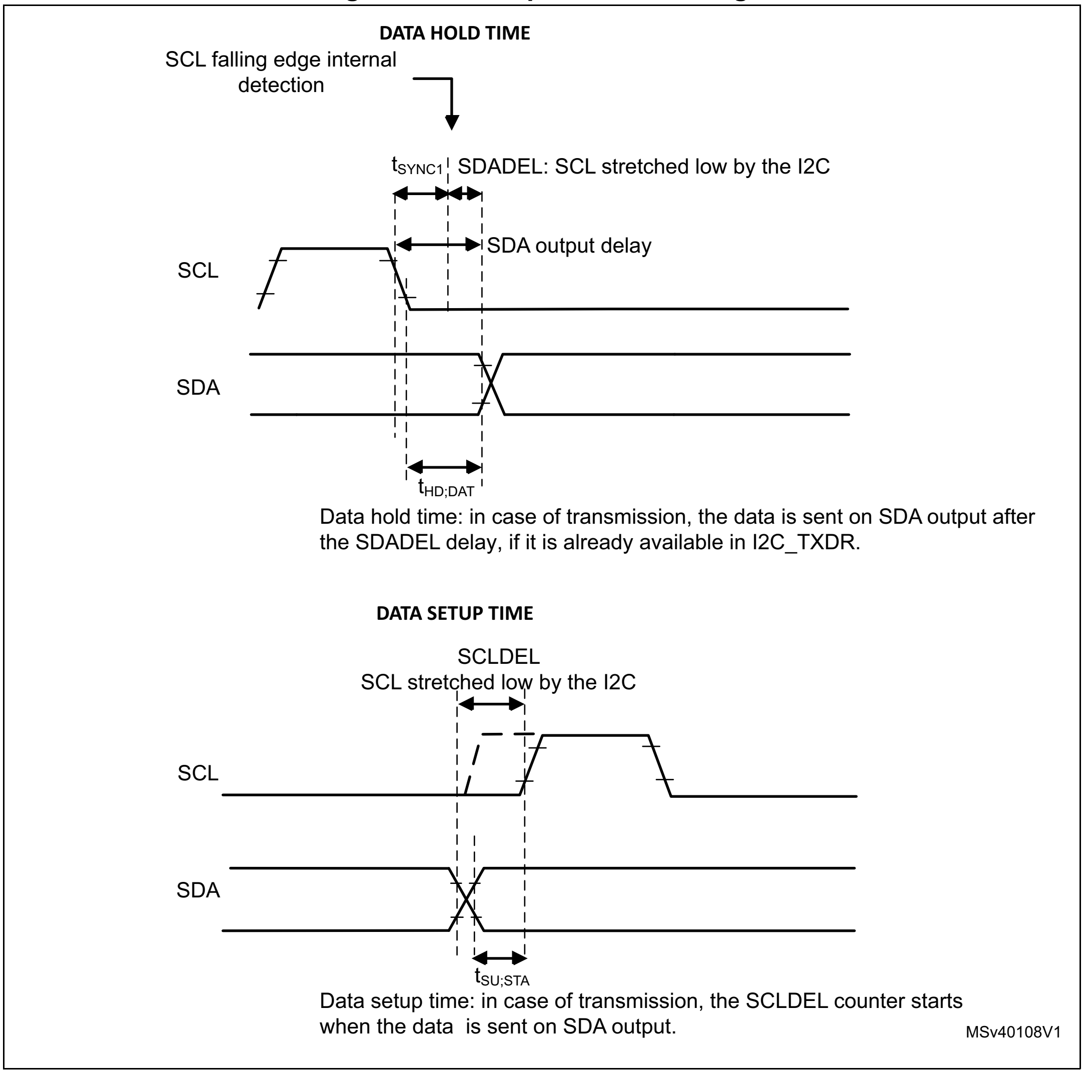
When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows to suppress spikes with a programmable length of 1 to 15 I2CCLK periods.

Caution: Changing the filter configuration is not allowed when the I2C is enabled.

I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in controller and target modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

Figure 166. Setup and hold timings



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is $t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$.

  T_{SDADEL} impacts the hold time $t_{HD;DAT}$.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC+1) + 1] x t_{I2CCLK} }

t_{SYNC1} duration depends on these parameters:

– SCL falling slope
– When enabled, input delay brought by the analog filter: $t_{AF(min)} < t_{AF} < t_{AF(max)}$ ns.
– When enabled, input delay brought by the digital filter: t_{DNF} = DNF x t_{I2CCLK}
– Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, you must program SDADEL in such a way that:

$\{t_{f\ (max)} + t_{HD;DAT\ (min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: $t_{AF(min)}$ / $t_{AF(max)}$ are part of the equation only when the analog filter is enabled. Refer to device datasheet for t_{AF} values.

The maximum $t_{HD;DAT}$ could be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - 260\ ns - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$.

Note: This condition can be violated when NOSTRETCH=0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.

Refer to *Table 71: I2C-SMBUS specification data setup and hold times* for t_f, t_r, $t_{HD;DAT}$ and $t_{VD;DAT}$ standard values.

- After sending SDA output, SCL line is kept at low level during the setup time. This setup time is t_{SCLDEL} = (SCLDEL+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.
  t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), you must program SCLDEL in such a way that:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC+1)] \times t_{I2CCLK}]\} - 1 <= SCLDEL$

Refer to *Table 71: I2C-SMBUS specification data setup and hold times* for t_r and $t_{SU;DAT}$ standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

Table 71. I^{2}C-SMBUS specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBUS		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100		50		250		ns
t_r	Rise time of both SDA and SCL signals	-	1000		300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300		300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is t_{SCLL} = (SCLL+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.
  t_{SCLL} impacts the SCL low time t_{LOW} .
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is t_{SCLH} = (SCLH+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}. t_{SCLH} impacts the SCL high time t_{HIGH} .

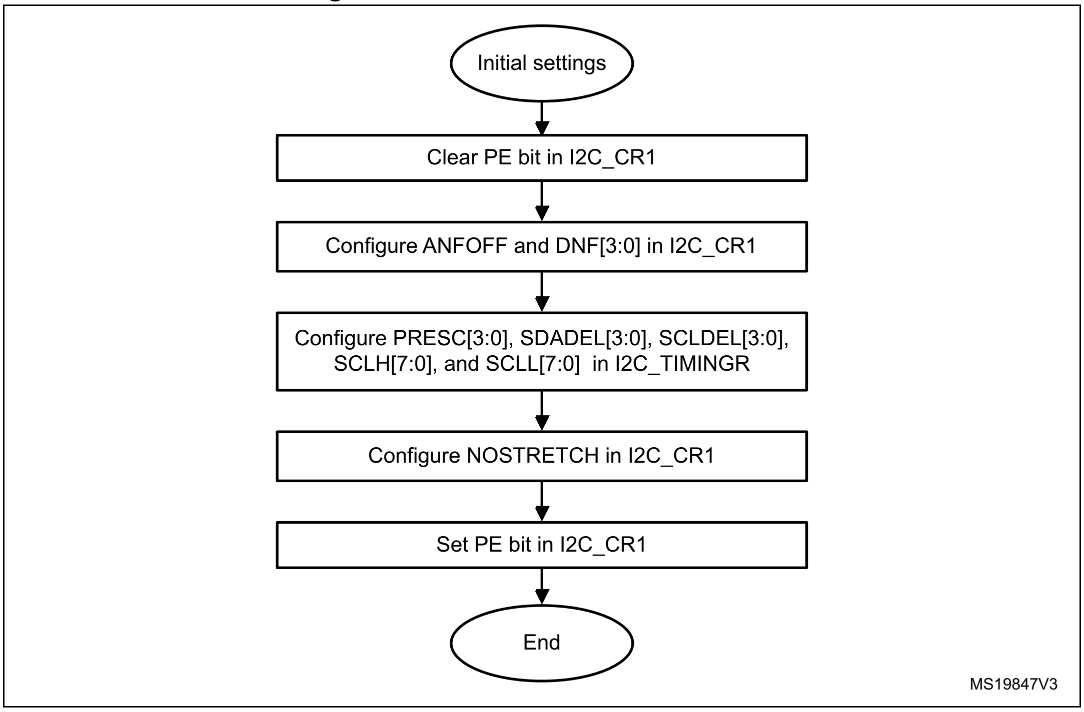
Refer to section *24.4.4: I2C initialization* for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C target NOSTRETCH mode must also be configured before enabling the peripheral. Refer to *: I2C target initialization* for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 167. I2C initialization flowchart



24.4.5 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1.

24.4.6 Data transfer

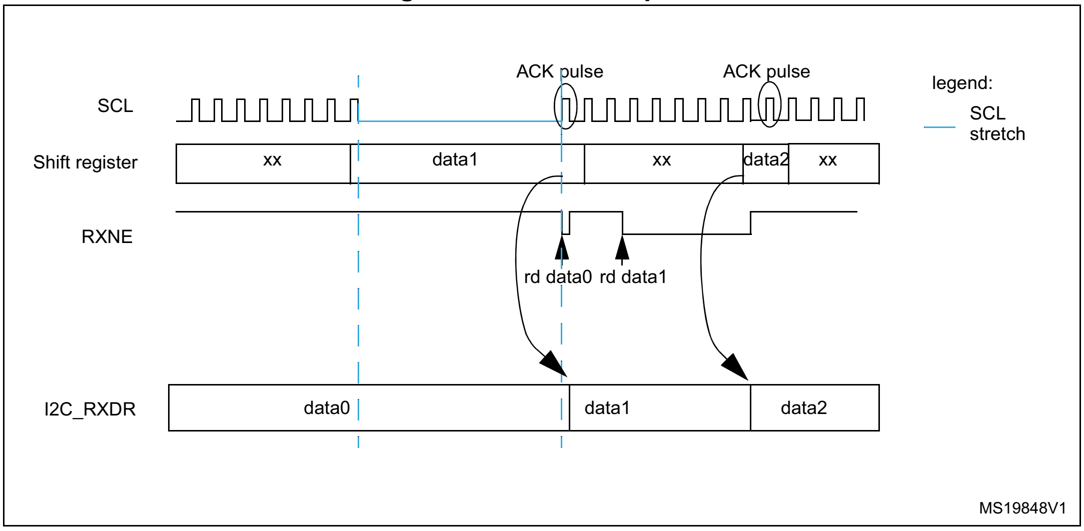
The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If

RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

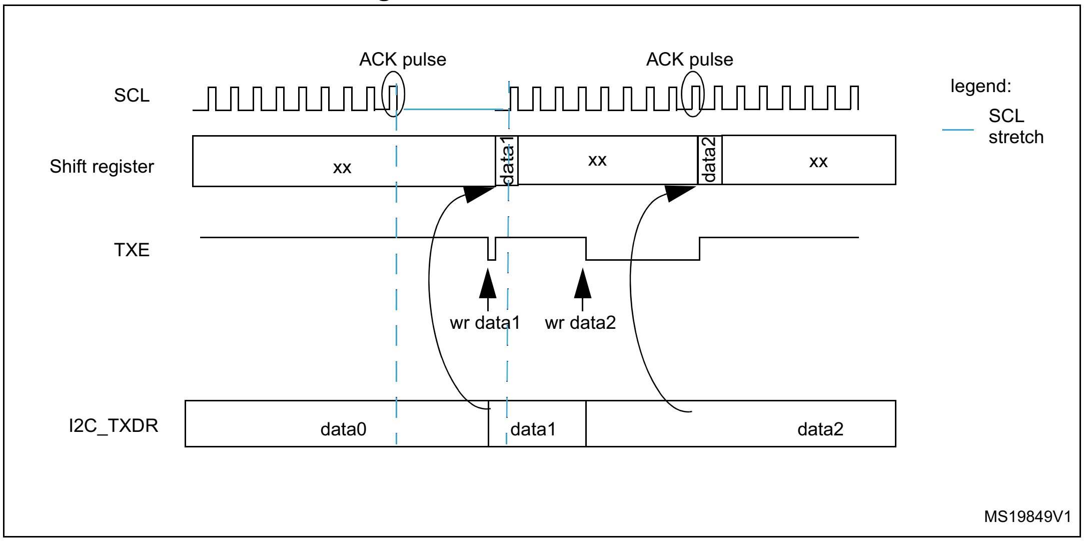
Figure 168. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

Figure 169. Data transmission



Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

– NACK, STOP and ReSTART generation in controller mode
– ACK control in target receiver mode
– PEC generation/checking when SMBus feature is supported

The byte counter is always used in controller mode. By default it is disabled in target mode, but it can be enabled by software by setting the SBC (Target Byte Control) bit in the I2C_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in controller mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = ‘1’ in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.
- **Software end mode** (AUTOEND = ‘0’ in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 72. I2C configuration table

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	x	0	1
Controller Tx/Rx + NBYTES + RESTART	x	0	0
Target Tx/Rx all received bytes ACKed	0	x	x
Target Rx with ACK control	1	1	x

24.4.7 I2C target mode

I2C target initialization

In order to work in target mode, you must enable at least one target address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the target own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.

  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.
- If additional target addresses are required, you can configure the 2nd target address OA2. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.
- The General Call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the controller does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled you must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

Target clock stretching (NOSTRETCH = 0)

In default mode, the I2C target stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled target addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag

is cleared (TXE=1). This stretch is released when the data is written to the I2C_TXDR register.

- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Target Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}.

Target without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C target does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if you clear the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, you ensure that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Target Byte Control Mode

In order to allow byte ACK control in target reception mode, Target Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in target reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. You can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

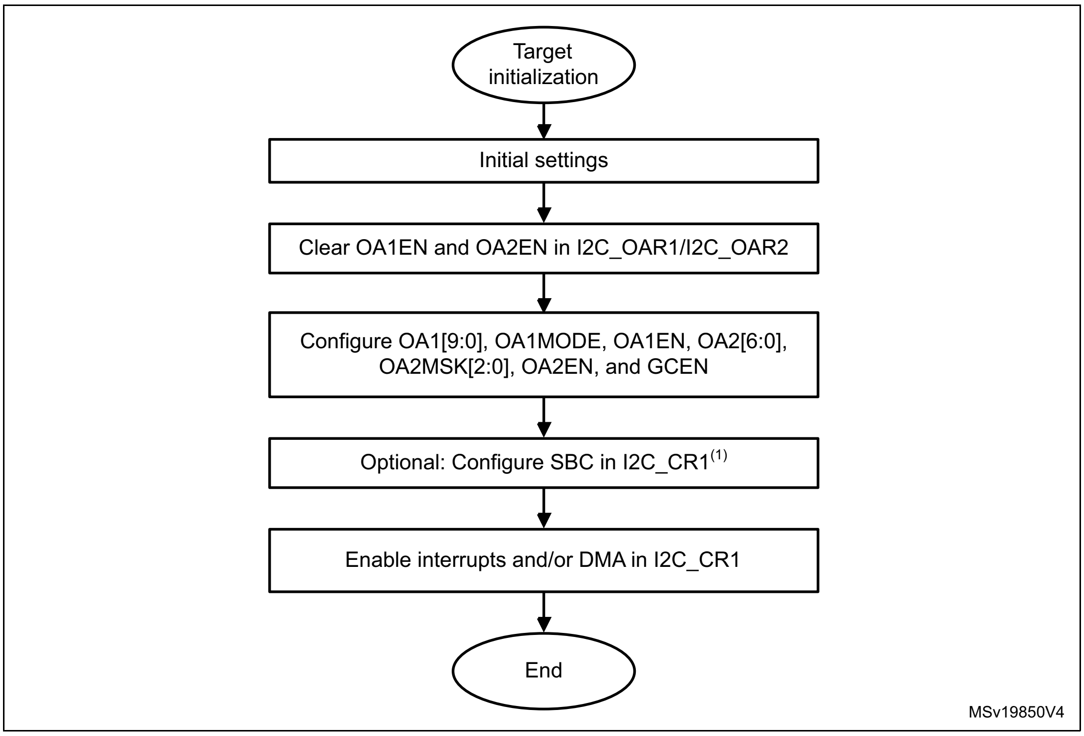
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the target is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

Caution: Target Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

Figure 170. Target initialization flowchart



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The target automatically releases the SCL and SDA lines in order to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to ‘0’. In this case, If TXE = 0 when the target address is received (ADDR=1), you can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Target Byte Control mode (SBC=1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR=1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH=1, the SCL clock is not stretched while the ADDR flag is set, so you cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If you need a TXIS event, (Transmit Interrupt or Transmit DMA request), you must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

Figure 171. Transfer sequence flowchart for I2C target transmitter, NOSTRETCH=0

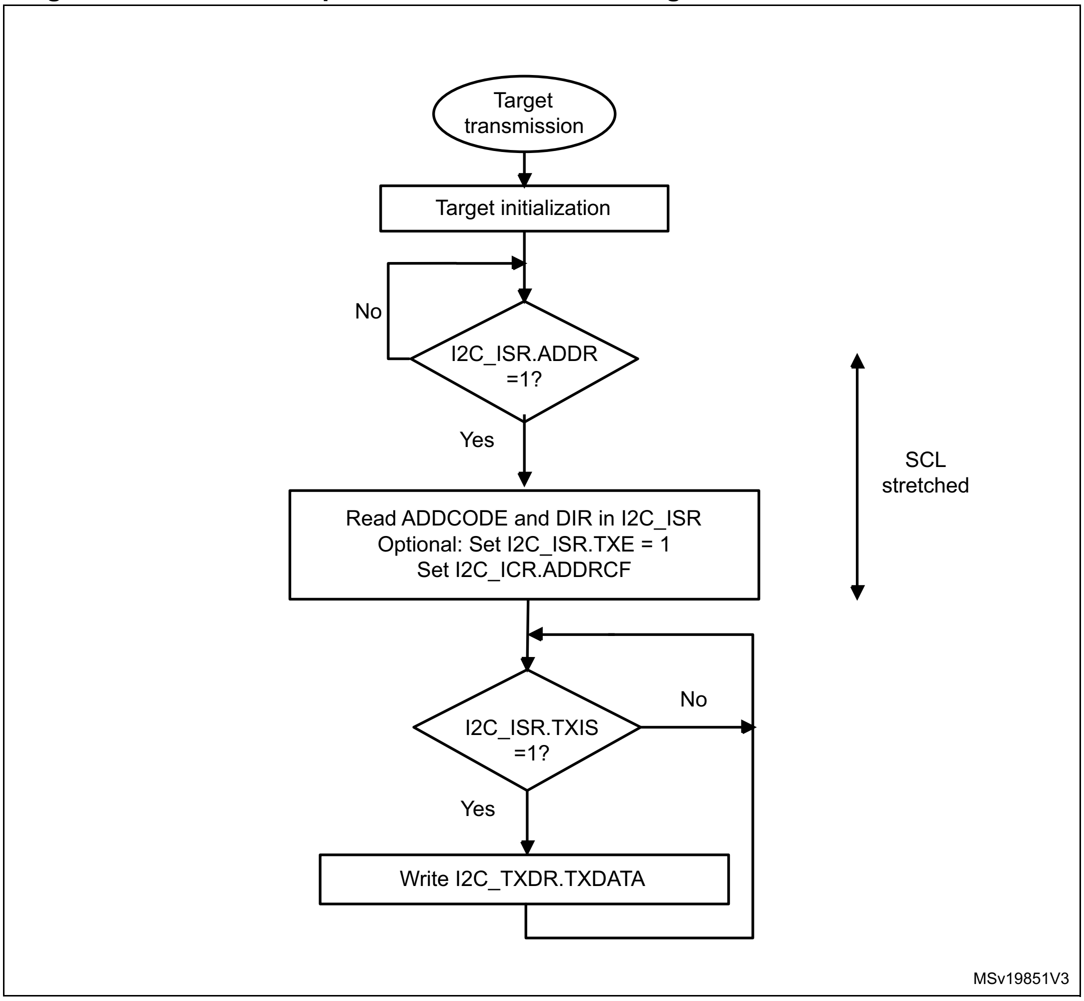


Figure 172. Transfer sequence flowchart for I2C target transmitter, NOSTRETCH=1

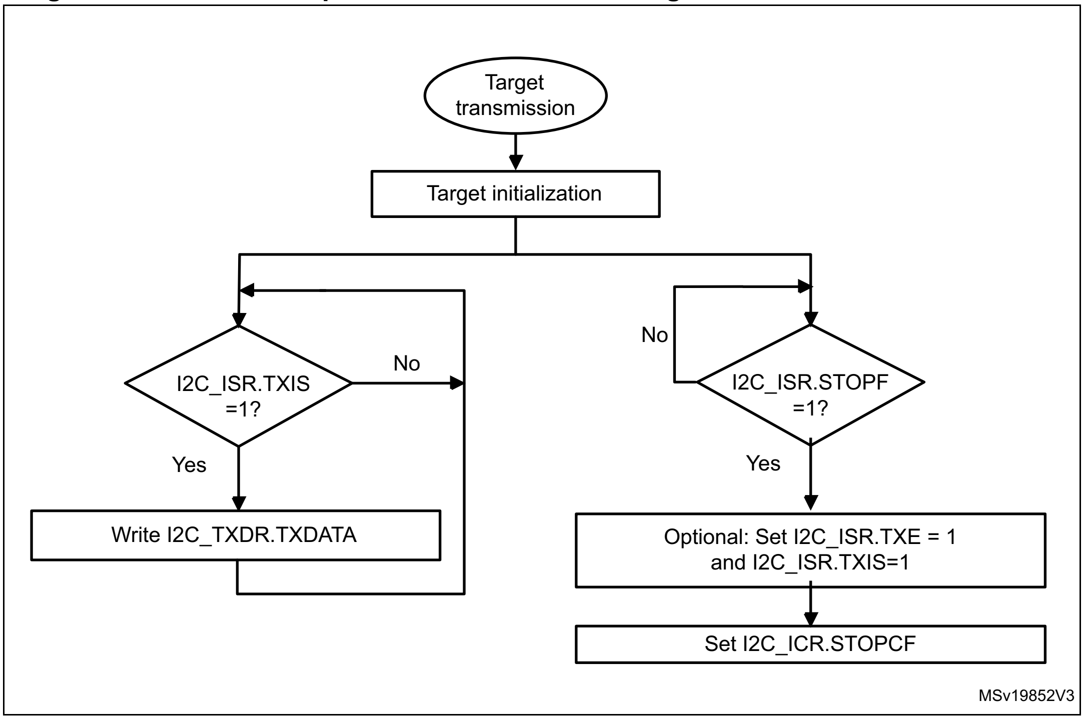
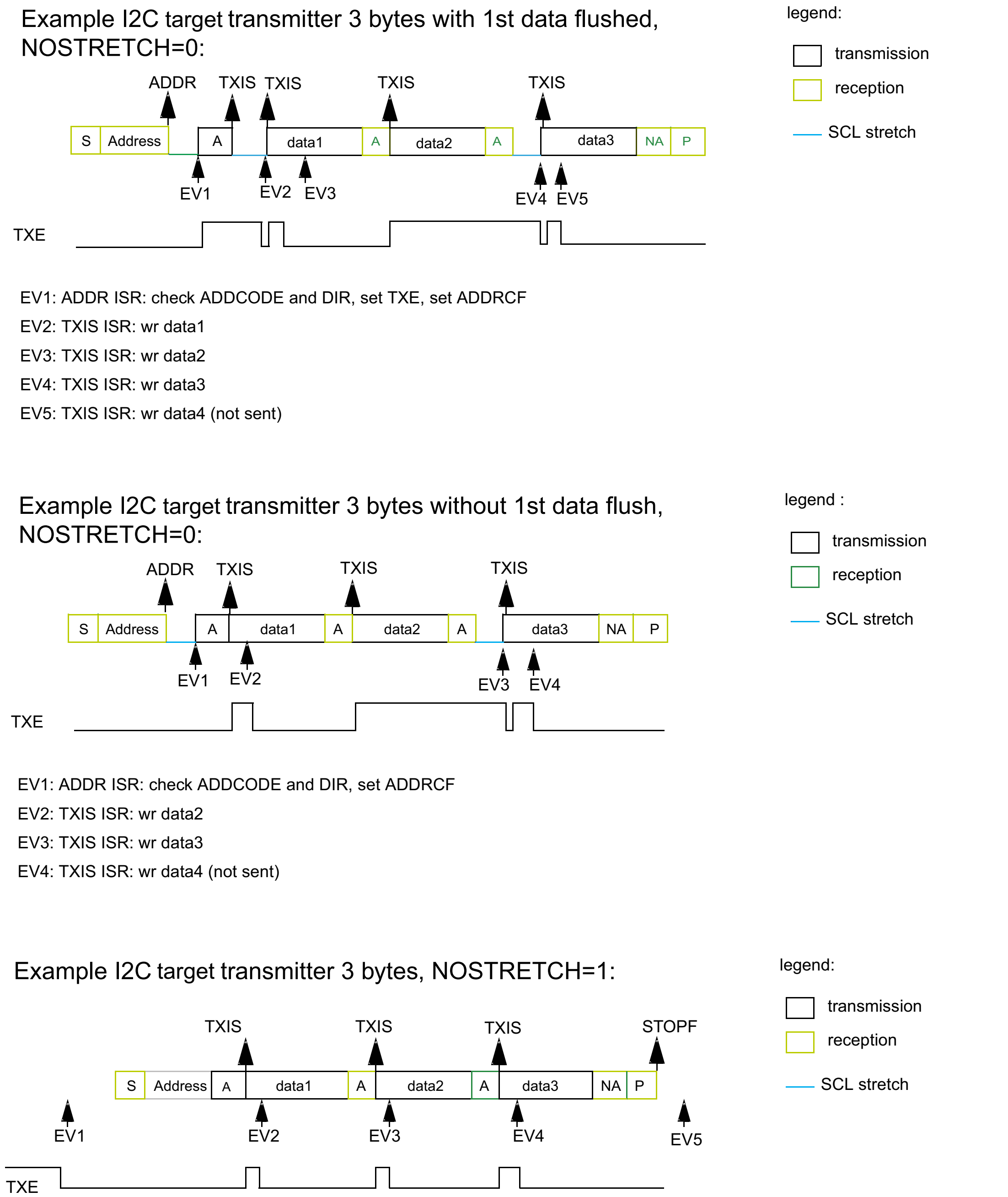


Figure 173. Transfer bus diagrams for I2C target transmitter



EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 174. Transfer sequence flowchart for target receiver with NOSTRETCH=0

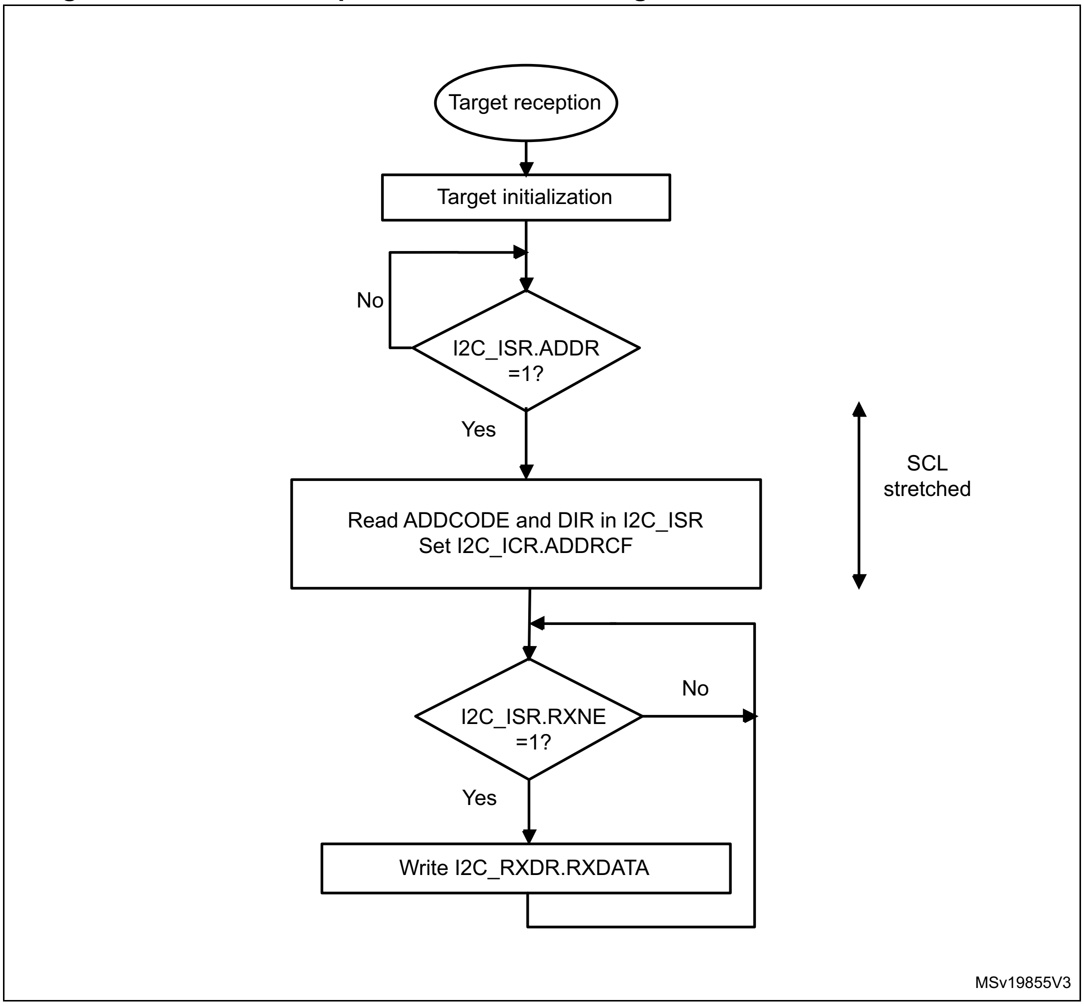


Figure 175. Transfer sequence flowchart for target receiver with NOSTRETCH=1

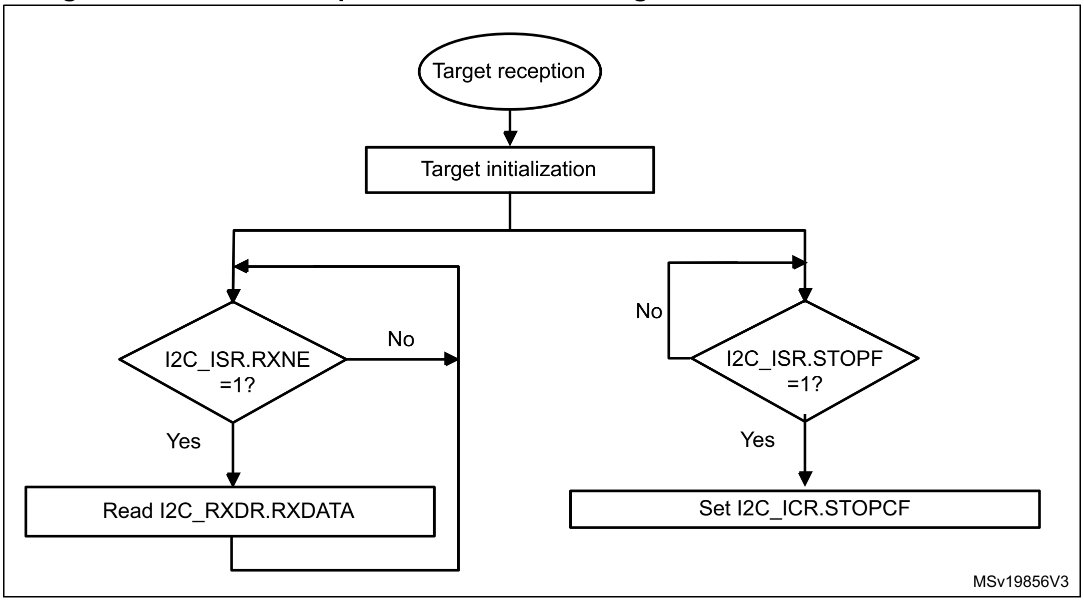
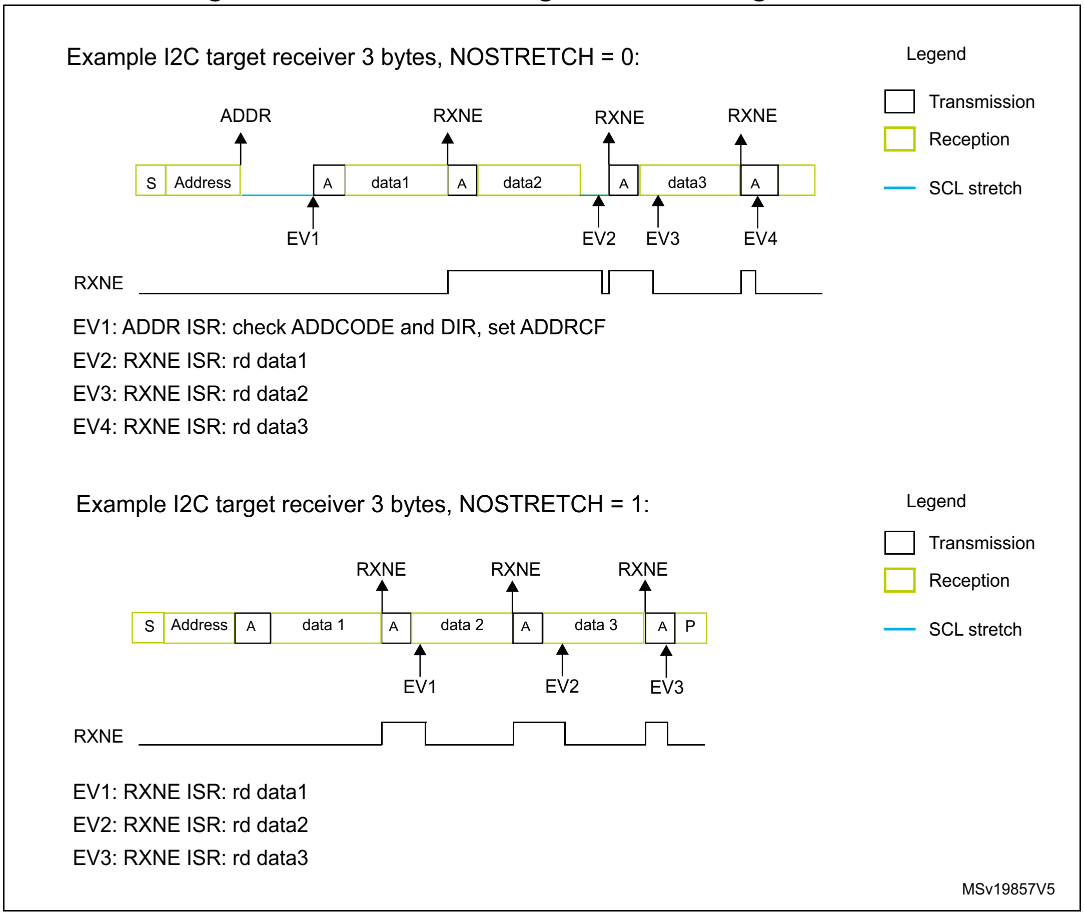


Figure 176. Transfer bus diagrams for I2C target receiver



24.4.8 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

A clock synchronization mechanism is implemented in order to support multi-controller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$

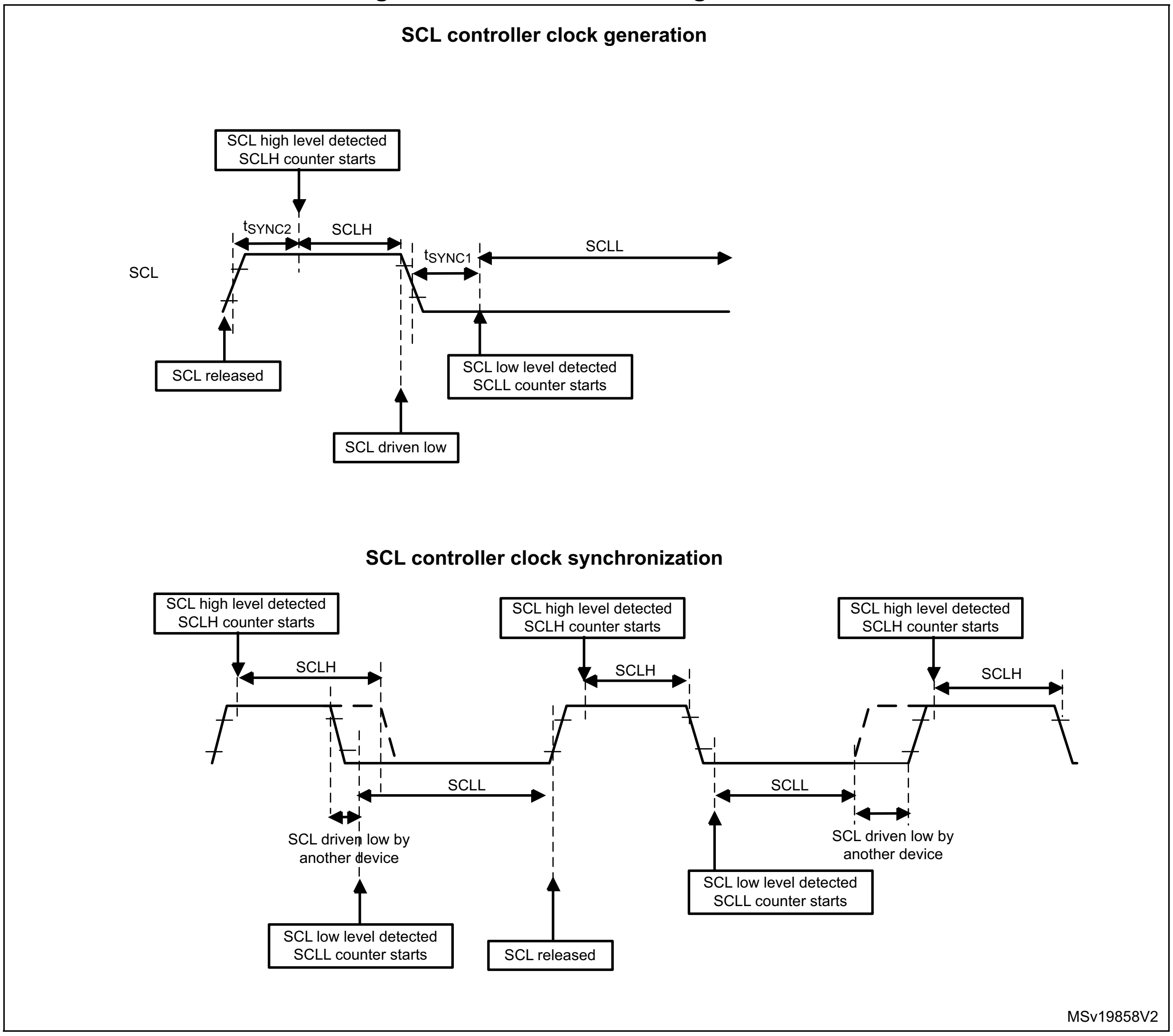
The duration of t_{SYNC1} depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of t_{SYNC2} depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

Figure 177. Controller clock generation



Caution: In order to be I^2C or SMBus compliant, the controller clock must respect the timings given below:

Table 73. I^2C-SMBUS specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBUS		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100		400		1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	µs
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	µs
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	µs
t_{HIGH}	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	µs
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	ns

Note: SCLL is also used to generate the t_{BUF} and $t_{SU:STA}$ timings.

SCLH is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 24.4.9: I2C_TIMINGR register configuration examples* for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

Controller communication initialization (address phase)

In order to initiate the communication, you must program the following parameters for the addressed target in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Target address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

You must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

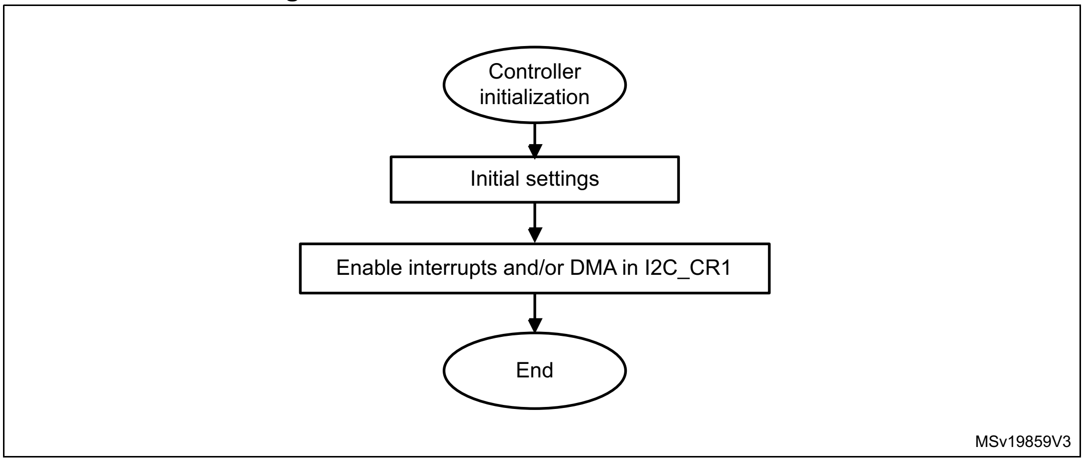
Then the controller automatically sends the START condition followed by the target address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t_{BUF}.

In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs. If the I2C is addressed as a target (ADDR=1) while the START bit is set, the I2C switches to target mode and the START bit is cleared when the ADDRCF bit is set.

Note: The same procedure is applied for a Repeated Start condition. In this case BUSY=1.

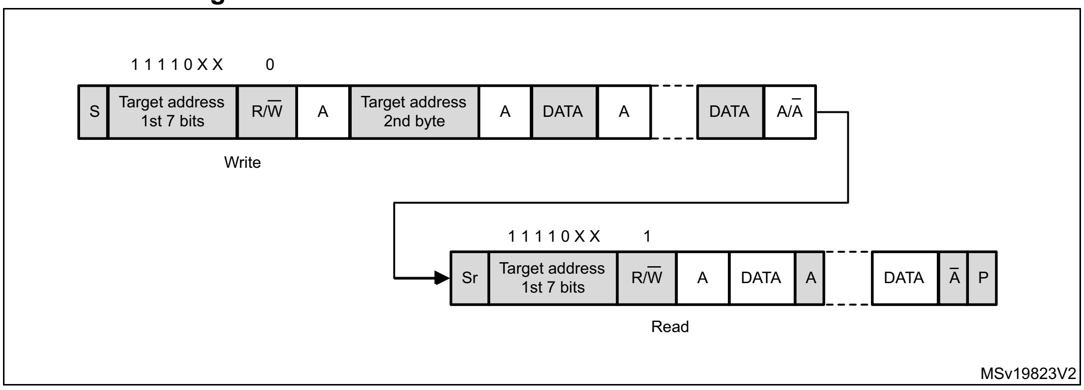
Figure 178. Controller initialization flowchart



Initialization of a controller receiver addressing a 10-bit address target

- If the target address is in 10-bit format, you can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the controller automatically sends the following complete sequence after the START bit is set: (Re)Start + Target address 10-bit header Write + Target address 2nd byte + REStart + Target address 10-bit header Read
- If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated start is set with the 10 bit target address configured with HEAD10R=1. In this case the controller sends this sequence: ReStart + Target address 10-bit header Read

Figure 179. 10-bit address read access with HEAD10R=1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper target address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.

Figure 180. Transfer sequence flowchart for I2C controller transmitter for N≤255 bytes

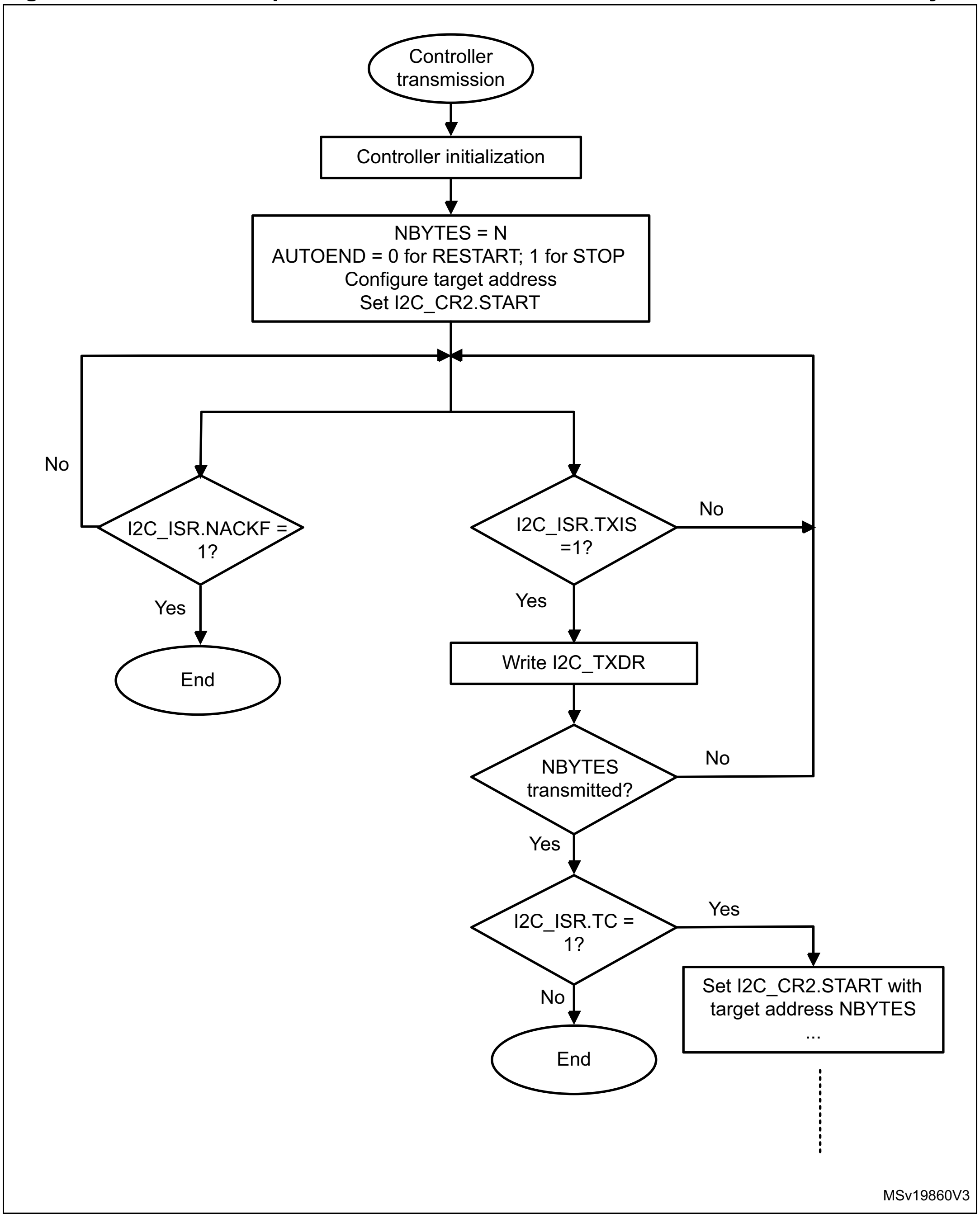


Figure 181. Transfer sequence flowchart for I2C controller transmitter for N>255 bytes

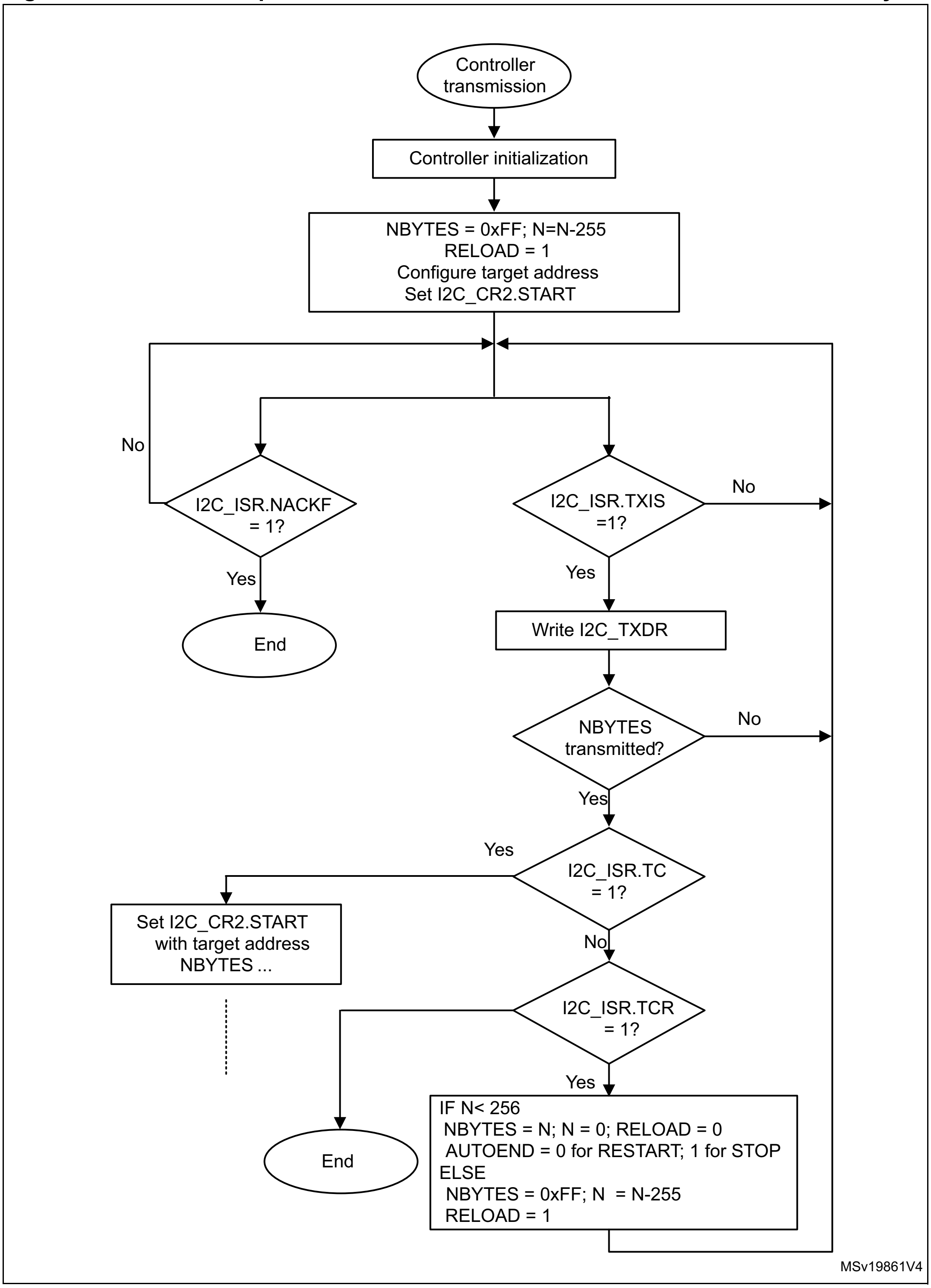
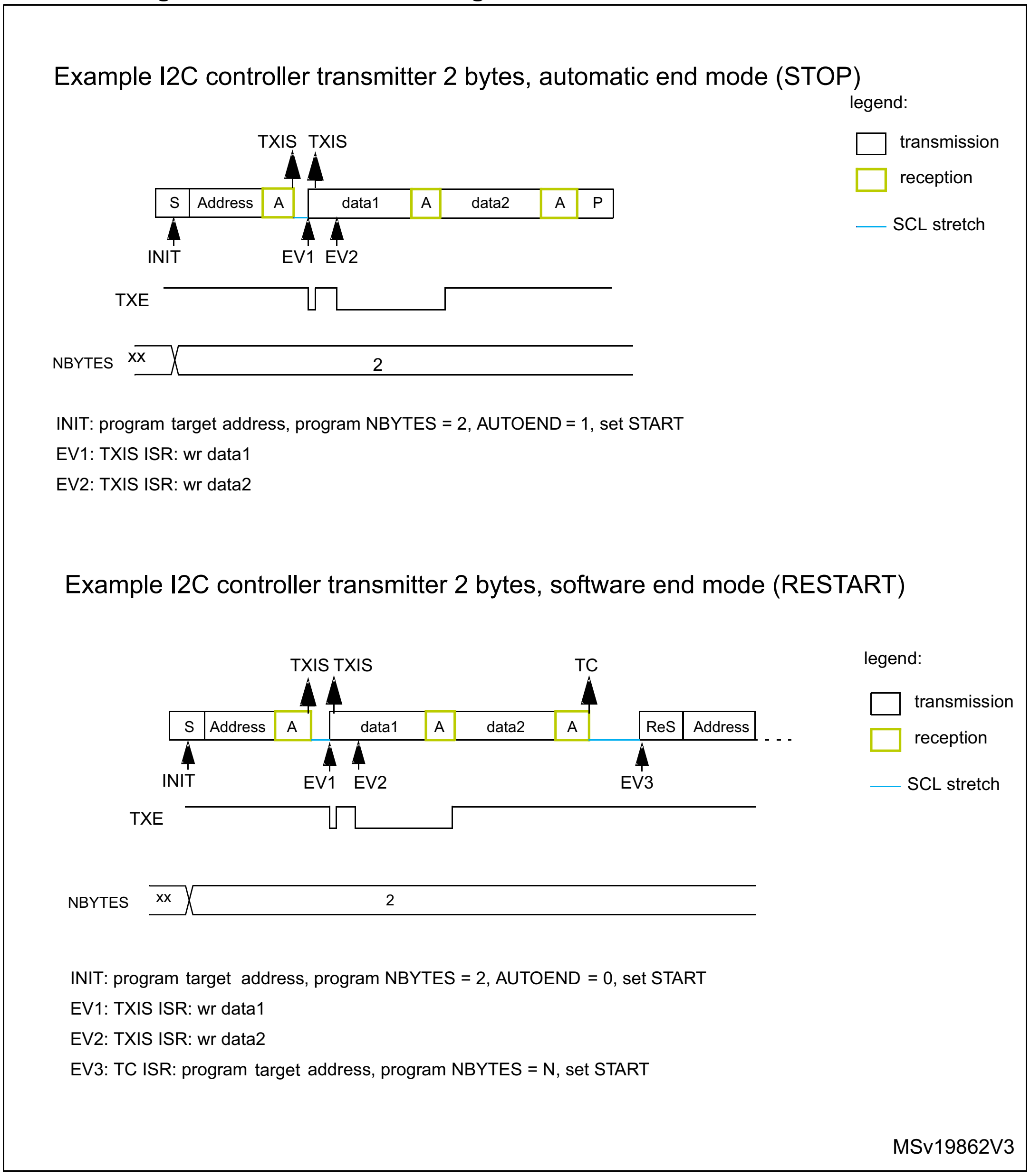


Figure 182. Transfer bus diagrams for I2C controller transmitter



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper target address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by target address, are sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 183. Transfer sequence flowchart for I2C controller receiver for N ≤ 255 bytes

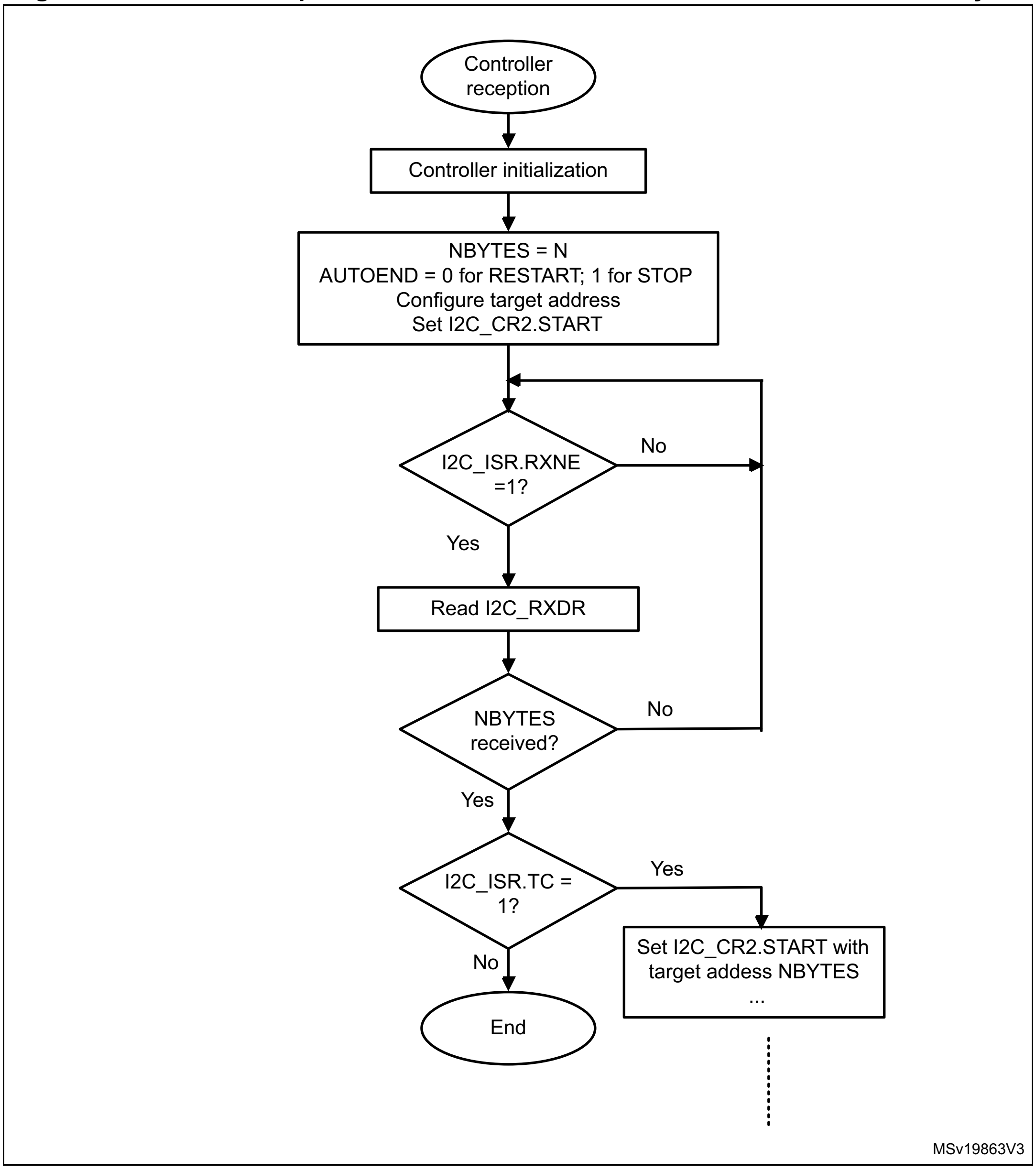


Figure 184. Transfer sequence flowchart for I2C controller receiver for N >255 bytes

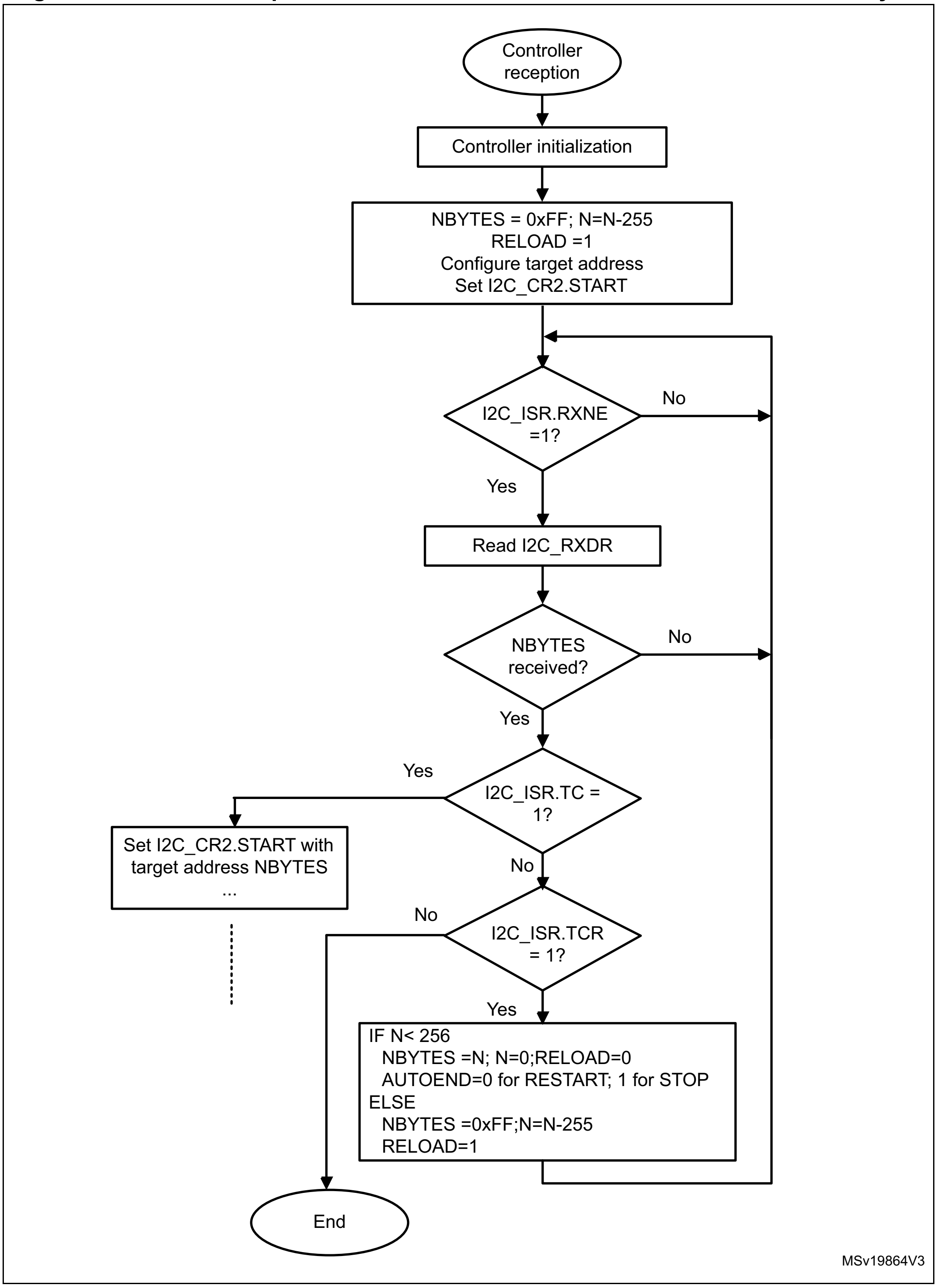
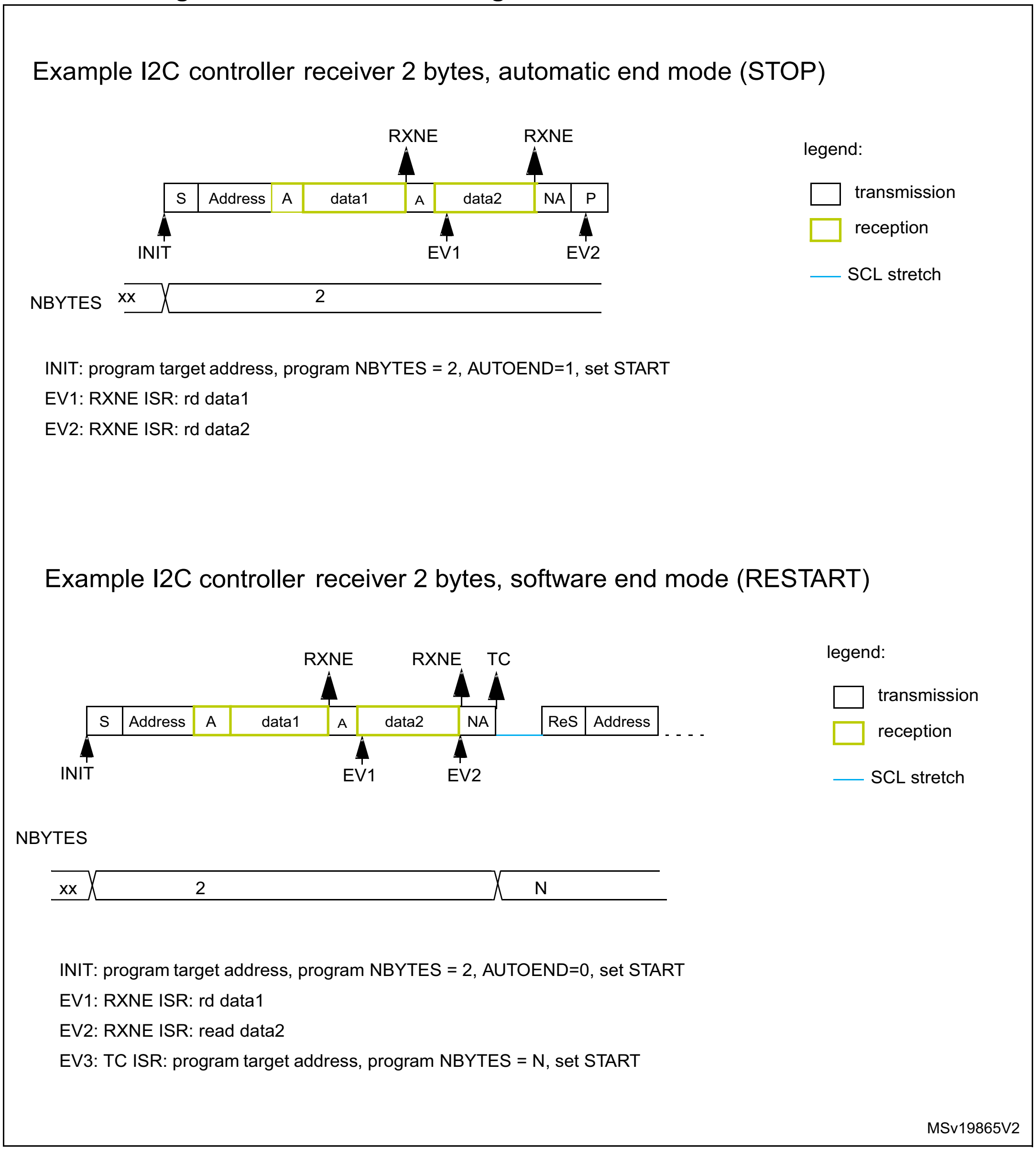


Figure 185. Transfer bus diagrams for I2C controller receiver



24.4.9 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I^2C specification.

Table 74. Examples of timings settings for f_{I2CCLK} = 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~1000 ns(4)
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns

24.4.10 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to *Section 24.3: I2C implementation*.

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBUS specification rev 2.0 (http://smbus.org).

The System Management Bus Specification refers to three types of devices.

- A target is a device that receives or responds to a command.
- A controller is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized controller that provides the main interface to the system's CPU. A host must be a controller-target and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as controller or target device, and also as a host.

SMBUS is based on I^2C specification rev 2.1.

Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification version 2.0 (http://smbus.org).

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification version 2.0 (http://smbus.org).

Received Command and Data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the Target Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to *SMBus target mode on page 608* section for more details.

Host Notify protocol

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus Host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The SMBus ALERT optional signal is supported. A target-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the Alert Response Address.

When configured as a target device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0*.*

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x_8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification version 2.0.

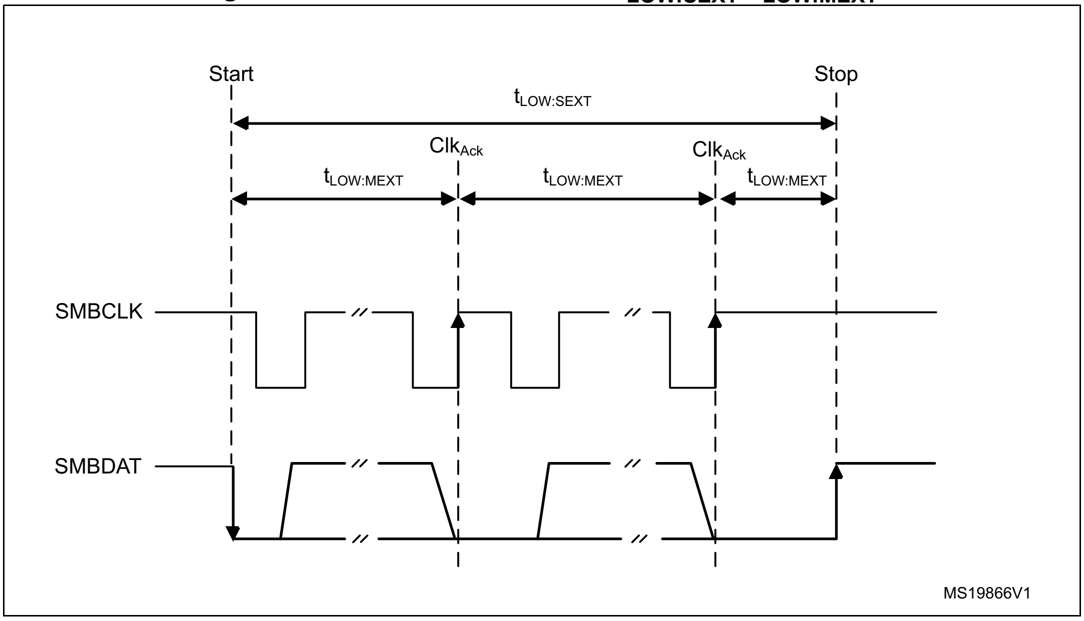
Table 75. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	ms
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	ms

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another target device or the controller also extend the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. Therefore, this parameter is measured with the target device as the sole target of a full-speed controller.

2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extend the clock causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed target device as the sole target of the controller.

Figure 186. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for t_{IDLE} greater than $t_{HIGH,MAX}$. (refer to *Table 73: I2C-SMBUS specification clock timings*)

This timing parameter covers the condition where a controller has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

24.4.11 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to *Section 24.3: I2C implementation*.

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

Received Command and Data Acknowledge control (Target mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in target mode, the Target Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to *SMBus target mode on page 608* for more details.

Specific address (Target mode)

The specific SMBus addresses should be enabled if needed. Refer to *Bus idle detection on page 605* for more details.

- The SMBus Device Default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus Host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The Alert Response Address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after NBYTES-1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C is enabled.

Table 76. SMBUS with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
Target Tx/Rx with PEC	1	0	x	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification version 2.0.

- $t_{TIMEOUT}$ check

  In order to enable the $t_{TIMEOUT}$ check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the $t_{TIMEOUT}$ parameter. The TIDLE bit must be configured to ‘0’ in order to detect the SCL low level timeout.

  Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register.

  If SCL is tied low for a time greater than (TIMEOUTA+1) x 2048 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 77: Examples of TIMEOUTA settings (max $t_{TIMEOUT}$ = 25 ms)*.

Caution: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

  Depending on if the peripheral is configured as a controller or as a target, The 12-bit TIMEOUTB timer must be configured in order to check $t_{LOW:SEXT}$ for a target and

$t_{LOW:MEXT}$ for a controller. As the standard specifies only a maximum, you can choose the same value for the both.

Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TIMEOUTB+1) x 2048 x t_{I2CCLK}, and in the timeout interval described in *Bus idle detection on page 605* section, the TIMEOUT flag is set in the I2C_ISR register.

Refer to *Table 78: Example of TIMEOUTB settings*

Caution: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

Bus Idle detection

In order to enable the t_{IDLE} check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the t_{IDLE} parameter. The TIDLE bit must be configured to '1 in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than (TIMEOUTA+1) x 4 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

Refer to *Table 79: Example of TIMEOUTA settings (max t_{IDLE} = 50 µs)*

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

24.4.12 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 24.3: I2C implementation*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 77. Examples of TIMEOUTA settings (max $t_{TIMEOUT}$ = 25 ms)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms

- Configuring the maximum duration of $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ to 8 ms:

Table 78. Example of TIMEOUTB settings

f_{I2CCLK}	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 µs

Table 79. Example of TIMEOUTA settings (max t_{IDLE} = 50 µs)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	t_{TIDLE}
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs

24.4.13 SMBus target mode

This section is relevant only when SMBus feature is supported. Refer to *Section 24.3: I2C implementation*.

In addition to 2C target transfer management (refer to *Section 24.4.7: I2C target mode*) some additional software flowcharts are provided to support SMBus.

SMBus Target transmitter

When the IP is used in SMBus, SBC must be programmed to ‘1’ in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts are NBYTES-1 and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the NBYTES-1 data transfer.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 187. Transfer sequence flowchart for SMBus target transmitter N bytes + PEC

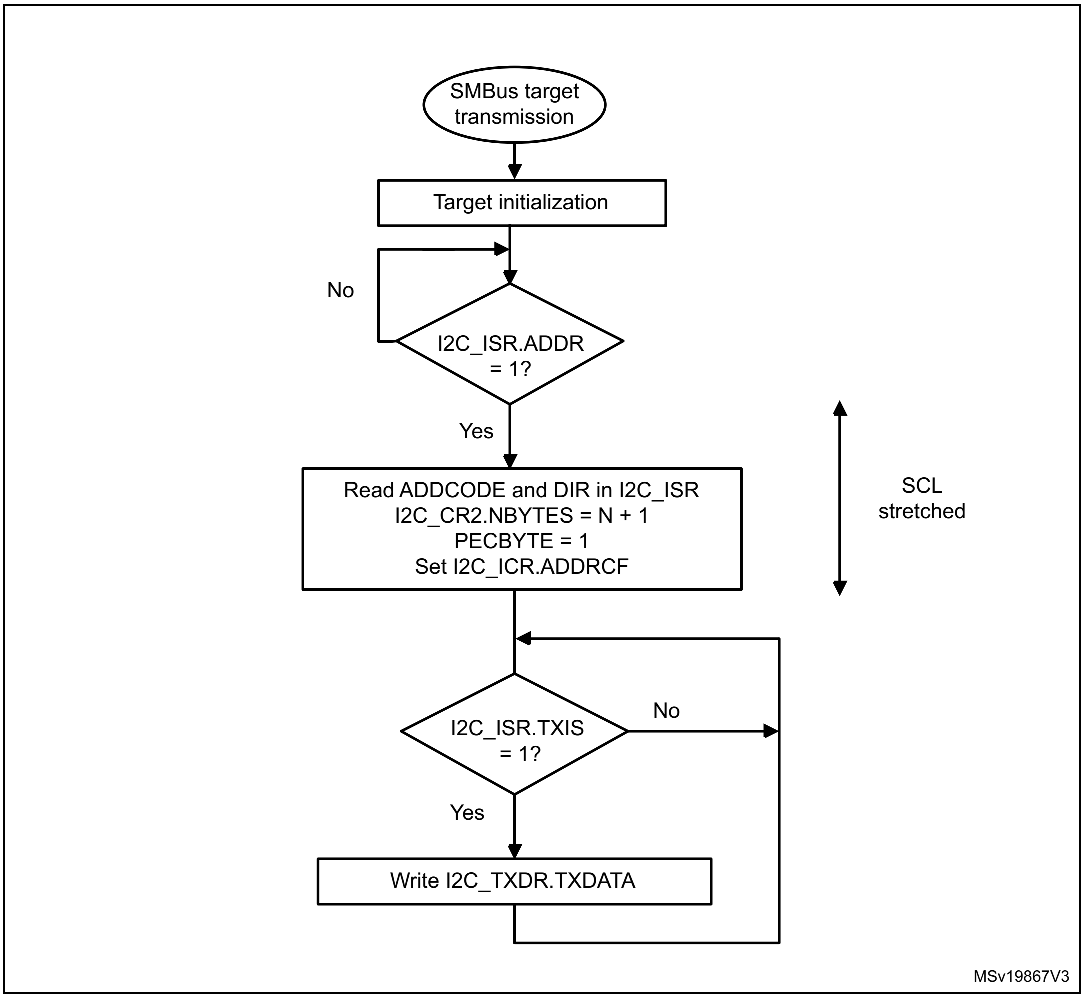
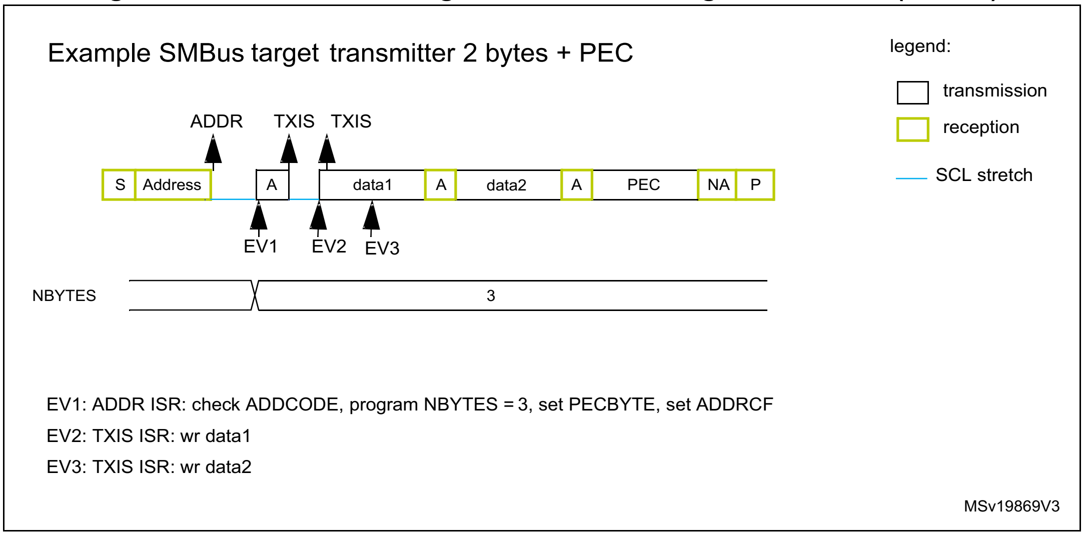


Figure 188. Transfer bus diagrams for SMBus target transmitter (SBC=1)



SMBus Target receiver

When the I2C is used in SMBus mode, SBC must be programmed to ‘1’ in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to *Target Byte Control Mode on page 583* for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES-1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, you can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES-1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 189. Transfer sequence flowchart for SMBus target receiver N Bytes + PEC

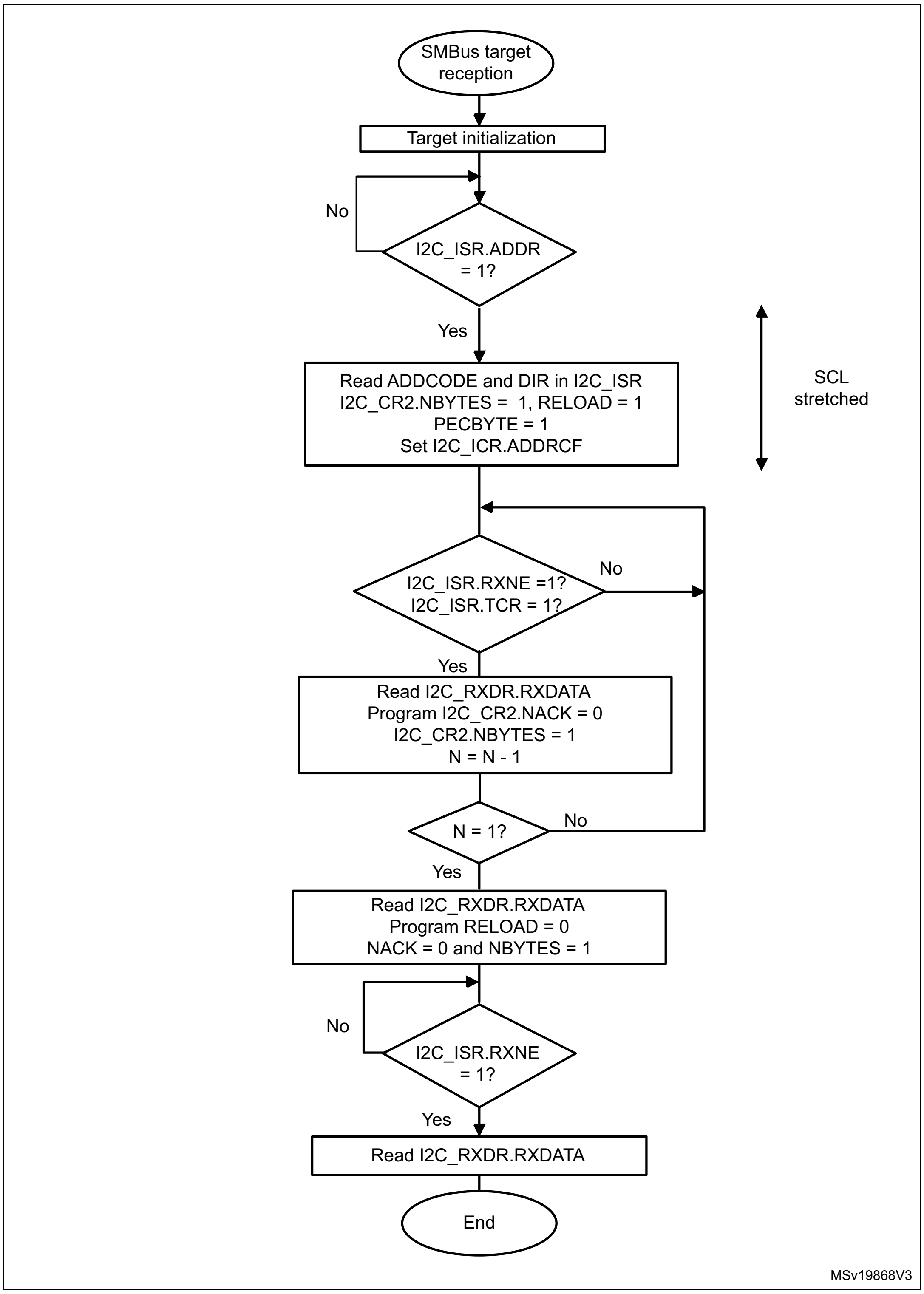
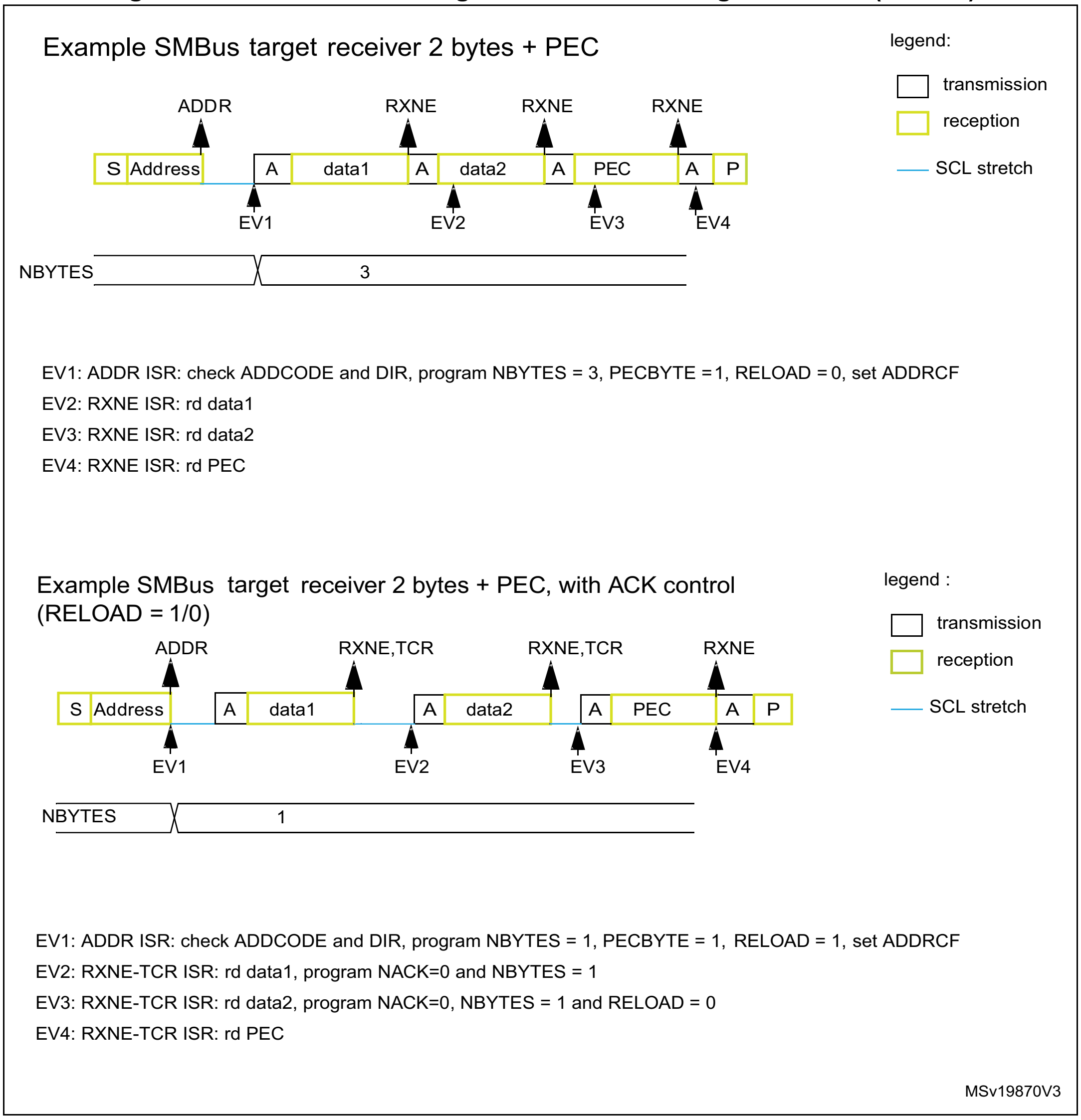


Figure 190. Bus transfer diagrams for SMBus target receiver (SBC=1)



This section is relevant only when SMBus feature is supported. Refer to *Section 24.3: I2C implementation*.

In addition to I2C controller transfer management (refer to *Section 24.4.8: I2C controller mode*) some additional software flowcharts are provided to support SMBus.

SMBus Controller transmitter

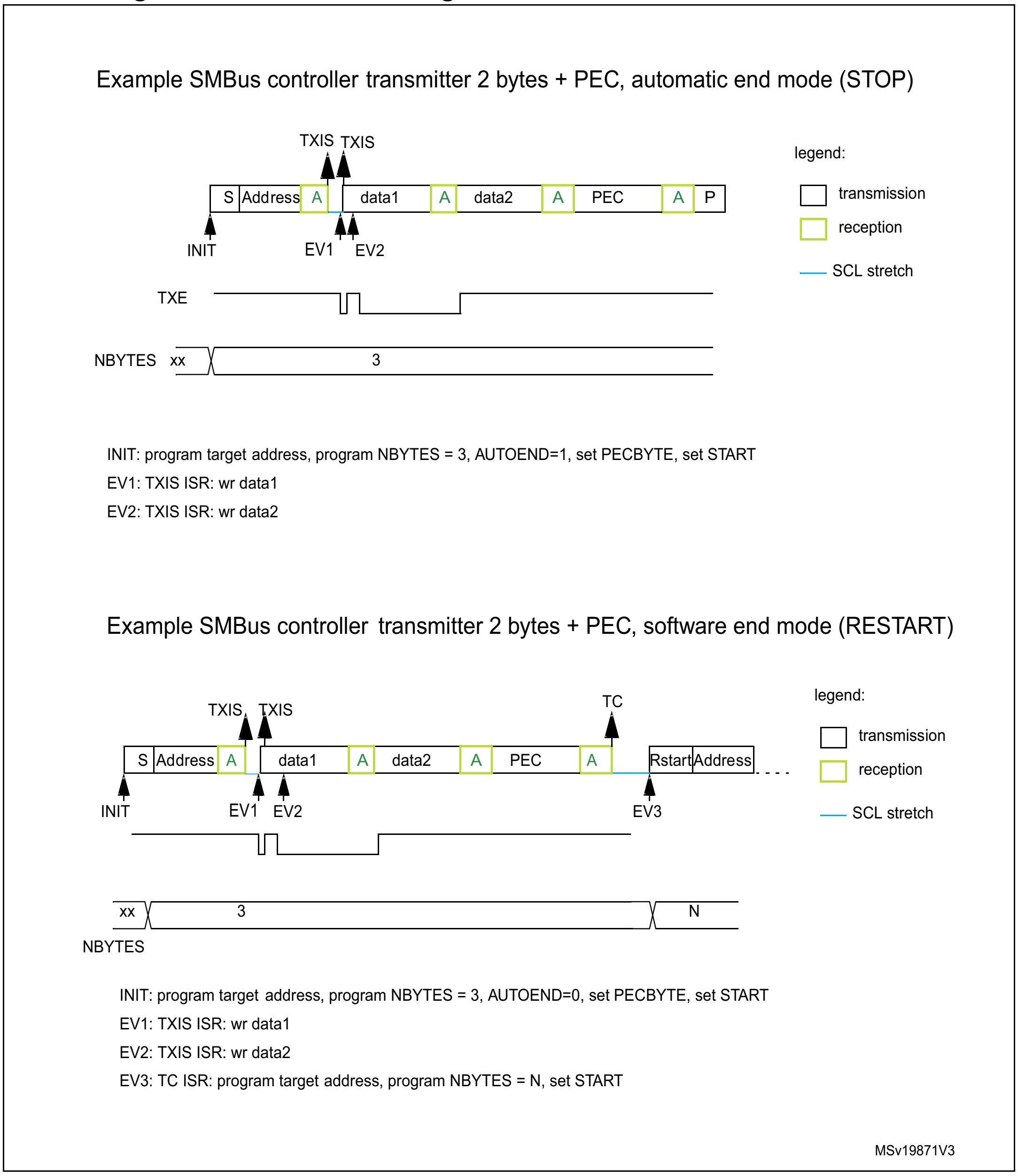
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, automatic end mode should be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 191. Bus transfer diagrams for SMBus controller transmitter



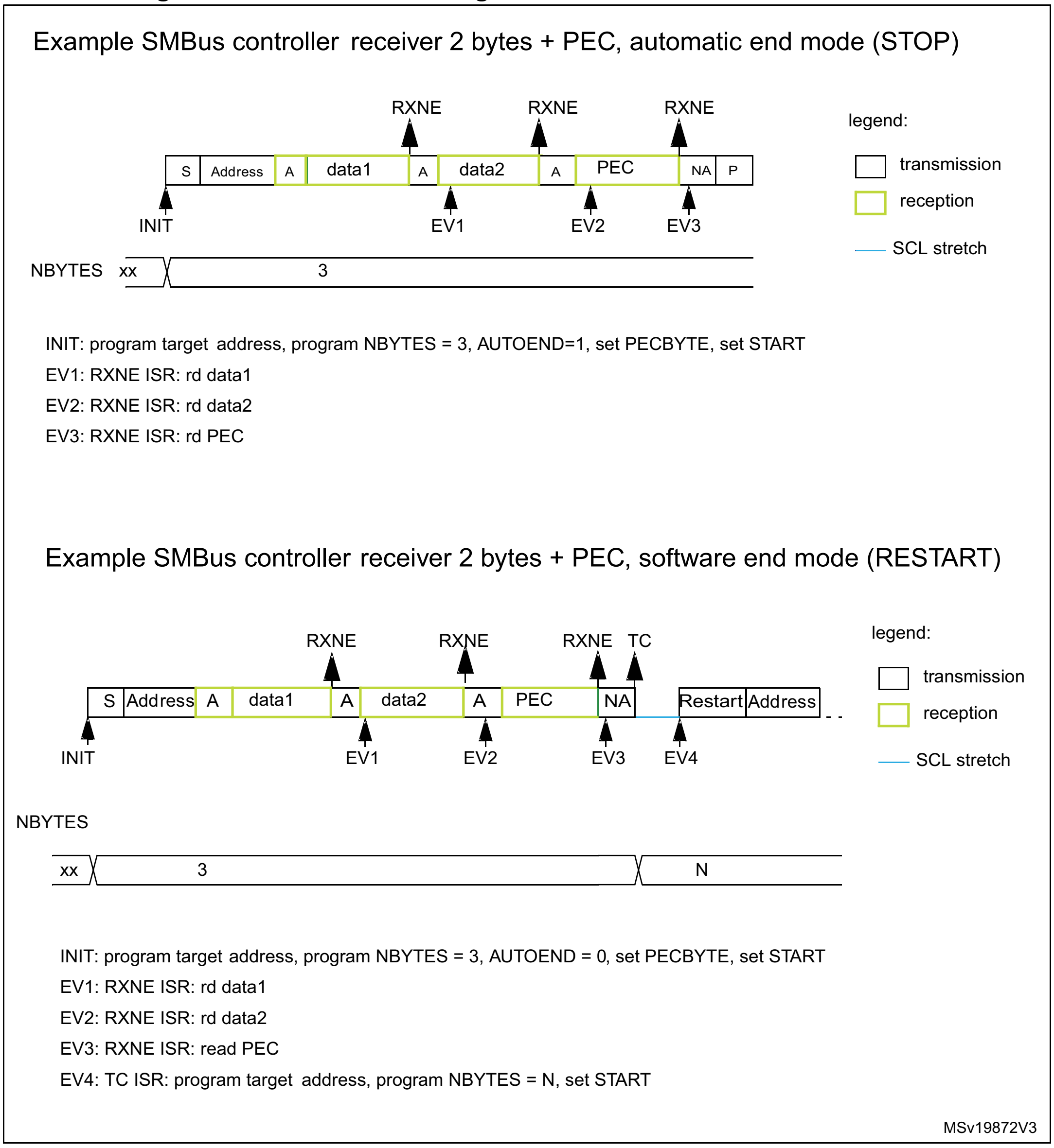
SMBus Controller receiver

When the SMBus controller wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the target address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the target address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 192. Bus transfer diagrams for SMBus controller receiver



24.4.14 Error conditions

The following are the error conditions which may cause communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as controller or addressed target (that is,not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.
- In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte should be sent. The content of the I2C_TXDR register is sent if TXE=0, 0xFF if not.
  - When a new byte should be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Packet Error Checking Error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to *Section 24.3: I2C implementation*.

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Timeout Error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 24.3: I2C implementation*.

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{LOW:MEXT}$ parameter)
- Target cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus $t_{LOW:SEXT}$ parameter)

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 24.3: I2C implementation*.

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

24.4.15 DMA requests

Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see *Section 10: DMA controller (DMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In controller mode: the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter on page 594*.

- In target mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to *SMBus Target transmitter on page 608* and *SMBus Controller transmitter on page 612*.

Note: If DMA is used for transmission, the TXIE bit does not need to be enabled.

Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to *Section 10: DMA controller (DMA) on page 220*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.
- In target mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see *Section 24.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to *SMBus Target receiver on page 610* and *SMBus Controller receiver on page 613*.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

24.5 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 80. I2C Interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Receive buffer not empty	RXNE	Read I2C_RXDR register	RXIE
Transmit buffer interrupt status	TXIS	Write I2C_TXDR register	TXIE
Stop detection interrupt flag	STOPF	Write STOPCF=1	STOPIE

Table 80. I2C Interrupt requests (continued)

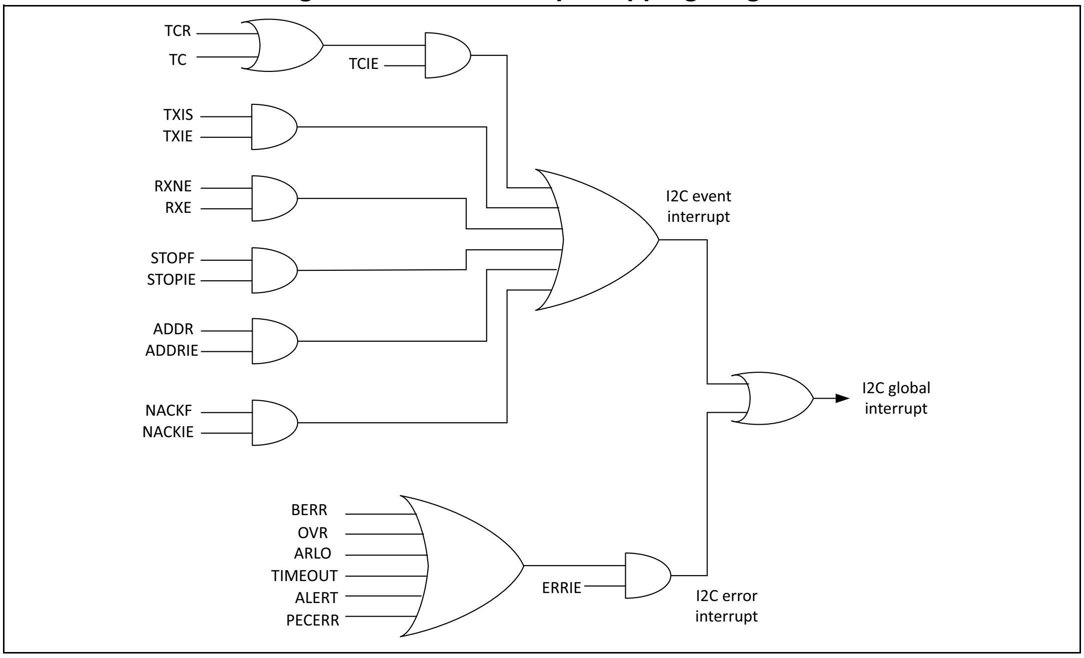
Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Transfer Complete Reload	TCR	Write I2C_CR2 with NBYTES[7:0] ≠ 0	TCIE
Transfer complete	TC	Write START=1 or STOP=1	
Address matched	ADDR	Write ADDRCF=1	ADDRIE
NACK reception	NACKF	Write NACKCF=1	NACKIE
Bus error	BERR	Write BERRCF=1	ERRIE
Arbitration loss	ARLO	Write ARLOCF=1	
Overrun/Underrun	OVR	Write OVRCF=1	
PEC error	PECERR	Write PECERRCF=1	
Timeout/t_{LOW} error	TIMEOUT	Write TIMEOUTCF=1	
SMBus Alert	ALERT	Write ALERTCF=1	

Depending on the product implementation, all these interrupts events can either share the same interrupt vector (I2C global interrupt), or be grouped into 2 interrupt vectors (I2C event interrupt and I2C error interrupt). Refer to *Section 2.3.2: Interrupts* for details.

In order to enable the I2C interrupts, the following sequence is required:

1. Configure and enable the I2C IRQ channel in the NVIC.
2. Configure the I2C to generate interrupts.

Figure 193. I2C interrupt mapping diagram



24.6 I2C registers

Refer to *Section 1.3: Acronyms* for a list of abbreviations used in register descriptions.

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

24.6.1 Control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	Res.	NOSTRETCH	SBC
								rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **PECEN:** PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 24.3: I2C implementation.

Bit 22 **ALERTEN**: SMBus alert enable

Device mode (SMBHEN=0):

0: Releases SMBA pin high and Alert Response Address Header disabled: 0001100x followed by NACK.

1: Drives SMBA pin low and Alert Response Address Header enables: 0001100x followed by ACK.

Host mode (SMBHEN=1):

0: SMBus Alert pin (SMBA) not supported.

1: SMBus Alert pin (SMBA) supported.

Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 24.3: I2C implementation.

Bit 21 **SMBDEN**: SMBus Device Default address enable

0: Device default address disabled. Address 0b1100001x is NACKed.

1: Device default address enabled. Address 0b1100001x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 24.3: I2C implementation.

Bit 20 **SMBHEN**: SMBus Host address enable

0: Host address disabled. Address 0b0001000x is NACKed.

1: Host address enabled. Address 0b0001000x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 24.3: I2C implementation.

Bit 19 **GCEN**: General call enable

0: General call disabled. Address 0b00000000 is NACKed.

1: General call enabled. Address 0b00000000 is ACKed.

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.

0: Clock stretching enabled

1: Clock stretching disabled

Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 16 **SBC**: Target byte control

This bit is used to enable hardware byte control in target mode.

0: Target byte control disabled

1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

0: DMA mode disabled for reception

1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable

0: DMA mode disabled for transmission

1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF

0: Analog noise filter enabled

1: Analog noise filter disabled

Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter

These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}

0000: Digital filter disabled

0001: Digital filter enabled and filtering capability up to 1 t_{I2CCLK}

...

1111: digital filter enabled and filtering capability up to15 t_{I2CCLK}

Note: If the analog filter is also enabled, the digital filter is added to the analog filter. This filter can only be programmed when the I2C is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generate an interrupt:
Arbitration Loss (ARLO)
Bus Error detection (BERR)
Overrun/Underrun (OVR)
Timeout detection (TIMEOUT)
PEC error detection (PECERR)
Alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer Complete interrupt enable
0: Transfer Complete interrupt disabled
1: Transfer Complete interrupt enabled
Note: Any of these events generate an interrupt:
Transfer Complete (TC)
Transfer Complete Reload (TCR)

Bit 5 **STOPIE**: STOP detection Interrupt enable
0: Stop detection (STOPF) interrupt disabled
1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (target only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disable
1: Peripheral enable
Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

24.6.2 Control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD 10R	ADD10	RD_W RN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw									

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE=0.

0: No PEC transfer.

1: PEC transmission/reception is requested

Note: Writing '0' to this bit has no effect.

This bit has no effect when RELOAD is set.

This bit has no effect is target mode when SBC=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 24.3: I2C implementation.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC=0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE=0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing '0' to this bit has no effect.

This bit is used in target mode only: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (controller mode)

The bit is set by software, cleared by hardware when a Stop condition is detected, or when PE = 0.

In Controller Mode:

0: No Stop generation.

1: Stop generation after current byte transfer.

Note: Writing '0' to this bit has no effect.

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCF bit in the I2C_ICR register.

0: No Start generation.

1: Restart/Start generation:

– If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.

– Otherwise setting this bit generates a START condition once the bus is free.

Note: Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10 bit target address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.

1: The controller only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode,

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer.

1: Controller requests a read transfer.

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:8 **SADD[9:8]**: Target address bit 9:8 (controller mode)

In 7-bit addressing mode (ADD10 = 0):
These bits are don't care

In 10-bit addressing mode (ADD10 = 1):
These bits should be written with bits 9:8 of the target address to be sent

Note: Changing these bits when the START bit is set is not allowed.

Bits 7:1 **SADD[7:1]**: Target address bit 7:1 (controller mode)

In 7-bit addressing mode (ADD10 = 0):
These bits should be written with the 7-bit target address to be sent

In 10-bit addressing mode (ADD10 = 1):
These bits should be written with bits 7:1 of the target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

Bit 0 **SADD0**: Target address bit 0 (controller mode)

In 7-bit addressing mode (ADD10 = 0):
This bit is don't care

In 10-bit addressing mode (ADD10 = 1):
This bit should be written with bit 0 of the target address to be sent

Note: Changing these bits when the START bit is set is not allowed.

24.6.3 Own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:8]		OA1[7:1]							OA1[0]
rw					rw	rw		rw							rw

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

Bit 15 **OA1EN**: Own Address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE** Own Address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN=0.

Bits 9:8 **OA1[9:8]**: Interface address

7-bit addressing mode: don’t care
10-bit addressing mode: bits 9:8 of address

Note: These bits can be written only when OA1EN=0.

Bits 7:1 **OA1[7:1]**: Interface address

Bits 7:1 of address

Note: These bits can be written only when OA1EN=0.

Bit 0 **OA1[0]**: Interface address

7-bit addressing mode: don’t care
10-bit addressing mode: bit 0 of address

Note: This bit can be written only when OA1EN=0.

24.6.4 Own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw			rw							

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

Bit 15 **OA2EN**: Own Address 2 enable

0: Own address 2 disabled. The received target address OA2 is NACKed.
1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own Address 2 masks

000: No mask
001: OA2[1] is masked and don’t care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don’t care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don’t care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don’t care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don’t care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don’t care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don’t care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN=0.

As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address

bits 7:1 of address

Note: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.

24.6.5 Timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

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

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to *I2C timings on page 576*) and for SCL high and low level counters (refer to *I2C controller initialization on page 590*).

t_{PRESC} = (PRESC+1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge in transmission mode.

t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge SDA edge in transmission mode.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH+1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL+1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C is disabled (PE = 0).

24.6.6 Timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

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

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

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

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

In controller mode, the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected

In target mode, the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$= (TIMEOUTB+1) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN=0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled: when SCL is low for more than $t_{TIMEOUT}$ (TIDLE=0) or high for more than t_{IDLE} (TIDLE=1), a timeout error is detected (TIMEOUT=1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN=0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A

This field is used to configure:

– The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE=0

$t_{TIMEOUT}$= (TIMEOUTA+1) x 2048 x t_{I2CCLK}

– The bus idle condition (both SCL and SDA high) when TIDLE=1

t_{IDLE}= (TIMEOUTA+1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Refer to Section 24.3: I2C implementation.

24.6.7 Interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r							r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

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

Bits 23:17 **ADDCODE[6:0]**: Address match code (Target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 **DIR**: Transfer direction (Target mode)

This flag is updated when an address match event occurs (ADDR=1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a Stop condition is detected, or when PE=0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 24.3: I2C implementation.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 24.3: I2C implementation.

Bit 11 **PECERR**: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 24.3: I2C implementation.

Bit 10 **OVR**: Overrun/Underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced Start or Stop condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting *BERRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 7 **TCR**: Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE=0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer Complete (controller mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE=0.

Bit 5 **STOPF**: Stop detection flag

This flag is set by hardware when a Stop condition is detected on the bus and the peripheral is involved in this transfer:

– either as a controller, provided that the STOP condition is generated by the peripheral.

– or as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 4 **NACKF**: Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE=0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to ‘1’ by software when NOSTRETCH=1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN=1).

Note: This bit is cleared by hardware when PE=0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to ‘1’ by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE=0.

24.6.8 Interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERT CF	TIM OUTCF	PECCF	OVRCF	ARLO CF	BERR CF	Res.	Res.	STOP CF	NACK CF	ADDR CF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 24.3: I2C implementation.

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 24.3: I2C implementation.

Bit 11 **PECCF**: PEC Error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 24.3: I2C implementation.

Bit 10 **OVRCF**: Overrun/Underrun flag clear

Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration Lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

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

Bit 5 **STOPCF**: Stop detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear
Writing 1 to this bit clears the ACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

24.6.9 PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

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

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

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

Bits 7:0 **PEC[7:0]** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 24.3: I2C implementation.

24.6.10 Receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

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

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

Bits 7:0 **RXDATA[7:0]** 8-bit receive data
Data byte received from the I^{2}C bus.

24.6.11 Transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

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

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

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data
Data byte to be transmitted to the I^{2}C bus.
Note: These bits can be written only when TXE=1.

24.7 I2C register map

The table below provides the I2C register map and reset values.

Table 81. I2C register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	**I2C_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	Res.	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x4	**I2C_CR2**	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	**I2C_OAR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0xC	**I2C_OAR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	**I2C_TIMINGR**	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**I2C_ TIMEOUTR**	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**I2C_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	**I2C_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	**I2C_PECR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	**I2C_RXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 81. I2C register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	**I2C_TXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

25 Universal synchronous asynchronous receiver transmitter (USART)

25.1 USART introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a fractional baud rate generator.

It supports synchronous one-way communication and half-duplex single wire communication. It also supports the LIN (local interconnection network), Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). It also supports multiprocessor communications.

High speed data communication is possible by using the DMA (direct memory access) for multibuffer configuration.

25.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to give flexibility between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data, that can be enabled/disabled by software. FIFOs come with status flags for FIFOs states.
- A common programmable transmit and receive baud rate of up to 2Mbit/s with the clock frequency at 16 MHz and oversampling is by 8
- Dual clock domain with a dedicated kernel clock allowing baud rate programming independent from the PCLK reprogramming.
- Auto baud rate detection
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications
- Wakeup from mute mode (by idle line detection or address mark detection)

25.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T=0 and T=1 asynchronous protocols for Smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for Smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

25.4 USART implementation

Table 82 describes the USART and LPUART implementation on the STM32WL30xx/31xx/33xx.

Table 82. USART / LPUART features

USART modes/features(1)	USART	LPUART
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode (Master/Slave)	X	-
Smartcard mode	X	-
Single-wire half-duplex communication	X	X
IrDA SIR ENDEC block	X	-
LIN mode	X	-
Dual clock domain	X	X
Receiver timeout interrupt	X	-
Modbus communication	X	-
Auto baud rate detection	X	-
Driver Enable	X	X
USART data length	7, 8 and 9 bits	
Tx/Rx FIFO	X	X
Tx/Rx FIFO size	8	

1. X = supported.

25.5 USART functional description

Any USART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX**: Receive Data Input.
  This is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.
- **TX:** Transmit Data Output.
  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire and Smartcard modes, this I/O is used to transmit and receive the data.

Serial data are transmitted and received through these pins in normal USART mode. The frames are comprised of:

- an Idle Line prior to transmission or reception
- a start bit
- a data word (7, 8 or 9 bits) least significant bit first
- 0.5, 1, 1.5, 2 Stop bits indicating that the frame is complete
- The USART interface uses a baud rate generator
- a status register (USART_ISR)
- Receive and transmit data registers (USART_RDR, USART_TDR). When FIFO mode is enabled, writing into USART_TDR adds one data to the transmit FIFO; and reading from USART_RDR removes one data from the receive FIFO.
- a baud rate register (USART_BRR)
- a guardtime register (USART_GTPR) in case of Smartcard mode

Refer to *Section 25.7: USART registers* for the definitions of each bit.

The following pin is required to interface in synchronous mode and Smartcard mode:

- **SCLK:** This pin acts as Clock output in synchronous master and Smartcard modes. It acts as Clock input is Synchronous slave mode. In Synchronous Master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX. This can be used to control peripherals that have shift registers (for example LCD drivers). The clock phase and polarity are software programmable.

In Smartcard mode, SCLK output can provide the clock to the Smartcard.

- **NSS**: This pin acts as Slave Select input in Synronous slave mde.

The following pins are required in RS232 Hardware flow control mode:

- **nCTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS:** Request to send indicates that the USART is ready to receive data (when low).

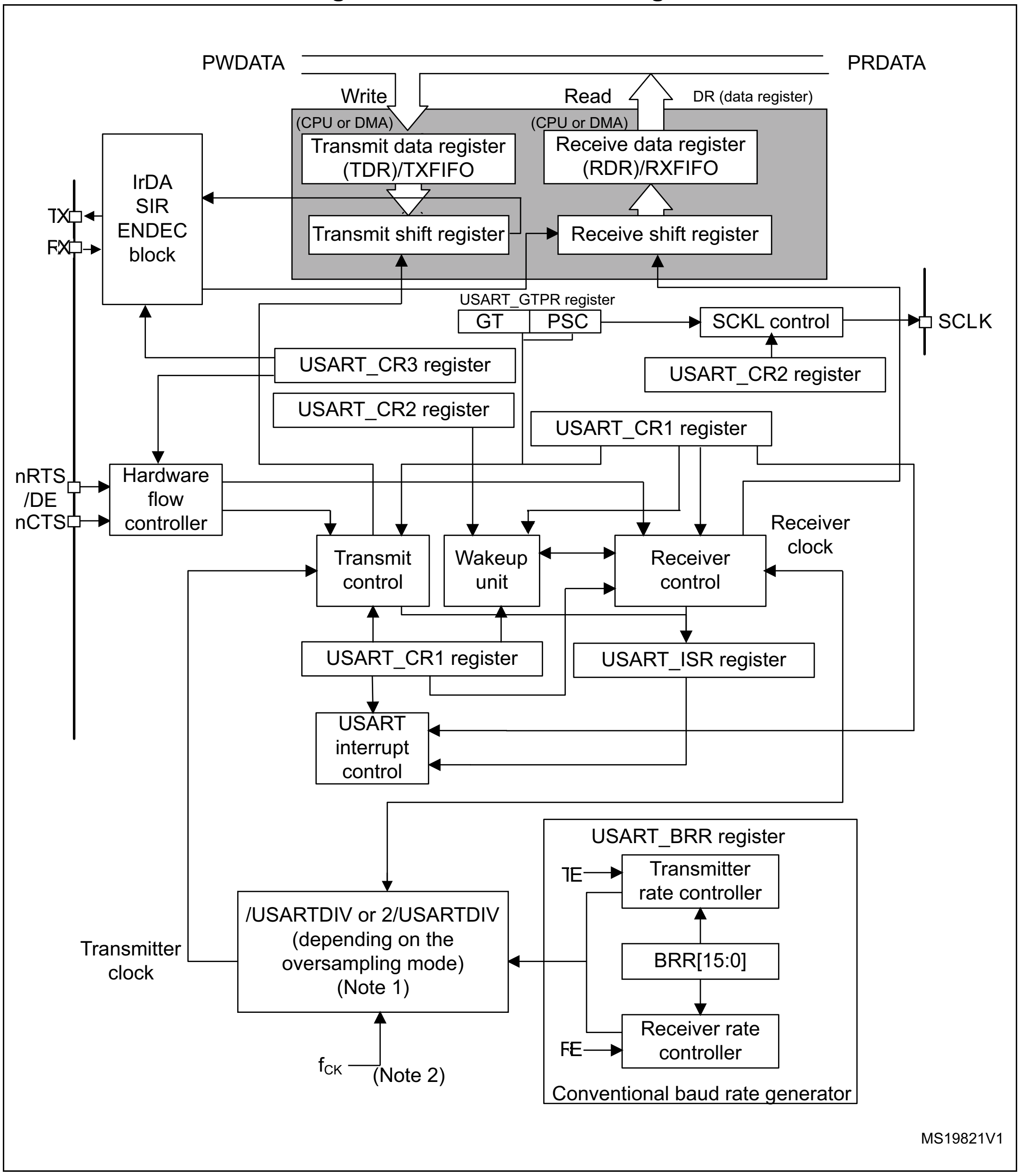
The following pin is required in RS485 Hardware control mode:

- **DE**: Driver Enable activates the transmission mode of the external transceiver.

Note: DE and nRTS share the same pin.

Note: NSS and nCTS share the same pin.

Figure 194. USART block diagram



1. For details on coding USARTDIV in the USARTx_BRR register, refer to *Section 25.5.5: Baud rate generation*.
2. f_{CK} is 16 MHz

25.5.1 USART character description

The word length can be selected as being either 7 or 8 or 9 bits by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 194*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and Autobaudrate (0x7F and 0x55 frames detection) are not supported.

In default configuration, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

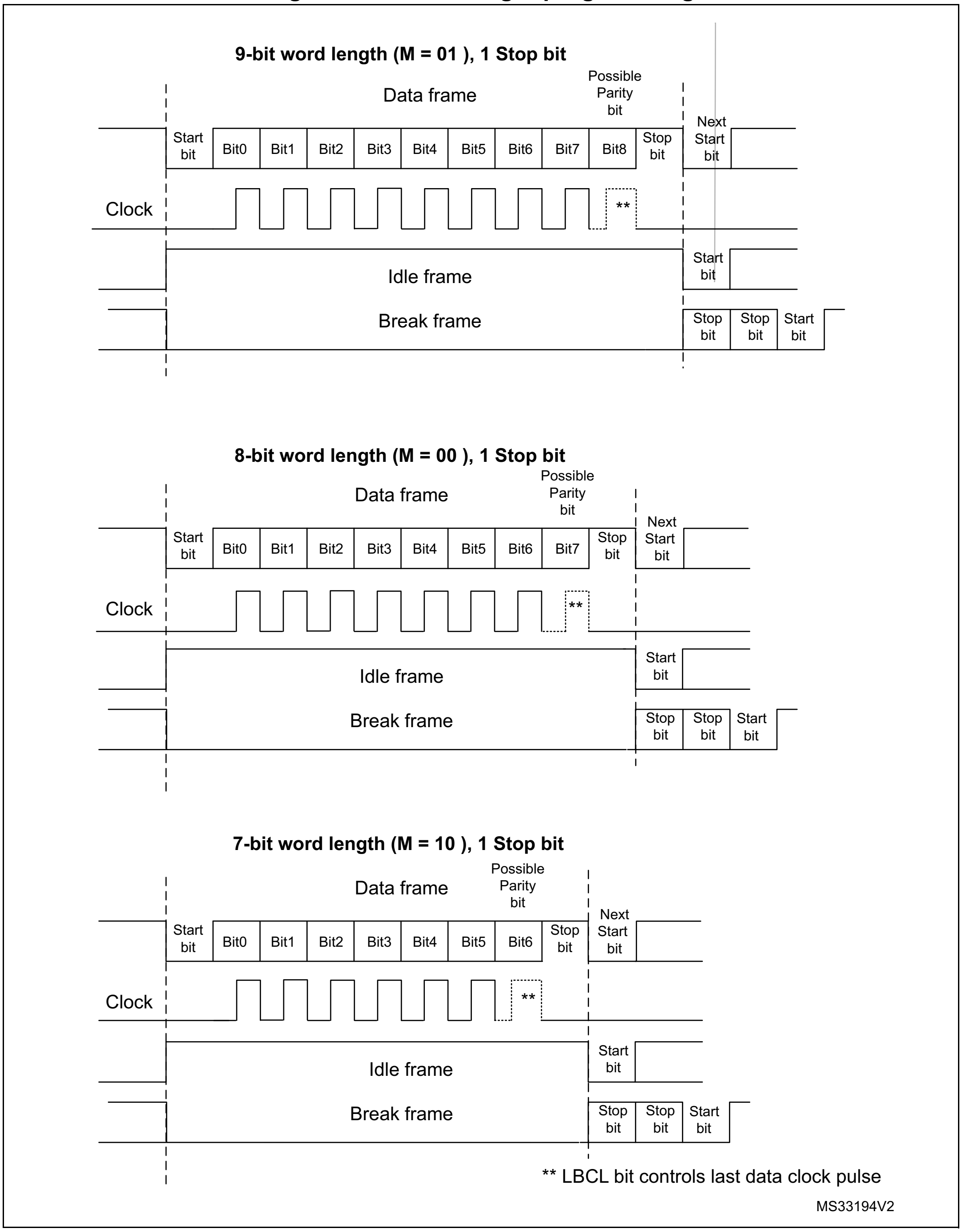
An ***Idle character*** is interpreted as an entire frame of “1”s. (The number of “1” ‘s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

Figure 195. Word length programming



25.5.2 FIFOs and thresholds

The USART can operate in FIFO mode, with the FIFO buffers having a depth of 16 bytes.

The USART come with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO) . The FIFO mode is enabled by setting the bit 29 FIFOEN in USARTx_CR1 register.

The FIFO mode is supported only on UART, SPI and Smartcard modes.

Being 9 bits the maximum data word length, the TXFIFO is 9-bits wide. However the RXFIFO is by default 12-bits wide. This is due to the fact that the receiver does not only put the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO with its flags. However reading RDR provides only the data. The status flags are available in the USART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupt are triggered. These thresholds are programmed through bit fields RXFTCFG and TXFTCFG in USARTx_CR3 control register.

In this case:

- The receive interrupt in generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.
- The transmit interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

RXFIFO threshold

The RXFIFO threshold is configured using the RXFTCFG bits fields in the USARTx_CR3 register.

When the number of received data is equal to the programmed RXFTCFG, the flag RXFT in the USART_ISR register is set.

Having RXFT flag set means that there are RXFTCFG data received: 1 data in USARTx_RDR and (RXFTCFG - 1) data in the RXFIFO. So, when the RXFTCFG is programmed to «101» , the RXFT flag is set when 8 data are received: 7 data in the RXFIFO and 1 data in the USARTx_RDR. Consequently, the 9th received data does not set the overrun flag.

25.5.3 Transmitter

The transmitter can send data words of either 7 or 8 or 9 bits depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin and the corresponding clock pulses are output on the SCLK pin.

Character transmission

During an USART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the USARTx_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, data written to the transmit data register USART_TDR, is queued in the TXFIFO.

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by USART: 0.5, 1, 1.5 and 2 stop bits.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters become frozen. The current data being transmitted is lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

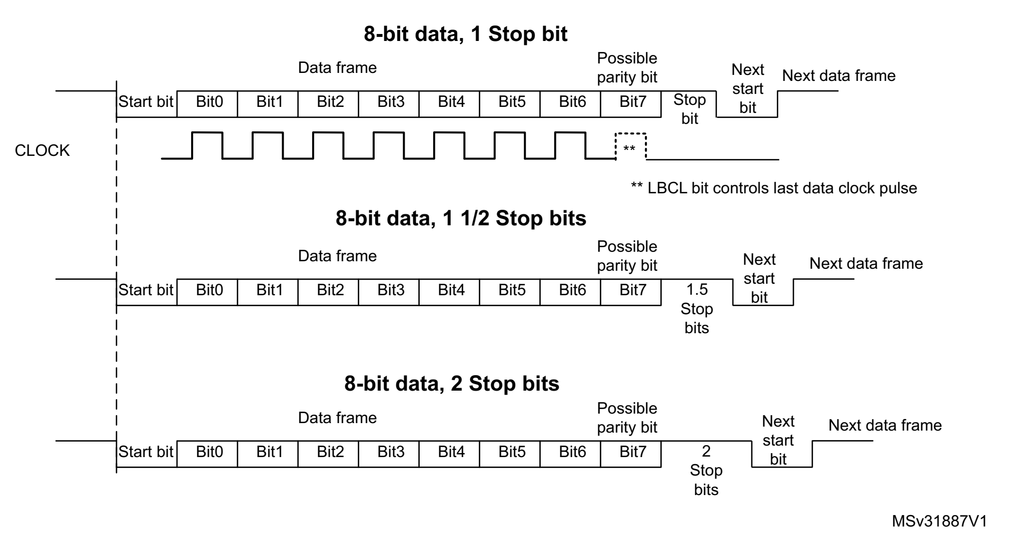
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 Stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in Smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Figure 196*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 196. Configurable stop bits



Character transmission procedure

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   a) When FIFO mode is disabled, writing a data in the USART_TDR clears the TXE flag.
   b) When FIFO mode is enabled, writing a data in the USART_TDR adds one data to the TXFIFO and write operations in the USART_TDR are made when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. After writing the last data into the USART_TDR register, wait until TC=1.
   a) When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   b) When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

Single byte communication

- When FIFO mode is disabled:

clearing the TXE flag is always performed by a write to the transmit data register.

The TXE flag is set by hardware and it indicates:

  – The data has been moved from the USARTx_TDR register to the shift register and the data transmission has started.
  – The USARTx_TDR register is empty.
  – The next data can be written in the USARTx_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is taking place, a write instruction to the USARTx_TDR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the USARTx_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware and it indicates:
  – The TXFIFO is not full
  – The USART_TDR register is empty
  – The next data can be written in the USART_TDR register without overwriting the previous data. When a transmission is taking place, a write operation to the

USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO into the shift register at the end of the current transmission.

When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in USART_TDR . It is cleared when the TXFIFO is full.

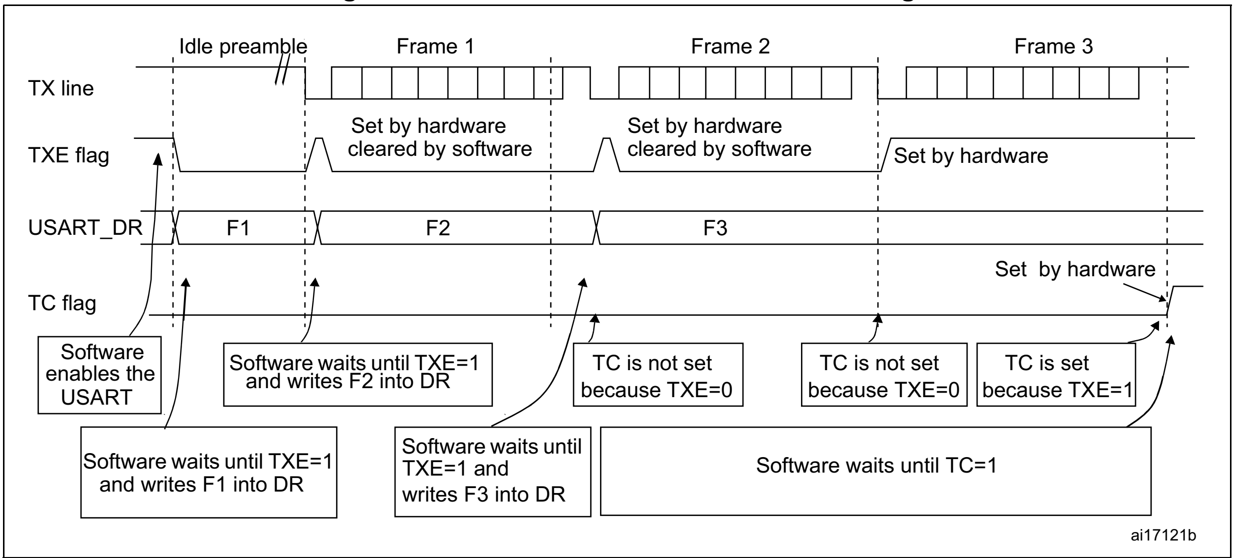
This flag generates an interrupt if the TXFNFIE bit is set.

Alternatively, interrupts can be generated and data can be written into FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed triger level.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data in the USART_TDR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low power mode (see *Figure 197: TC/TXE behavior when transmitting*).

Figure 197. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 195*).

If a ‘1’ is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

25.5.4 Receiver

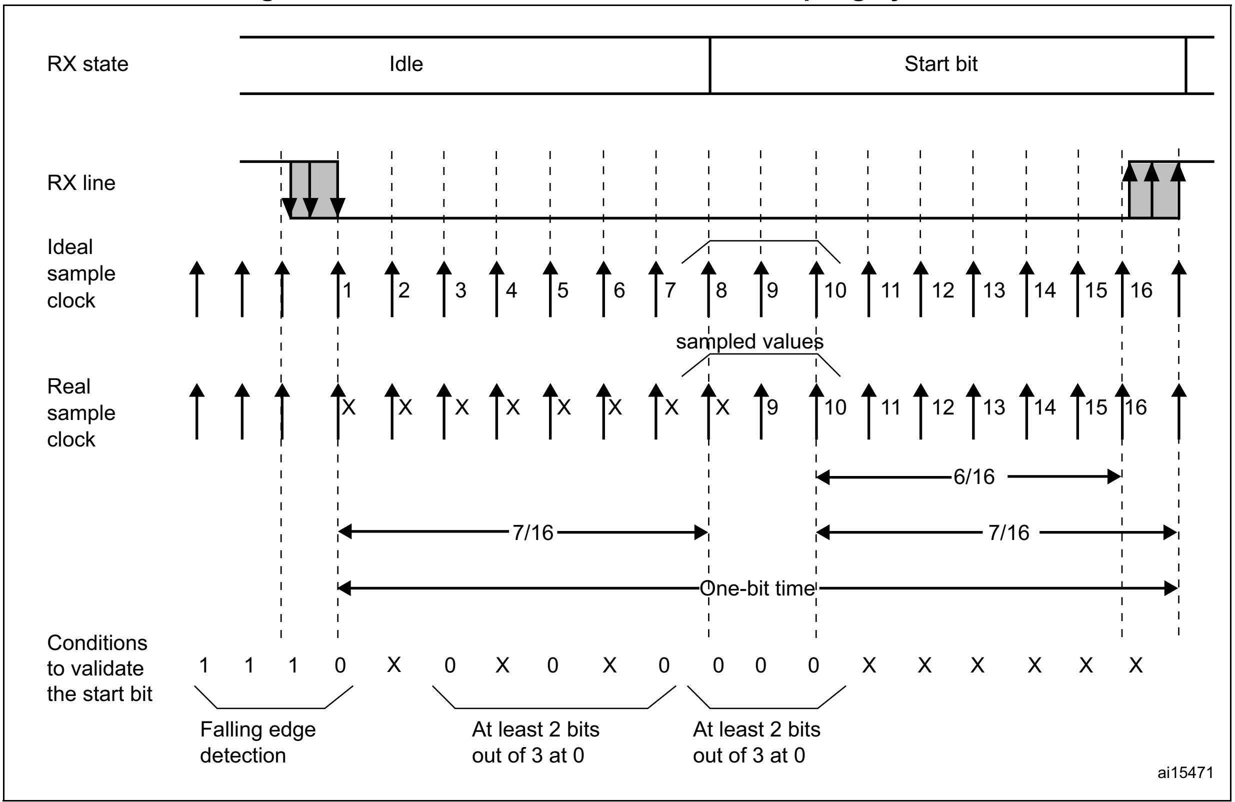
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 198. Start bit detection when oversampling by 16 or 8



Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set, interrupt generated if RXNEIE=1 (RXFNE flag set, interrupt generated if RXFNEIE=1 if FIFO mode is enabled) i)f the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NF noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither conditions a. or b. are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data shifts in least significant bit first (default configuration) through the RX pin.

Character reception procedure

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set indicating that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. A read of the USART_RDR gets the oldest entry in the RXFIFO. When a data in received, it is stored in the RXFIFO, with error bits associated with that data.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error has been detected during reception.
- In multibuffer communication:
  - When FIFO mode is disabled, the RXNE is set after every byte received and is cleared by the DMA read of the Receive data Register.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is data in the RXFIFO to be read.
- In single buffer mode,
  - When FIFO mode is disabled: clearing the RXNE flag is performed by a software read to the USARTx_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USARTx_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read of the USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register. When the

RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as for a received data character plus an interrupt if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled:
  An overrun error occurs when a character is received when RXNE has not been reset. data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  - The ORE bit is set.
  - The RDR content is not lost. The previous data is available when a read to
  - USARTx_RDR is performed.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXNEIE bit is set or EIE bit is set.
- FIFO mode enabled:
  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full. Data can not be transferred from the shift register to the USARTx_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty. An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  - The ORE bit is set.
  - The first entry in the RXFIFO is not lost. It is available when a read to USART_RDR is performed.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE bit is set or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USARTx_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

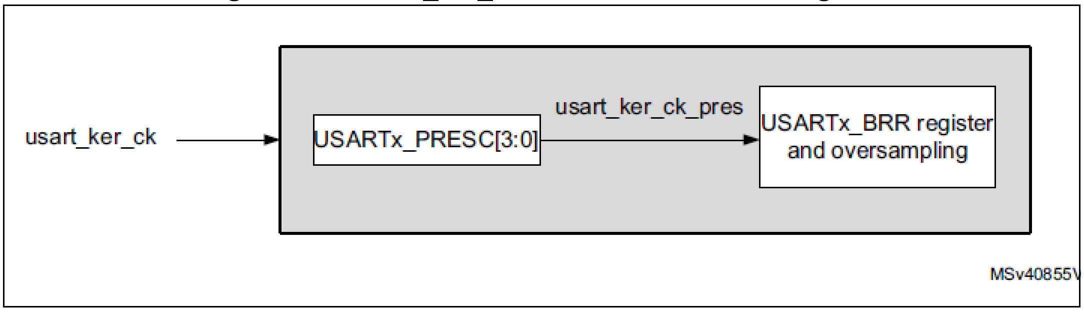
- *if RXNE=1, then the last valid data is stored in the receive register RDR and can be read,*
- *if RXNE=0, then it means that the last valid data has already been read and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source and the proper oversampling method

The clock source frequency is f_{CK}(16 MHz).

The f_{CK} can be divided by a programmable factor in the USARTx_PRESC register.

Figure 199. usart_ker_ck clock divider block diagram



The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This allows a trade of between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register and can be either 16 or 8 times the baud rate clock (*Figure 200* and *Figure 201*)

Depending on the application:

- Select oversampling by 8 (OVER8=1) to achieve higher speed (up to f_{CKPRES}/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 25.5.6: Tolerance of the USART receiver to clock deviation*)
- Select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum f_{CKPRES}/16.

where f_{CKPRES} is the USART input clock divided by a prescaler.

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. There are two options:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NF bit is set
- A single sample in the center of the received bit

  Depending on the application:
  - select the three samples' majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to Table 83) because this indicates that a glitch occurred during the sampling.
  - select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver's tolerance to clock deviations (see Section 25.5.6: Tolerance of the USART receiver to clock deviation). In this case the NF bit is never set.

When noise is detected in a frame:

- The NF bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NF bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI mode.

Note: Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to '0 ' by hardware.

Figure 200. Data sampling when oversampling by 16

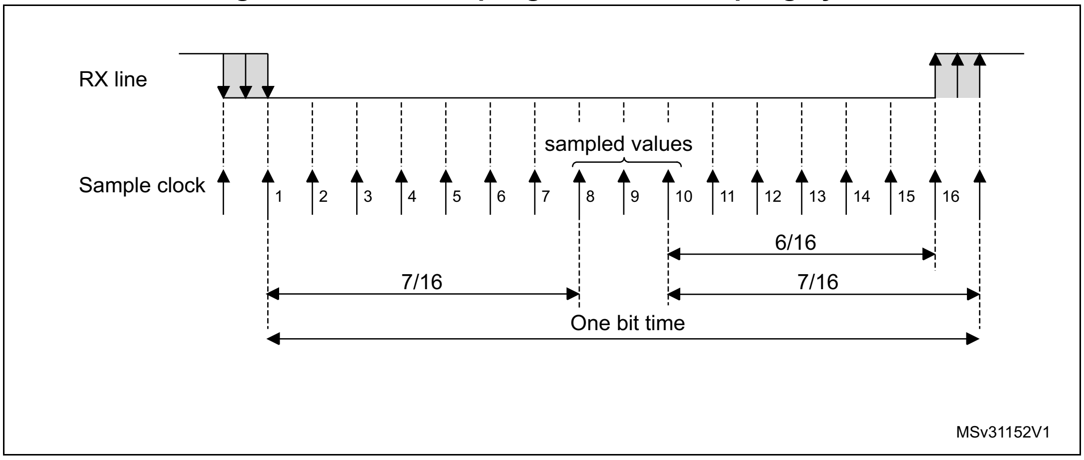


Figure 201. Data sampling when oversampling by 8

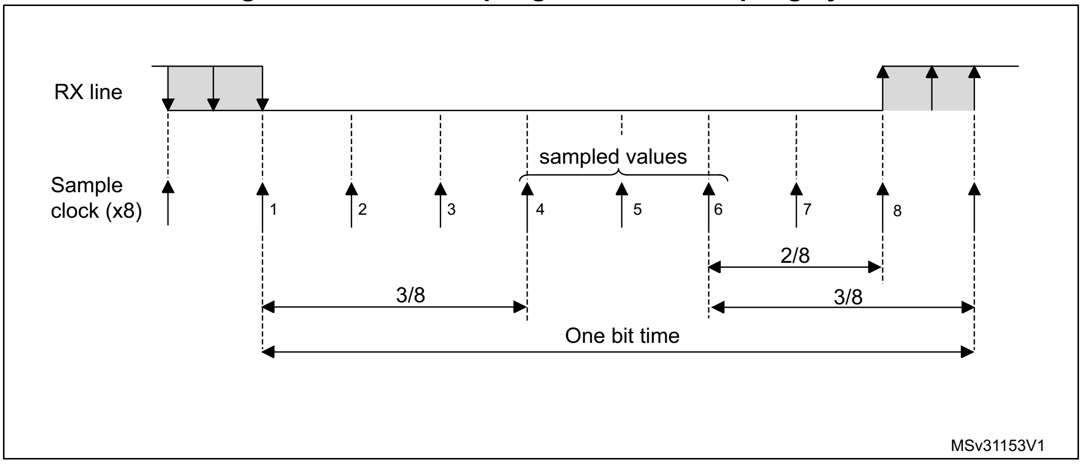


Table 83. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware
- The invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

- **0.5 stop bit (reception in Smartcard mode)**: No sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit iis selected.
- ***1 stop bit***: Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
- ***1.5 stop bits (Smartcard mode)***: When transmitting in Smartcard mode, the device must check that the data is correctly sent. Thus the receiver block must be enabled (RE =1 in the USART_CR1 register) and the stop bit is checked to test if the Smartcard has detected a parity error. In the event of a parity error, the Smartcard forces the data signal low during the sampling - NACK signal-, which is flagged as a framing error. Then, the FE flag is set with the RXNE (RXFNE in case FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be decomposed into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through. Refer to *Section 25.5.14: Receiver Timeout* for more details.
- ***2 stop bits***: Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. If a framing error is detected during the first stop bit the framing error flag is set. The second stop bit is not checked for framing error. The RXNE (RXFNE in case FIFO mode is enabled) flag is set at the end of the first stop bit.

25.5.5 Baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the USART_BRR register.

Equation 1: Baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the equation is:

$$\text{Tx/Rx baud} = \frac{f_{CKPRES}}{\text{USARTDIV}}$$

In case of oversampling by 8, the equation is:

$$\text{Tx/Rx baud} = \frac{2 \times f_{CKPRES}}{\text{USARTDIV}}$$

Equation 2: Baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)

$$\text{Tx/Rx baud} = \frac{f_{CKPRES}}{\text{USARTDIV}}$$

fckpres is the USART clock which is the USART input clock divided by a prescaler onfigured in the USART_PRES regsiter.

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16d.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 baud with f_{CKPRES} = 8 MHz.

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR = USARTDIV = 833d = 0341h
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (1667d = 683h)
  BRR[3:0] = 3h >>1 = 1h
  BRR = 0x681

25.5.6 Tolerance of the USART receiver to clock deviation

The asynchronous receiver of the USART works correctly only if the total clock system deviation is less than the tolerance of the USART receiver. The causes which contribute to the total deviation are:

- DTRA: Deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- DQUANT: Error due to the baud rate quantization of the receiver
- DREC: Deviation of the receiver’s local oscillator
- DTCL: Deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$\mathrm{DTRA} + \mathrm{DQUANT} + \mathrm{DREC} + \mathrm{DTCL} < \mathrm{USART\ receiver's\ tolerance}$$

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 84*, *Table 85*, depending on the following choices:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 84. Tolerance of the USART receiver when BRR [3:0] = 0000 (high-density devices)

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16	4.86	2.77	4.16

Table 85. Tolerance of the USART receiver when BRR[3:0] is different from 0000 (high-density devices)

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7	4.31	2.22	3.33

Note: The data specified in Table 84, Table 85 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M=01 or 9- bit times when M = 10).

25.5.7 Auto baud rate detection

The USART is able to detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance
- The system is using a relatively low accuracy clock source and this mechanism allows the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed (When oversampling by 16, the baud rate is between f_{CK}/65535 and f_{CK}/16. When oversampling by 8, the baudrate is between f_{CK}/65535 and f_{CK}/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be chosen. There are four modes based on different character patterns.

The modes can be chosen through the ABRMOD[1:0] field in the USART_CR2 register. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are:

- **Mode 0**: Any character starting with a bit at 1.
  In this case the USART measures the duration of the Start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.
  In this case, the USART measures the duration of the Start and of the 1st data bit. The

measurement is done falling edge to falling edge, ensuring better accuracy in the case of slow signal slopes.

- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).
  In this case, the baudrate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to Bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.
  In this case, the baudrate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit0 is sampled at BRs, Bit1 to Bit6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate transition of RX line. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled, in case of auto baud rate error, the ABRE flag is set with RXNE and FE.

When FIFO management is enabled, in case of auto baud rate error, the ABRE flag is set with RXFNE and FE.

In case of FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launch the auto baud rate detection, the user should make sure that the RXFIFO is empty using the RXFNE flag in USARTx_ISR register.

Note: The BRR value may be corrupted if the USART is disabled (UE=0) during an auto baud rate operation.

25.5.8 Multiprocessor communication

It is possible to perform multiprocessor communication with the USART (with several USARTs connected in a network). For instance one of the USARTs can be the master, its TX output connected to the RX inputs of the other USARTs. The others are slaves, their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In order to use the mute mode feature, the MME bit must be set in the USARTx_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two UCLK cycles) otherwise mute mode might remain active.

In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

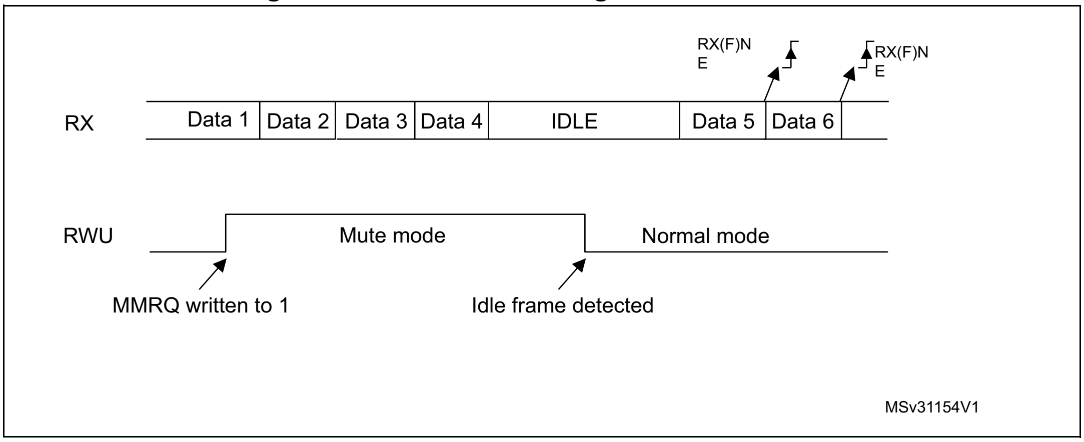
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 202*.

Figure 202. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a '1' otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-

bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering mute mode.

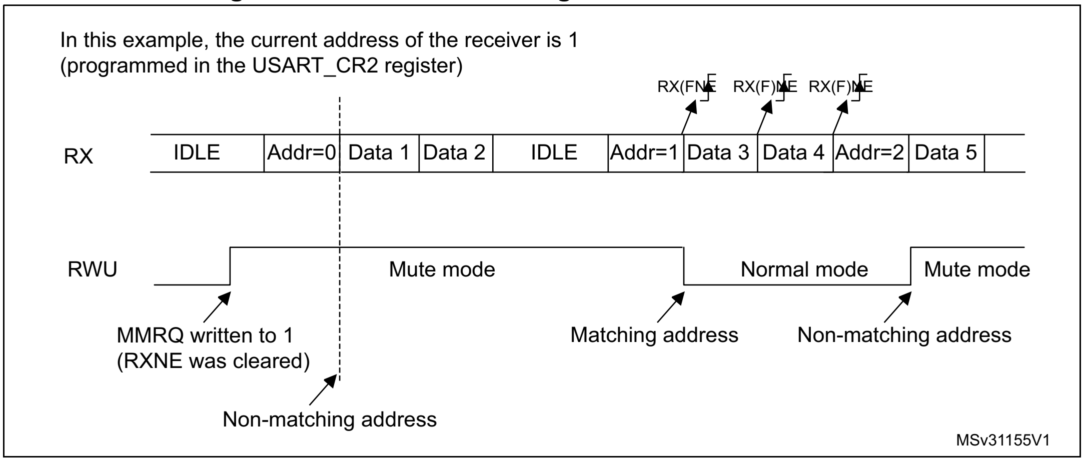
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 203*.

Figure 203. Mute mode using address mark detection



25.5.9 Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half duplex, block transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE=1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

25.5.10 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 86*.

Table 86. Frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

25.5.11 LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 25.4: USART implementation on page 638*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USART_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure explained in *Section 25.5.2* has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0 bits as a break character. Then 2 bits of value ‘1 are sent to allow the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as ‘0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the *Figure 204: Break detection in LIN mode (11-bit break length - LBDL bit is set)*.

Examples of break frames are given on *Figure 205: Break detection in LIN mode vs. Framing error detection*.

Figure 204. Break detection in LIN mode (11-bit break length - LBDL bit is set)

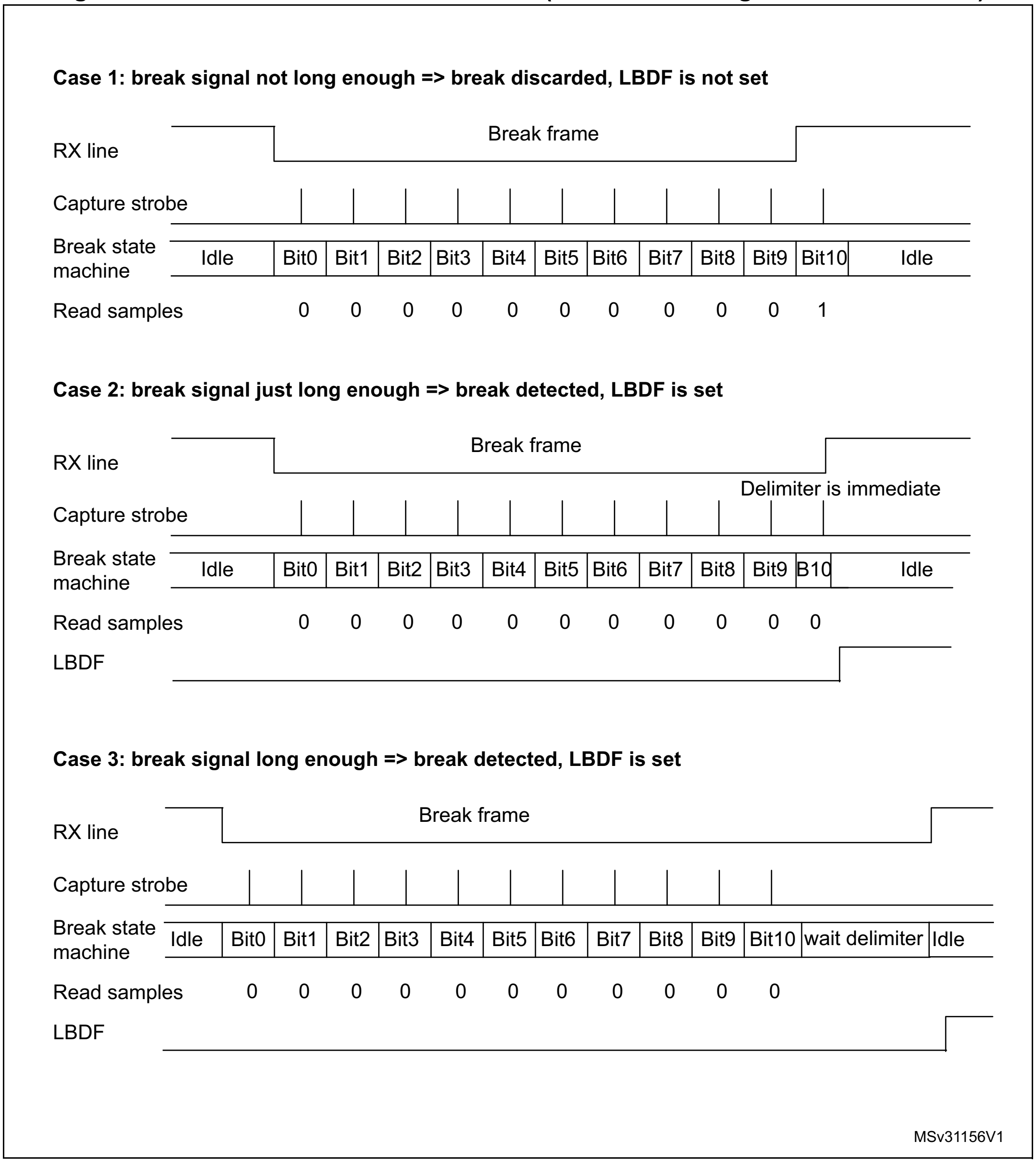
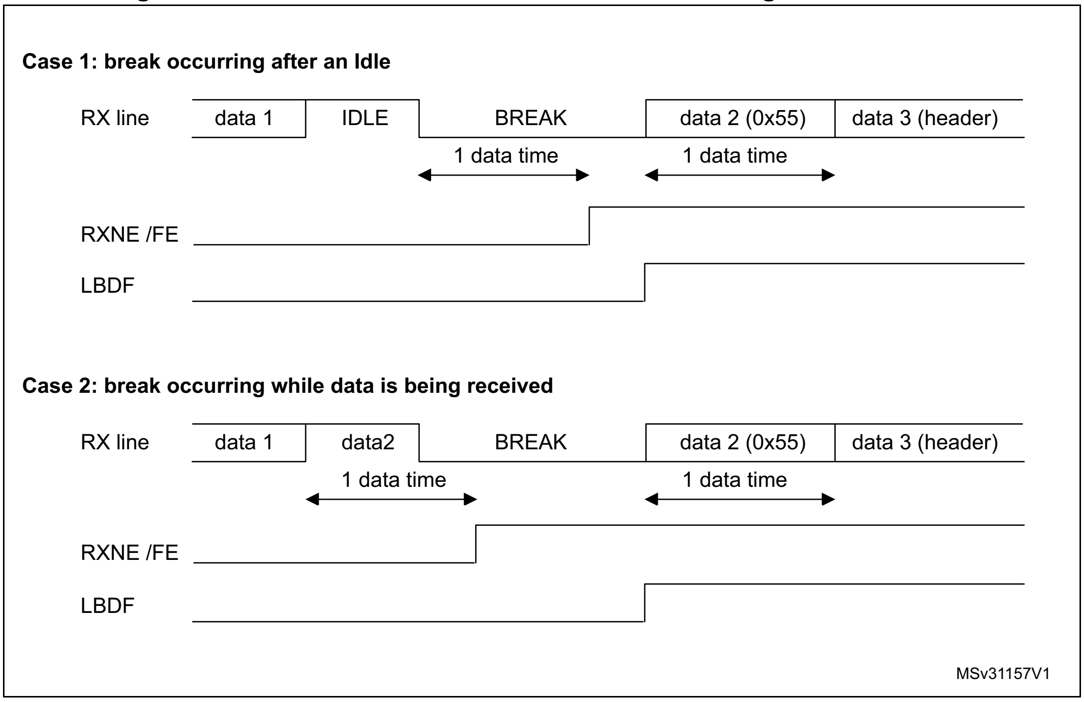


Figure 205. Break detection in LIN mode vs. Framing error detection



25.5.12 USART synchronous mode

Master Mode

The synchronous master mode is selected by writing the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 206*, *Figure 207* and *Figure 208*).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous master mode the USART transmitter works exactly like in asynchronous mode. But as SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In this mode the USART receiver works in a different manner compared to the asynchronous mode. If RE=1, the data is sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: *In master mode, the SCLK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data is being transmitted (the data* *register USART_DR written). This means that it is not possible to receive synchronous data without transmitting data.*

Figure 206. USART example of synchronous Master transmission

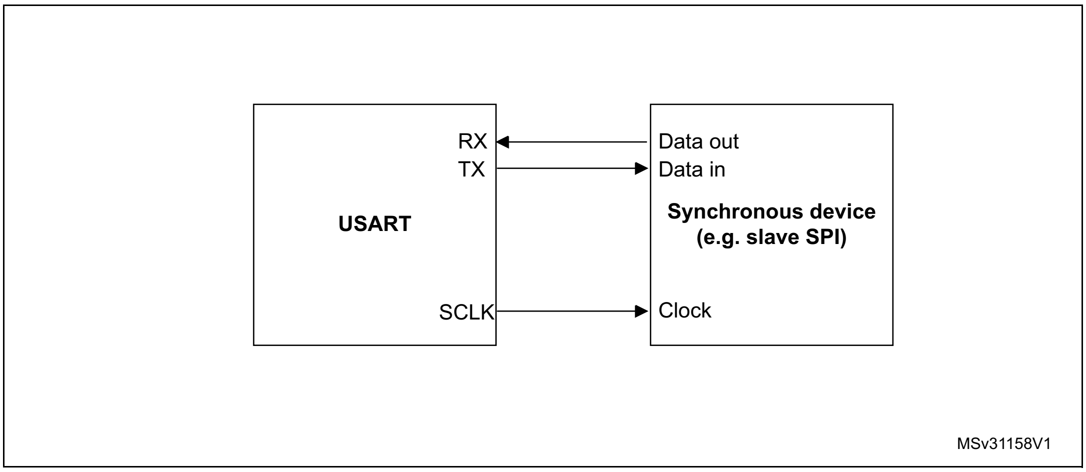


Figure 207. USART data clock timing diagram (M=0)

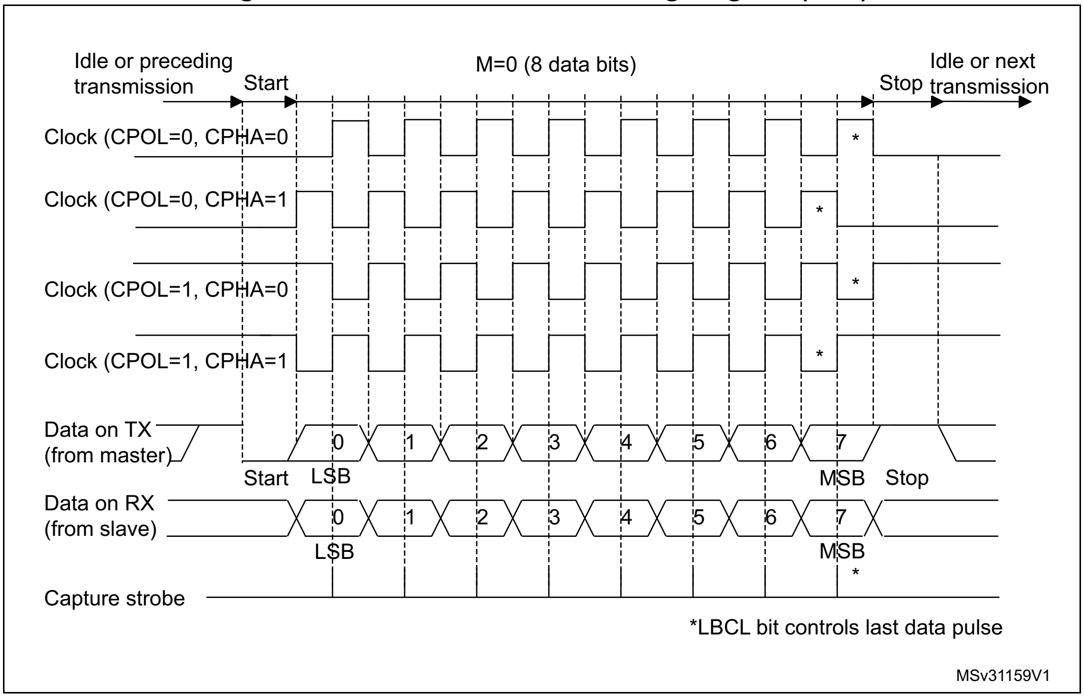


Figure 208. USART data clock timing diagram (M bits = 01)

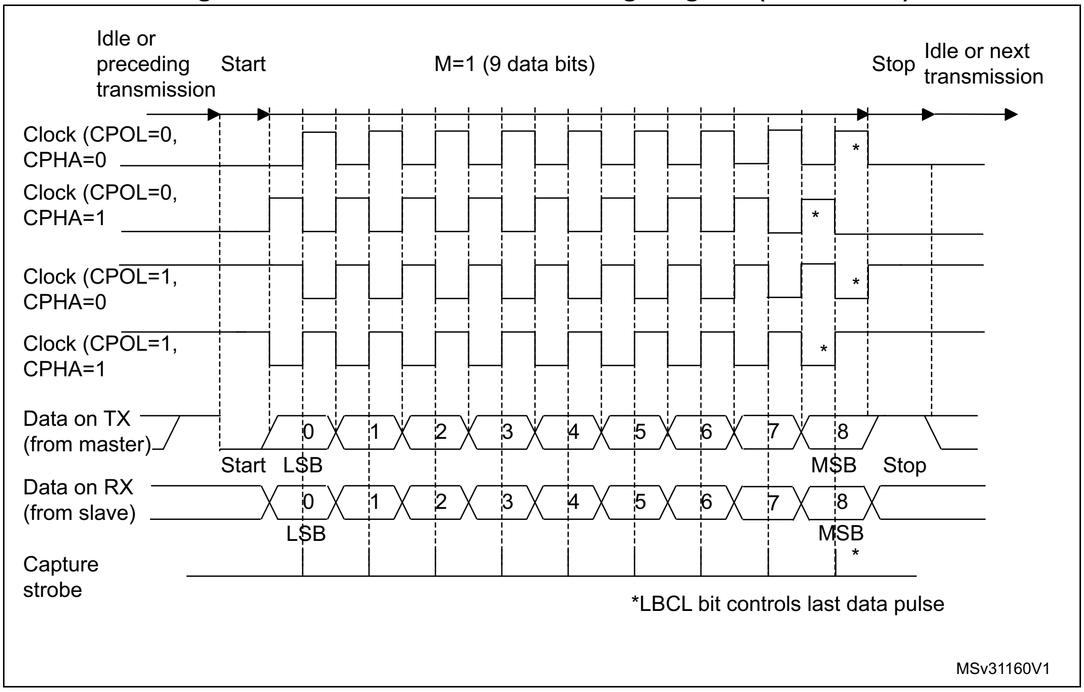
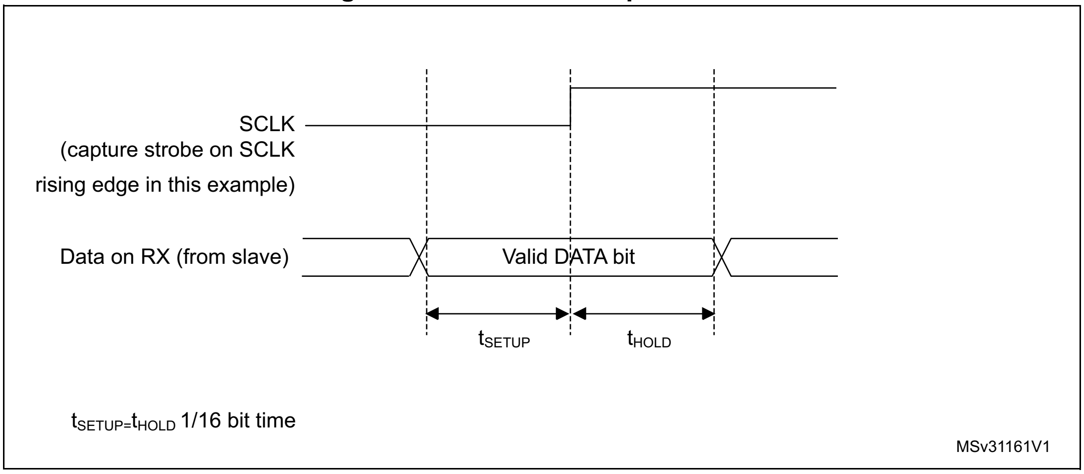


Figure 209. RX data setup/hold time



Slave mode

The synchronous slave mode is selected by writing the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The SCLK pin is the input of the USART in slave mode.

Note: When the peripheral is used in SPI slave mode, the peripheral clock source (fck_pres) must be greater than 3xSCLK input clock.

The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 206*, *Figure 207* and *Figure 208*).

In slave transmission mode, an underun error flag is available. This flag is set when the first clock for data transmission appears while the software has not yet loaded any value into USARTx_TDR.

The slave supports the hardware and software NSS management.

Slave Select (NSS) pin management:

Hardware or software slave select management can be set using the DIS_NSS bit in the

USART_CR2 register.

- Software NSS management (DIS_NSS = 1)

SPI slave is always selected and NSS input pin is ignored .

The external NSS pin remains free for other application uses.

- Hardware NSS management (DIS_NSS = 0)

The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly

Note: When in SPI slave mode, the USART must be enabled before the master starts communication (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI Slave underrun error

When an underrun error occurs, the SPI slave sends the last data until the underrrun error flag is cleared in software.

The underrun flag is set at the beginning of the frame.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In underrun condition, it is allowed to write the TDR register. Clearing the underrun error would allow sending the new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the data is written in the USARTx_TDR too close to the first SCLK transmision edge. To avoid this underrun error, the USART_TDR should be written 3 UCLK cycles before the first first SCLK edge.

25.5.13 Single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflicts on the line must be managed by software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: As the TX line and the RX lines are connected together, all the transmitted data are stored in the RX FIFO as the data received from an external device. The software has to take care to discard its "own" information after a transmit phase.

In half-duplex mode, it is always wise to read back the transmitted data to check if they are correct as there is no hardware protection against possible collision between nodes.

If the software does not want to have the RX FIFO storing the transmitted value then it has to disable the receiver part while transmitting (by clearing the RE bit in USART_CR1 register).

25.5.14 Receiver Timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bit fields in the USARTx_RTOR register.

The receiver timeout counter starts counting

- From the end of the stop bit in case STOP = 00 and STOP = 11
- From the end of the second stop bit in case STOP = 10.
- From the beginning of the stop bit in case STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USARTx_ISR register is set. and a timeout is generated if RTOIE bit in USARTx_CR1 register is set.

25.5.15 Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to *Section 25.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit may be set in order to provide a clock to the Smartcard.

The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO 7816-3 standard. Both T=0 (character mode) and T=1 (block mode) are supported.

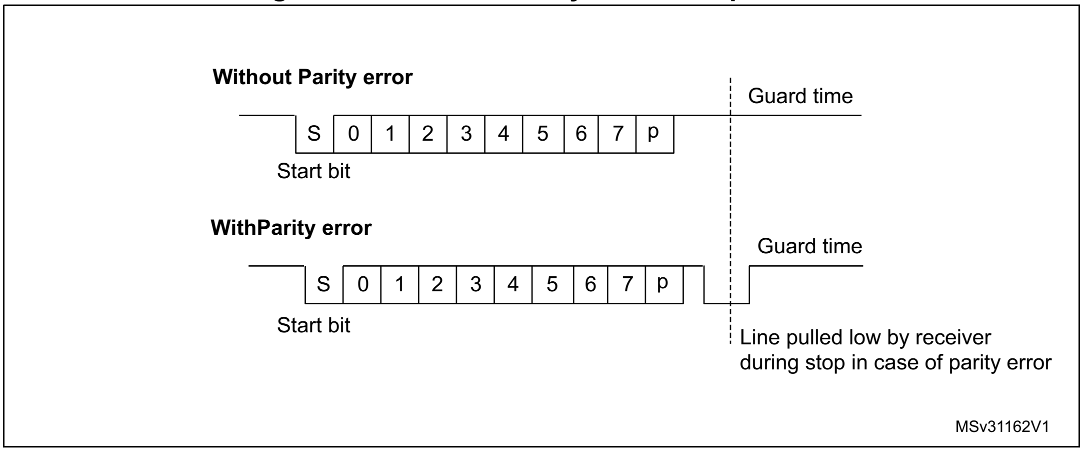
The USART should be configured as:

- 8 bits plus parity: where M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting abd receiving data: where STOP=11 in the USART_CR2 register. It is also possible to choose the 0.5 stop bit for receiving.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 210 shows examples of what can be seen on the data line with and without parity error.

Figure 210. ISO 7816-3 asynchronous protocol



When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol. The number of retries is programmed in the SCARCNT bit field. If the USART continues receiving the NACK after the programmed number of retries, it stops

transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

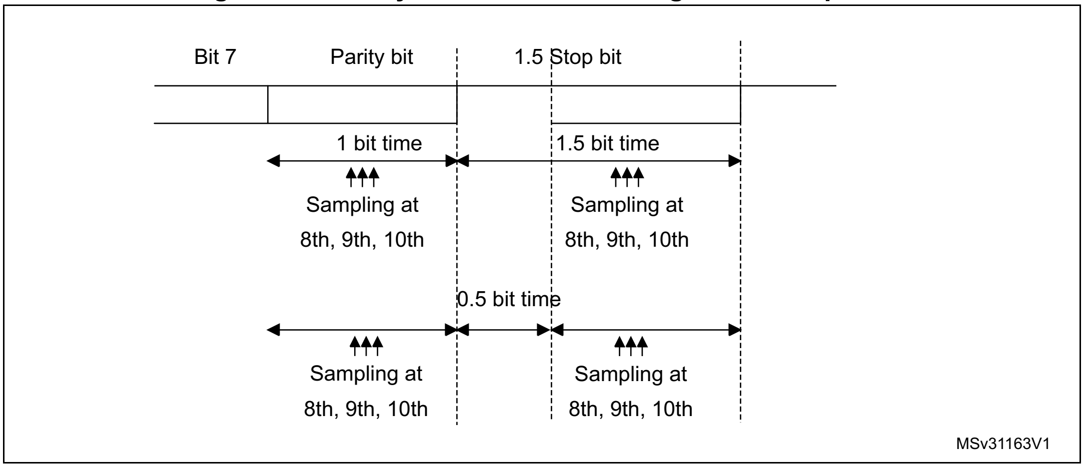
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bit field, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn’t repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: A break character is not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 211 details how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 211. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the Smartcard through the SCLK output. In Smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USART_. SCLK frequency can be programmed from $f_{CKPRES}/2$ to $f_{CKPRES}/62$, where f_{CKPRES} is the peripheral input clock divided by a programmed prescaler.

Block mode (T=1)

In T=1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the UART_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time) - 11 value, in order to allow the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baudtime units. If the Smartcard doesn't send a new character in less than the CWT period after the end of the previous character, the USART signals this to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the Smartcard protocol definition, the BWT/CWT values should be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value (0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

Note: When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH => the USART received character is ‘03’ and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card didn't answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103 -> inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B -> direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

25.5.16 IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 25.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 212*).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 212*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods are accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOP bits in the USART_CR2 register must be configured to "1 stop bit".

IrDA low-power mode

Transmitter

In low-power mode the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

Receiver:

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART_GTPR).

Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).

Figure 212. IrDA SIR ENDEC- block diagram

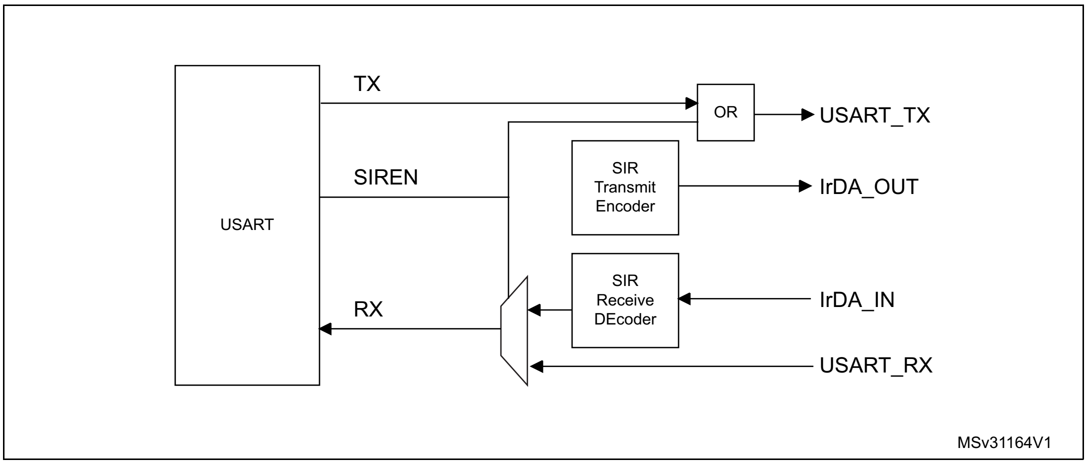
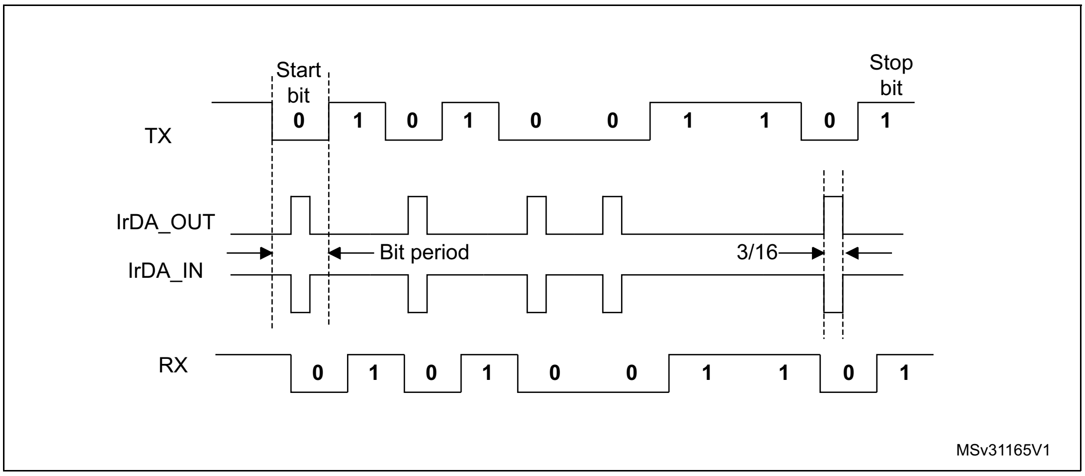


Figure 213. IrDA data modulation (3/16) -Normal Mode



25.5.17 Continuous communication using DMA

The USART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 25.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in or Section 25.5.4: Receiver. To perform continuous communication, When FIFO is disabled, you can clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

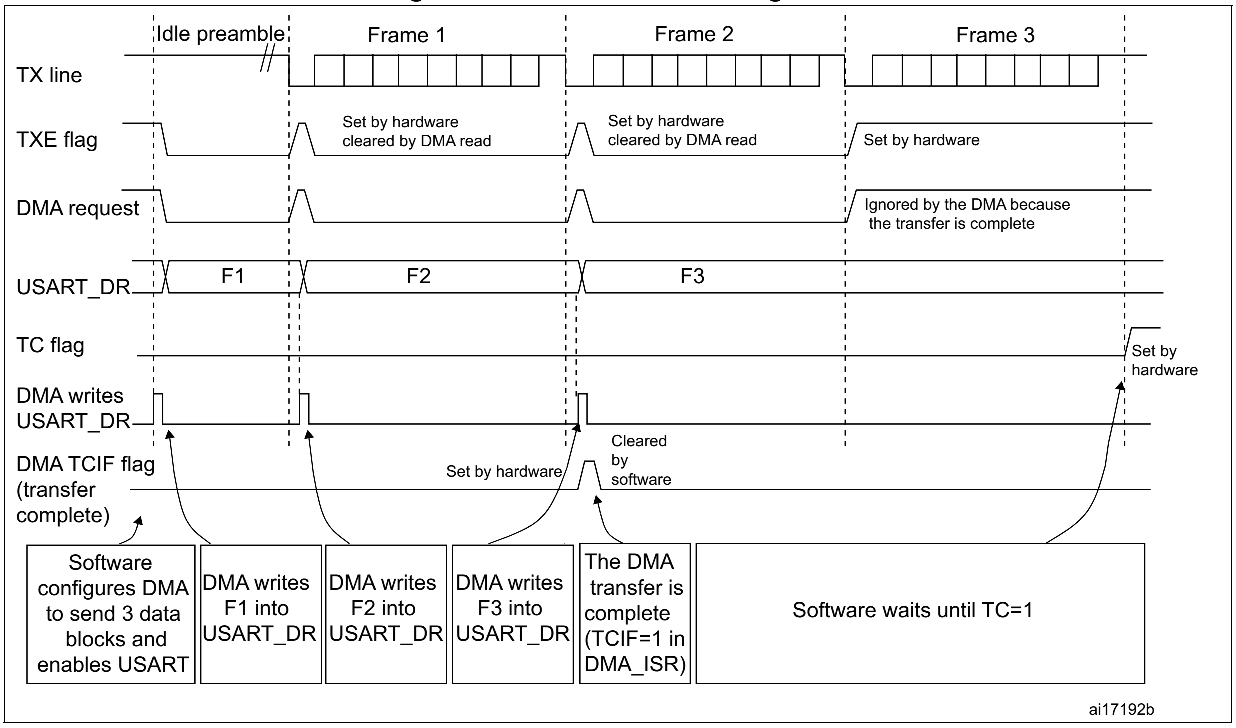
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to *Section 10: DMA controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or entering Stop mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 214. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

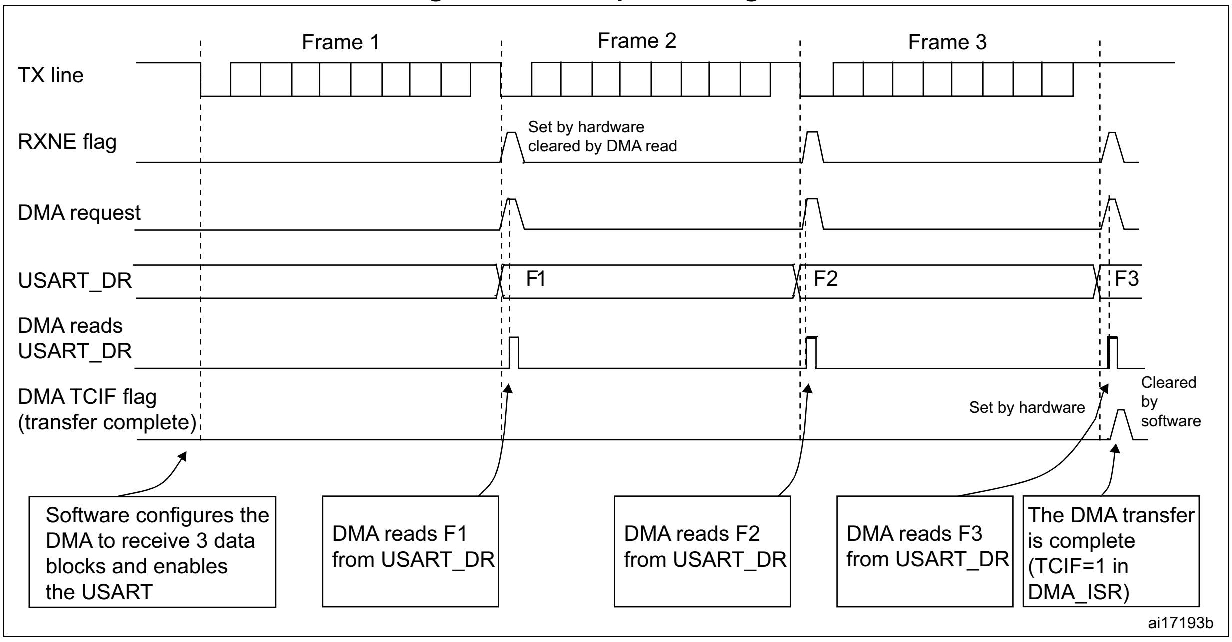
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data is loaded from the USART_RDR register to a SRAM area configured using the DMA peripheral (refer to *Section 10: DMA controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 215. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

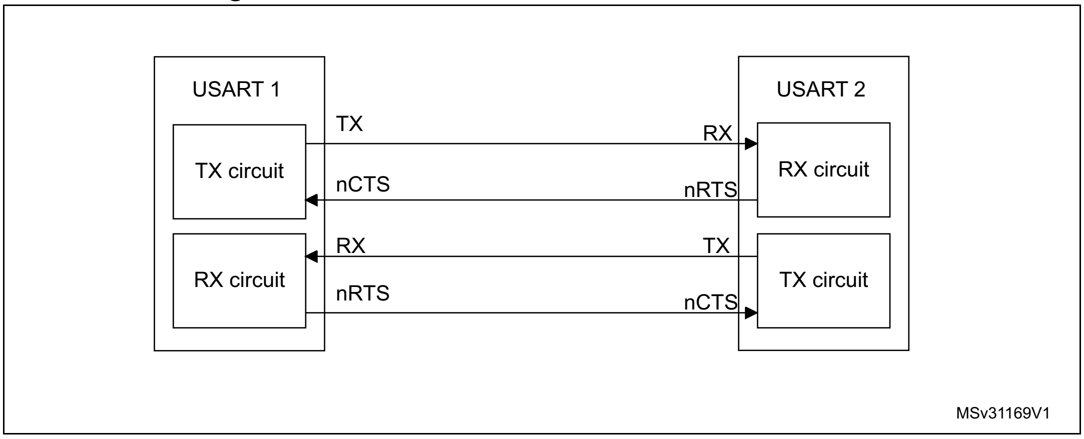
Error flagging and interrupt generation in multibuffer communication

In multibuffer communication if any error occurs during the transaction the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

25.5.18 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The *Figure 216* shows how to connect 2 devices in this mode:

Figure 216. Hardware flow control between 2 USARTs

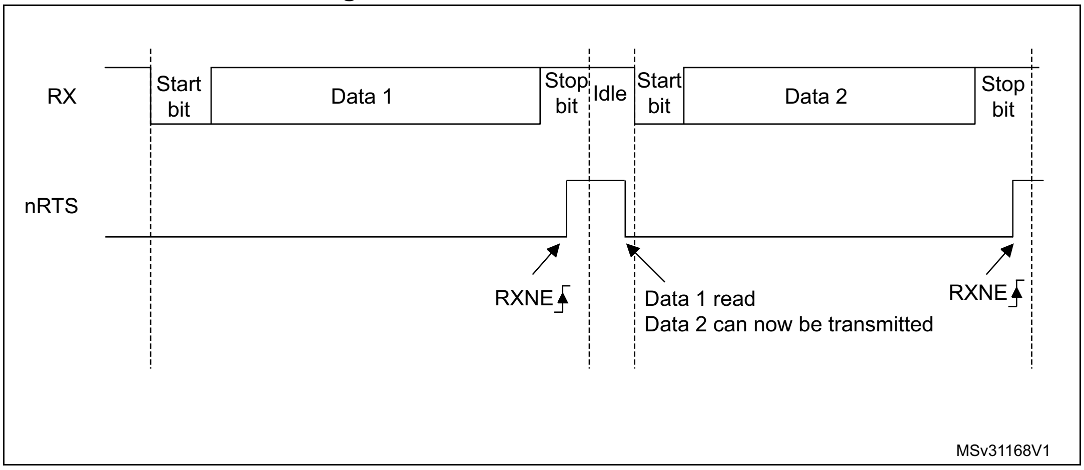


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the USART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 217* shows an example of communication with RTS flow control enabled.

Figure 217. RS232 RTS flow control



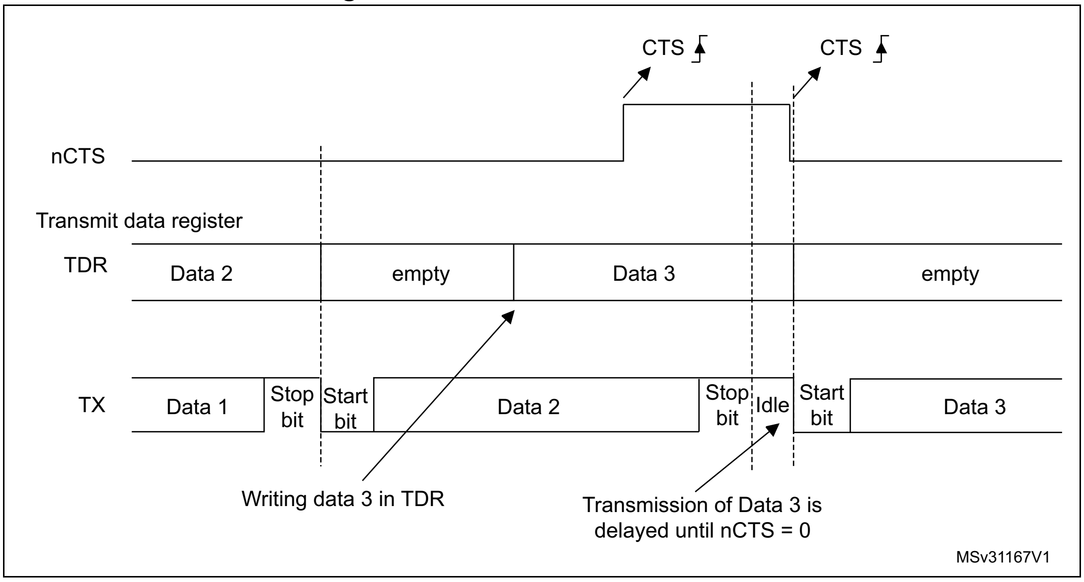
Note: When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 218* shows an example of communication with CTS flow control enabled.

Figure 218. RS232 CTS flow control



Note: For correct behavior, nCTS must be asserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 Driver Enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the START bit. It is programmed using the DEAT [4:0] bit fields in the USART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bit fields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

25.6 USART interrupts

Table 87. USART interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TXE	TXEIE
Transmit FIFO Not Full	TXFNF	TXFNFIE
Transmit FIFO Empty	TXFE	TXFEIE
CTS interrupt	CTSIF	CTSIE
Transmission Complete	TC	TCIE
Transmission Complete Before Guard Time	TCBGT	TCBGTIE
Receive data register not empty (data ready to be read)	RXNE	RXNEIE
Receive FIFO Not Empty	RXFNE	RXFNEIE
Receive FIFO Full	RXFF	RXFFIE
Overrun error detected	ORE	RXNEIE/RXF-NEIE
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
LIN break	LBDF	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NF or ORE or FE	EIE
Character match	CMF	CMIE
Receiver timeout error	RTOF	RTOIE
End of Block	EOBF	EOBIE
SPI slave underrun error	UDR	EIE

These events generate an interrupt if the corresponding Enable Control Bit is set.

25.7 USART registers

Refer to *Section 1.1: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

25.7.1 Control register 1 (USARTx_CR1)

Address offset: 0x00

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE/ TXFNFIE	TCIE	RXNEIE /RXFNEIE	IDLEIE	TE	RE	Res.	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **RXFFIE** :RXFIFO Full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated when RXFF=1 in the USART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 30 **TXFEIE** :TXFIFO empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated when TXFE=1 in the USART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 29 **FIFOEN** :FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bit field can only be written when the USART is disabled (UE=0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit, with bit 12 (M0) determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n Stop bit

M[1:0] = 01: 1 Start bit, 9 Data bits, n Stop bit

M[1:0] = 10: 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE=0).

Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and Autobaudrate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of Block interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Section 25.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bit field can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Refer to Section 25.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bit field can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Refer to Section 25.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the USART. When set, the USART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit, with bit 28 (M1) determine the word length. It is set or cleared by software. See Bit 28 (M1)description.

This bit can only be written when the USART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bit field can only be written when the USART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bit field can only be written when the USART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bit field can only be written when the USART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever PE=1 in the USART_ISR register

Bit 7 **TXEIE/TXFNFIE**: Transmit data regsiter empty/TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever TXE/TXFNF =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever TC=1 in the USART_ISR register

Bit 5 **RXNEIE/RXFNEIE**: Receive data register not empty/RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever ORE=1 or RXNE/RXFNE=1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever IDLE=1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In Smartcard mode, when TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 Reserved, must be kept at reset value

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the USART is kept, but all the status flags, in the USART_ISR are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low power mode

1: USART enabled

Note: In order to go into low power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

Note: in Smartcard mode, (SCEN = 1), the SCLK is always available when CLKEN = 1, regardless of the UE bit value.

25.7.2 Control register 2 (USARTx_CR2)

Address offset: 0x04

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS.	Res.	Res.	SLVEN.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:28 **ADD[7:4]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.

This is used in multiprocessor communication during Mute mode or Stop mode, for wakeup with 7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. It may also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.

This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.

This is used in multiprocessor communication during Mute mode or Stop mode, for wakeup with address mark detection.

This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement. (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bit field can only be written when ABREN = 0 or the USART is disabled (UE=0).

Note: If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bit field can only be written when the USART is disabled (UE=0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)

1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bit field can only be written when the USART is disabled (UE=0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: TX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the TX line.

This bit field can only be written when the USART is disabled (UE=0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: RX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the RX line.

This bit field can only be written when the USART is disabled (UE=0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another UART.

This bit field can only be written when the USART is disabled (UE=0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN Synch Breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support LIN mode, this bit is reserved and forced by hardware to ‘0’.Refer to Section 25.4: USART implementation.

Bits 13:12 **STOP[1:0]**: STOP bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bit field can only be written when the USART is disabled (UE=0).

Bit 11 **CLKEN**: Clock enable

This bit allows the user to enable the SCLK pin.

0: SCLK pin disabled

1: SCLK pin enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Note: In Smartcard mode, in order to provide correctly the SCLK clock to the smartcard, the steps below must be respected:

- UE = 0
- SCEN = 1
- GTPR configuration
- CLKEN= 1
- UE = 1

Bit 10 **CPOL**: Clock polarity

This bit allows the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on SCLK pin outside transmission window

1: Steady high value on SCLK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 207* and *Figure 208*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the SCLK pin

1: The clock pulse of the last data bit is output to the SCLK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF=1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length
This bit is for selection between 11 bit or 10 bit break detection.
0: 10-bit break detection
1: 11-bit break detection
This bit can only be written when the USART is disabled (UE=0).
Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the USART is disabled (UE=0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**
When the DSI_NSS bit is set, the NSS pin input is ignored.
0: SPI slave selection depends on NSS input pin.
1: SPI slave is always selected and NSS input pin is ignored.
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous Slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value

Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

25.7.3 Control register 3 (USARTx_CR3)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG			RXFTIE.	RXFTCFG			TCBGTIE	TXFTIE	Res.	Res.		SCARCNT2:0]			Res.
rw			rw	rw			rw	rw				rw	rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG:** TXFIFO threshold configuration

000:TXFIFO reaches 1/8 of its depth.

001:TXFIFO reaches 1/4 of its depth.

010:TXFIFO reaches 1/2 of its depth.

011:TXFIFO reaches 3/4 of its depth.

100:TXFIFO reaches 7/8 of its depth.

101:TXFIFO becomes empty.

Remaining combinations: Reserved.

Bit28 **RXFTIE**: RXFIFO threshold interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG:** Receive FIFO threshold configuration

000:Receive FIFO reaches 1/8 of its depth.

001:Receive FIFO reaches 1/4 of its depth.

010:Receive FIFO reaches 1/2 of its depth.

011:Receive FIFO reaches 3/4 of its depth.

100:Receive FIFO reaches 7/8 of its depth.

101:Receive FIFO becomes full.

Remaining combinations: Reserved.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever TCBGT=1 in the USARTx_ISR register

Note: If the USART does not support the Smartcard mode, this bit is reserved and forced by hardware to ‘0’.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bits 22:20 Reserved, must be kept at reset value.

Bit 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bit-field specifies the number of retries in transmit and receive, in Smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bit field must be programmed only when the USART is disabled (UE=0).

When the USART is enabled (UE=1), this bit field may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmit mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Refer to Section 25.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit allows the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Section 25.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on Reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for Smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE (RXFNE is the case where FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 : **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data is written directly in USARTx_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE=0).

Note: This control bit allows checking the communication flow w/o reading the data

Bit 11 **ONEBIT**: One sample bit method enable

This bit allows the user to select the sample method. When the one sample bit method is selected the noise detection flag (NF) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE=0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF=1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

This bit can only be written when the USART is disabled (UE=0)

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE=0).

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE=0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE=1 or ORE=1 or NF=1or UDR = 1 in the USART_ISR register).

0: Interrupt is inhibited

1: An interrupt is generated when FE=1 or ORE=1 or NF=1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

25.7.4 Baud rate register (USARTx_BRR)

This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000

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

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

Bits 15:4 **BRR[15:4]**
BRR[15:4] = USARTDIV[15:4]

Bits 3:0 **BRR[3:0]**
When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].
When OVER8 = 1:
BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
BRR[3] must be kept cleared.

25.7.5 Guard time and prescaler register (USARTx_GTPR)

Address offset: 0x10

Reset value: 0x0000

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

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

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

Bits 15:8 **GT[7:0]**: Guard time value

This bit-field is used to program the Guard time value in terms of number of baud clock periods.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bit field can only be written when the USART is disabled (UE=0).

Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

In IrDA Low-power and normal IrDA mode:

PSC[7:0] = IrDA Normal and Low-Power Baud Rate

Used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

In Smartcard mode:

PSC[4:0]: Prescaler value

Used for programming the prescaler for dividing the USART source clock to provide the Smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bit field can only be written when the USART is disabled (UE=0).

Note: Bits [7:5] must be kept cleared if Smartcard mode is used.

This bit field is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Refer to Section 25.4: USART implementation.

25.7.6 Receiver timeout register (USARTx_RTOR)

Address offset: 0x14

Reset value: 0x0000

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

Bits 31:24 **BLEN[7:0]**: Block Length

This bit-field gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In Smartcard mode, the Block length counter is reset when TXE=0 (TXFE = 0 in case FIFO mode is enabled).

This bit-field can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bit-field gives the Receiver timeout value in terms of number of baud clocks.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the Start Bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on the fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to “0x00000000” when the Receiver timeout feature is not supported. Refer to Section 25.4: USART implementation.

25.7.7 Request register (USARTx_RQR)

Address offset: 0x18

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w_r0	w_r0	w_r0	w_r0	w_r0

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

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, Writing 1 to this bit sets the TXE flag.
This allows to discard the transmit data. This bit must be used only in Smartcard mode, when data has not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to ‘0’
When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO . This sets the flag TXFE (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data is written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO i.e. clears the bit RXFNE.
This allows to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: In the case the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF flag in the USART_ISR and request an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to ‘0’. Refer to Section 25.4: USART implementation.

25.7.8 Interrupt & status register (USARTx _ISR)

Address offset: 0x1C

Reset value: 0x00C0 (In case FIFO disabled).

Reset value: 0x28000C0 (In case FIFO/Smartcard mode enabled).

Reset value: 0x08000C0 (In case FIFO enabled/Smartcard mode Disabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	RE ACK	TE ACK	Res.	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r		r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE/TX FNF	TC	RXNE/ RXFNE	IDLE	ORE	NF	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the programmed threshold in TXFTCFG in USARTx_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the USART_CR3 register.

0: TXFIFO doesn’t reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the programmed threshold in RXFTCFG in USARTx_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the USART_CR3 register.

0: Receive FIFO doesn’t reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG threshold is configured to «101», RXFT flag is set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USARTx_RDR. Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.

Bit 25 **TCBGT:** Transmission complete before guard time flagl

This bit indicates when the last data written in the USART_TDR has been transmitted correctly out of the shift register .

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if there is no NACK from the smartcard. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register. It is cleared by software, writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfuly (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the Smartcard mode, this bit is reserved and forced by hardware to ‘0’. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is "1".

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when RXFIFO is Full.

An interrupt is generated if the RXFFIE bit =1 in the USART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.

An interrupt is generated if the TXFEIE bit =1 (bit 30) in the USART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering Stop mode.

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted

1: Break character is transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE=1in the USART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_CR3 register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into USARTx_DR.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and forced by hardware to '0

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIE=1 in the USART_CR2 register.

It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to *Section 25.4: USART implementation.*

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 7 **TXE/TXFNF:** Transmit data register empty/TXFIFO not full

When FIFO mode is disabled, TXE is set by hardware when the content of the USARTx_TDR register has been transferred into the shift register. It is cleared by a write to the USARTx_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).

When FIFO mode is enabled, TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the USART_TDR. Every write in the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty . After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO. (TXFNF and TXFE is set at the same time).

An interrupt is generated if the TXEIE/TXFNFIE bit =1 in the USART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates when the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware if the transmission of a frame containing data is complete and if TXE/TXFE is set. An interrupt is generated if TCIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

An interrupt is generated if TCIE=1 in the USART_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 **RXNE/RXFNE**:Read data register not empty/RXFIFO not empty

RXNE bit is set by hardware when the content of the USARTx_RDR shift register has been transferred

to the USARTx_RDR register. It is cleared by a read to the USARTx_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USARTx_RQR register.

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the USART_RDR register. Every read of the USART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXNE/RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE/RXFNEIE=1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).

If mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USARTx_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled) . It is cleared by a software, writing 1 to the ORECF, in the USARTx_ICR register.

An interrupt is generated if RXNEIE/ RXFNEIE=1 or EIE = 1 in the USARTx_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NF**: START bit Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NF flag is set during multi buffer communication if the EIE bit is set.

Note: When the line is noise-free, the NF flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 25.5.6: Tolerance of the USART receiver to clock deviation).

Note: In FIFO mode, this error is associated with the character in the USARTx_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

In Smartcard mode, in transmission, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the USARTx_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: In FIFO mode, this error is associated with the character in the USARTx_RDR.

25.7.9 Interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.
														w	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

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

Bit 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bit 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag

Writing 1 to this bit clears the RTOF flag in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag

Writing 1 to this bit clears the LBDF flag in the USART_ISR register.

Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 25.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag

Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NF flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

25.7.10 Receive data register (USART_RDR)

Address offset: 0x24

Reset value: Undefined

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

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

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

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 194*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

25.7.11 Transmit data register (USART_TDR)

Address offset: 0x28

Reset value: Undefined

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

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

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

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USARTx_TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 194*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF=1.

25.7.12 Prescaler register (USARTx_PRESC)

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000

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

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

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

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaing combinations: Reserved.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is «1011» i.e. input clock divided by 256.

25.8 USART register map

The table below gives the USART register map and reset values.

Table 88. USART register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	Res.	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x04	USART_CR2	ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3	TXFTCF G[2:0]			RXFTIE	RXFTCF G[2:0]			TCBGTIE	TXFTIE	Res.	Res.	Res.	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value										0	0		0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	EOBCF	RTOCF	Res.	CTSCF	LBDCF	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value															0					0	0		0	0		0		0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 88. USART register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	USART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.2.2 on page 48* for the register boundary addresses.

26 Low power universal asynchronous receiver transmitter (LPUART)

26.1 LPUART introduction

The low power universal asynchronous receiver transmitted (LPUART) is an UART which allows bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to allow UART communications up to 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in Deepstop mode , the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memroy access) can be used for data transmission/reception.

26.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud/s to 9600 baud/s using a 32.768 kHz clock source.
- Higher baud rates can be achived by suing a higher frequency clock source
- Two internal FIFOs for transmit and receive data, that can be enabled/disabled by software. FIFOs come with status flags for FIFOs states.
- Dual clock domain allowing
  - UART functionality and wakeup from Deepstop mode
  - Convenient baud rate programming independent from the PCLK reprogramming
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytesare buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications

  The LPUART enters mute mode if the address does not match.
- Wakeup from mute mode (by idle line detection or address mark detection)

26.3 LPUART functional description

Any LPUART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX**: Receive Data Input.
  This is the serial data input.
- **TX:** Transmit Data Output.
  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

Through these pins, serial data is transmitted and received in normal LPUART mode as frames comprising:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (7 or 8 or 9 bits) least significant bit first
- 1, 2 Stop bits indicating that the frame is complete
- The LPUART interface uses a baud rate generator
- A status register (LPUART_ISR)
- Receive and transmit data registers (LPUART_RDR, LPUART_TDR)
- A baud rate register (LPUART_BRR)

Refer to *Section 26.5: LPUART registers* for the definitions of each bit.

The following pins are required in RS232 Hardware flow control mode:

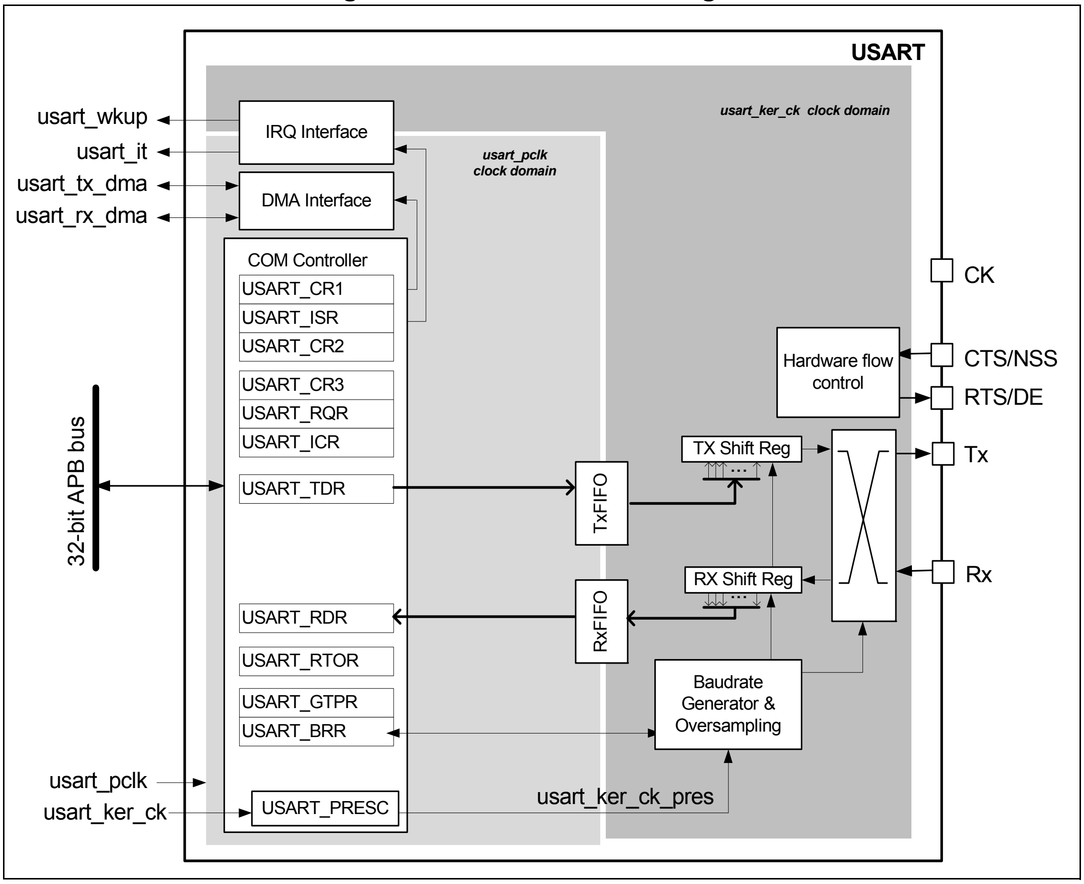
- **nCTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS:** Request to send indicates that the LPUART is ready to receive data (when low).

The following pin is required in RS485 Hardware control mode:

- **DE**: Driver Enable activates the transmission mode of the external transceiver.

Note: DE and nRTS share the same pin.

Figure 219. LPUART Block diagram



The simplified block diagram given in *Figure 219* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The lpuart_pclk clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
- The **lpuart_ker_ck** kernel clock domain

  The lpuart_ker_ck is the LPUART clock source. It is independent of the lpuart_pclk and delivered by the RCC. So, the LPUART registers can be written/read even when the lpuart_ker_ck is stopped.

There is no constraint between lpuart_pclk and lpuart_ker_ck: lpuart_ker_ck can be faster or slower than lpuart_pclk, with no more limitation than the ability for the software to manage the communication fast enough.

26.3.1 LPUART character description

Word length may be selected as being either 7 or 8 or 9 bits by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 220*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

In default configuration, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

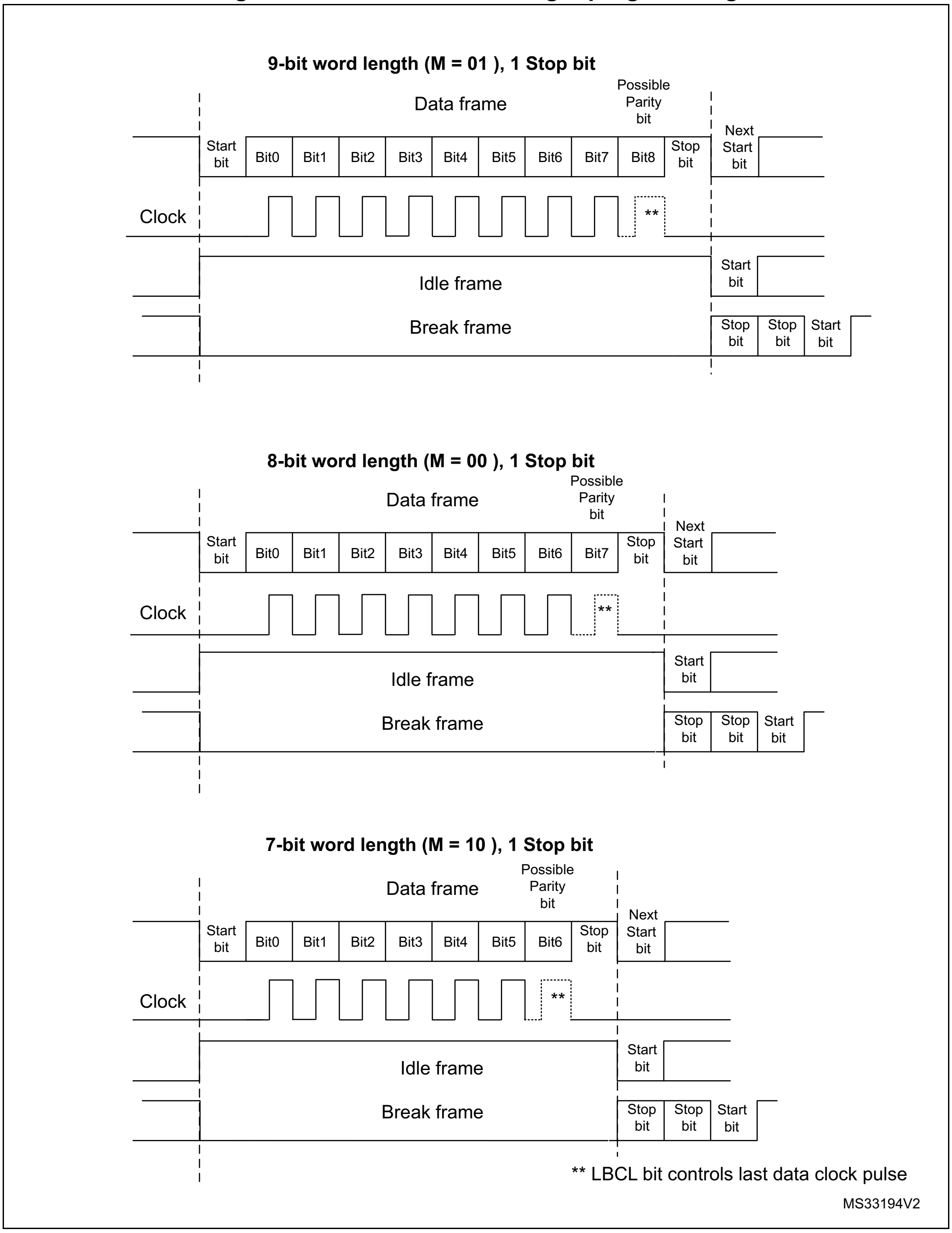
An ***Idle character*** is interpreted as an entire frame of “1”s. (The number of “1” ‘s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

Figure 220. LPUART Word length programming



26.3.2 FIFOs and thresholds

The LPUARTcan operate in FIFO mode, with the FIFO buffers having a depth of 16 bytes.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO) . The FIFO mode is enabled by setting the bit 29 FIFOEN in USARTx_CR1 register.

Being 9 bits the maximum data word length, the TXFIFO is 9-bits wide. However the RXFIFO is by default 12-bits wide. This is due to the fact that the receiver does not only put the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO with its flags. However, reading RDR provides only the data. The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupt are triggered. These thresholds are programmed through bit fields RXFTCFG and TXFTCFG in LPUART_CR3 control register.

In this case:

- The receive interrupt in generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.
- The transmit interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

RXFIFO threshold

The RXFIFO threshold is configured using the RXFTCFG bits fields in the LPUART_CR3 register.

When the number of received data is equal to the programmed RXFTCFG, the flag RXFT in the LPUART_ISR register is set.

Having RXFT flag set means that there are RXFTCFG data received: 1 data in LPUART_RDR and (RXFTCFG - 1) data in the RXFIFO. So, when the RXFTCFG is programmed to «101» , the RXFT flag is set when 16 data are received: 15 data in the RXFIFO and 1 data in the LPUARTx_RDR. Consequently, the 17th received data does not set the overrun flag.

26.3.3 Transmitter

The transmitter can send data words of either 7 or 8 or 9 bits depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see Figure 219).

When FIFO mode is enabled, data written to the LPUART_TDR register is queued in the TXFIFO.

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by LPUART: 1 and 2 stop bits.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters become frozen. The current data being transmitted is lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

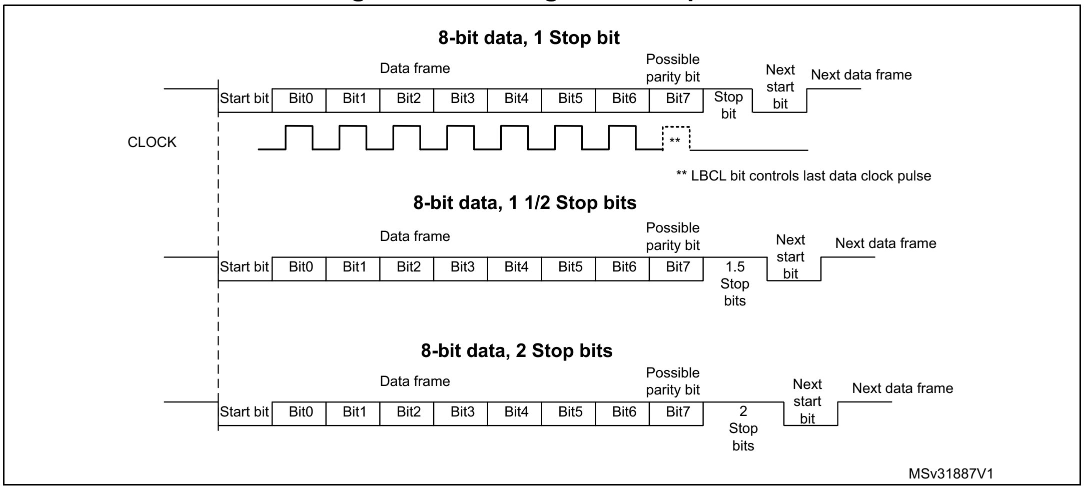
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 Stop bits***: This is supported by normal LPUART, single-wire and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 221. Configurable stop bits



Character transmission procedure

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO and write operations in the LPUART_TDR are made when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. After writing the last data into the LPUART_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the LPUART is disabled or enters the Halt mode to avoid corrupting the last transmission.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.

– When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

Single byte communication

- When FIFO mode is disabled:

Clearing the TXE bit is always performed by a write to the transmit data register.

The TXE flag is set by hardware and it indicates:

– The data has been moved from the LPUART_TDR register to the shift register and the data transmission has started.

– The LPUART_TDR register is empty.

– The next data can be written in the LPUART_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is taking place, a write instruction to the LPUART_TDR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO Not Full) flag is set by hardware and it indicates:
  – The TXFIFO is not full.
  – The LPUART_TDR register is empty.
  – The next data can be written in the LPUART_TDR register without overwriting the previous data. When a transmission is taking place, a write operation to the LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO into the shift register at the end of the current transmission.

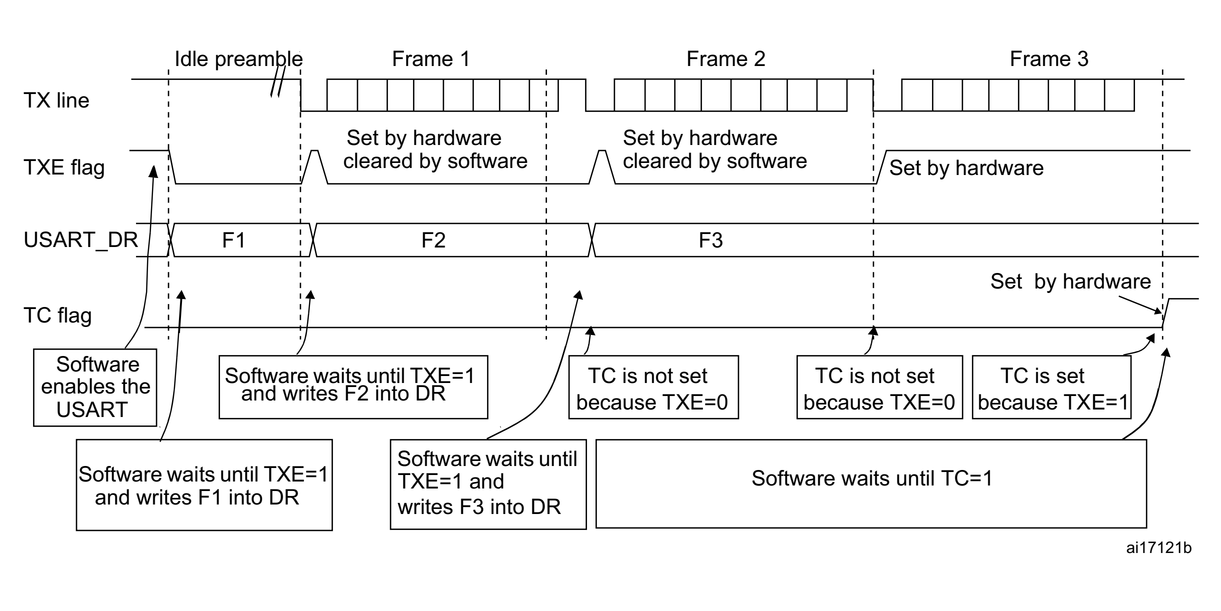
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO isfull. This flag generatesan interrupt if TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written into TXFIFO when the TXFIFO threshold ios reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC=1 before disabling the LPUART or causing the microcontroller to enter the low power mode (See *Figure 222: TC/TXE behavior when transmitting*).

Figure 222. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 220*).

If a ‘1’ is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

26.3.4 Receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In LPUART, for START bit detection, a falling edge should be detected first on the Rx line, then a sample is taken in the middle of the start bit to confirm that it is still ‘0’. If the start sample is at ‘1’, then the noise error flag (NF) is set, then the START bit is discarded and the receiver waits for a new START bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data shifts in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the RE bit LPUART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. A read of LPUART_RDR returns the oldest entry in the RXFIFO. When a data is received, it is stored in the RXFIFO, with error bits associated with that data.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is a data in the RXFIFO to be read.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE bit is performed by a software read to the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read of LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is

reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as for a received data character plus an interrupt if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled: An overrun error occurs when a character is received when RXNE has not been reset. data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  - The ORE bit is set.
  - The RDR content is not lost. The previous data is available when a read to
  - LPUART_RDR is performed.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXNEIE bit is set or EIE bit is set.
- FIFO mode enabled: an overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full. Data can not be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty. An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  - The ORE bit is set.
  - The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE bit is set or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

In case of FIFO mode is disabled, there are two possibilities

- *if RXNE=1, then the last valid data is stored in the receive register RDR and can be read,*
- *if RXNE=0, then it means that the last valid data has already been read and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

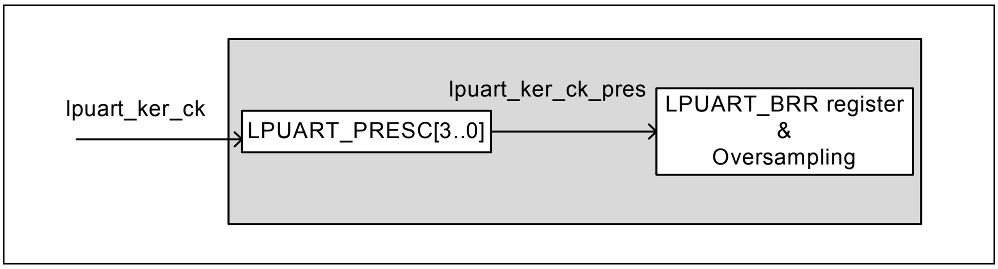
The choice of the clock source is done through the Clock Control system (see the *Section 6: Reset and clock controller (RCC)*). The clock source must be chosen before enabling the LPUART (by setting the UE bit).

The choice of the clock source must be done according to two criteria:

- Possible use of the LPUART in low power mode
- Communication speed.

The clock source frequency is lpuart_kern_ck.

When the dual clock domain and the wakeup from Deepstop mode features are supported, the lpuart_kern_ck clock source can be selected through *Chapter 6.6.2: Clock configuration register (RCC_CFGR)*. The lpuart_kern_ck can be divided by a programmable factor in the LPUARTx_PRESC register.



Choosing lpuart_kern_ck as LSE clock source may allow the LPUART to receive data while the MCU is in low power mode. Depending on the received data and wakeup mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active in order to allow LPUART communication.

When the LPUART clock source is configured to be LSE, it is possible to keep enabled this clock during Deepstop mode by setting the UCESM bit in LPUART_CR3 control register.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming baud as close as possible to the middle of the baud-period. Only a single sample is taken of each of the incoming bauds.

Note: There is no noise detection for data.

Framing error

A framing error is detected when:

The stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of

multibuffer communication an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode.

- ***1 stop bit***: Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
- ***2 stop bits***: Sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

26.3.5 Baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times f_{CKPRES}}{\text{LPUARTDIV}}$$

LPUARTDIV is coded on the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence the baud rate register value should not be changed during communication.

It is forbidden to write values less than 0x300 in the LPUART_BRR register.

fck must be in the range [3 x baudrate ..4096 x baudrate]

Table 89. Error calculation for programmed baudrates at f_{ck} = 32,768 KHz

Baud rate		f_{CK} = 32,768 KHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	300 Bps	300 Bps	0x6D3A	0
2	600 Bps	600 Bps	0x369D	0
3	1200 Bps	1200.087 Bps	0x1B4E	0.007
4	2400 Bps	2400.17 Bps	0xDA7	0.007
5	4800 Bps	4801.72 Bps	0x6D3	0.035
6	9600 Bps	9608.94 Bps	0x369	0.093

Table 90. Error calculation for programmed baudrates at f_{ck} = 16 MHz

Baud rate		f_{CK} = 16 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	9.6 kBps	9.60001 kBps	0x682AA	0
2	19.2 kBps	19.20003 kBps	0x34155	0
3	38.4 kBps	38.40024 kBps	0x1A0AA	0
4	57.6 kBps	57.60009 kBps	0x115C7	0
5	115.2 kBps	115.2018 kBps	0x8AE3	0.001
6	230.4 kBps	230.41008 kBps	0x4571	0.004
7	460.8 kBps	460.84608 kBps	0x22B8	0.01
8	921.6 kBps	921.69216 kBps	0x115C	0.01
9	1843.2 kBps	1843.38433 kBps	0x8AE	0.01
10	3686.4 kBps	3686.76867 kBps	0x457	0.01

26.3.6 Multiprocessor communication

It is possible to perform multiprocessor communication with the LPUART (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In order to use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two UCLK cycles) otherwise mute mode might remain active.

In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

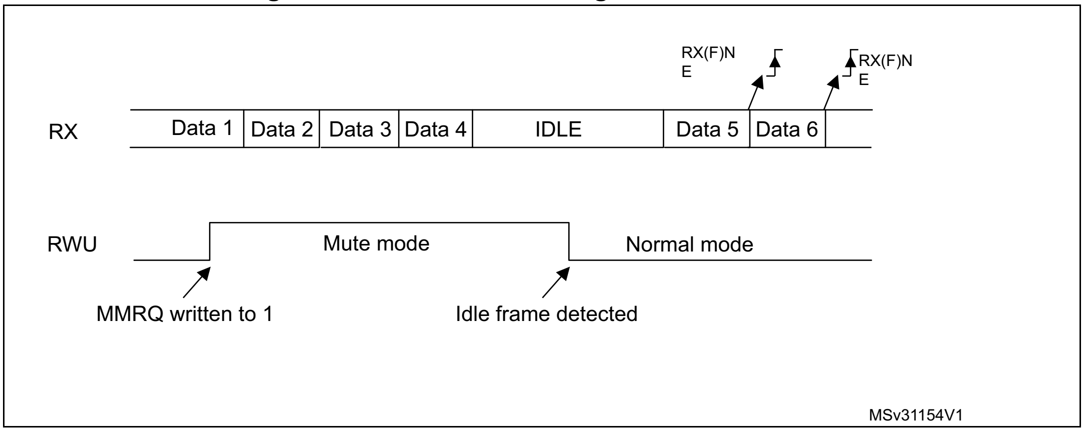
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 223*.

Figure 223. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’ otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

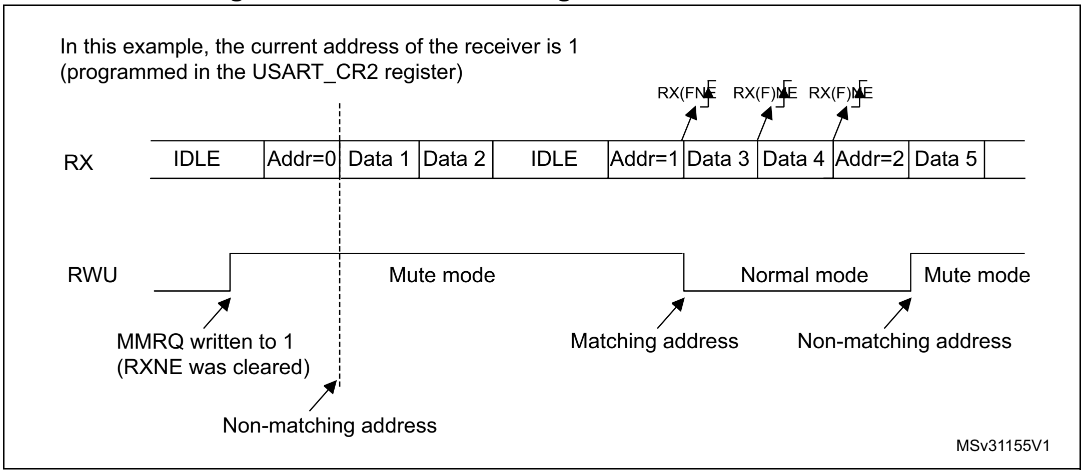
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 224*.

Figure 224. Mute mode using address mark detection



26.3.7 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 91*.

Table 91. Frame formats
Table 92:

M bits	PCE bit	LPUART frame[(1)]
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

26.3.8 Single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the LPUART_CR2 register,
- SCEN and IREN bits in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflicts on the line must be managed by software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART, in the case of 1-STOP bit configuration, the RXNE flag is set in the middle of the STOP bit.

26.3.9 Continuous communication using DMA

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to *Section 25.4: USART implementation* *to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in* *Section 26.3.4: Receiver**. To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART_ISR register.*

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to

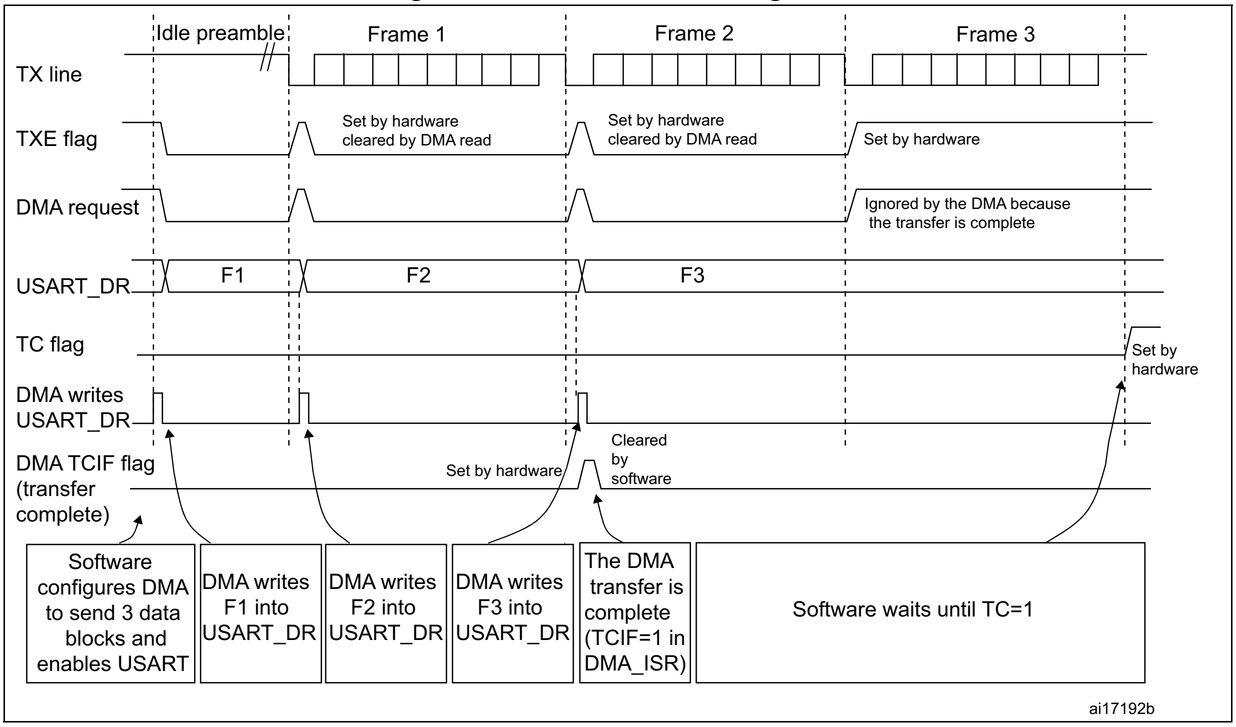
Section 10: DMA controller (DMA)) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the LPUART communication is complete. This is required to avoid corrupting the last transmission before disabling the LPUART or entering Deepstop mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 225. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

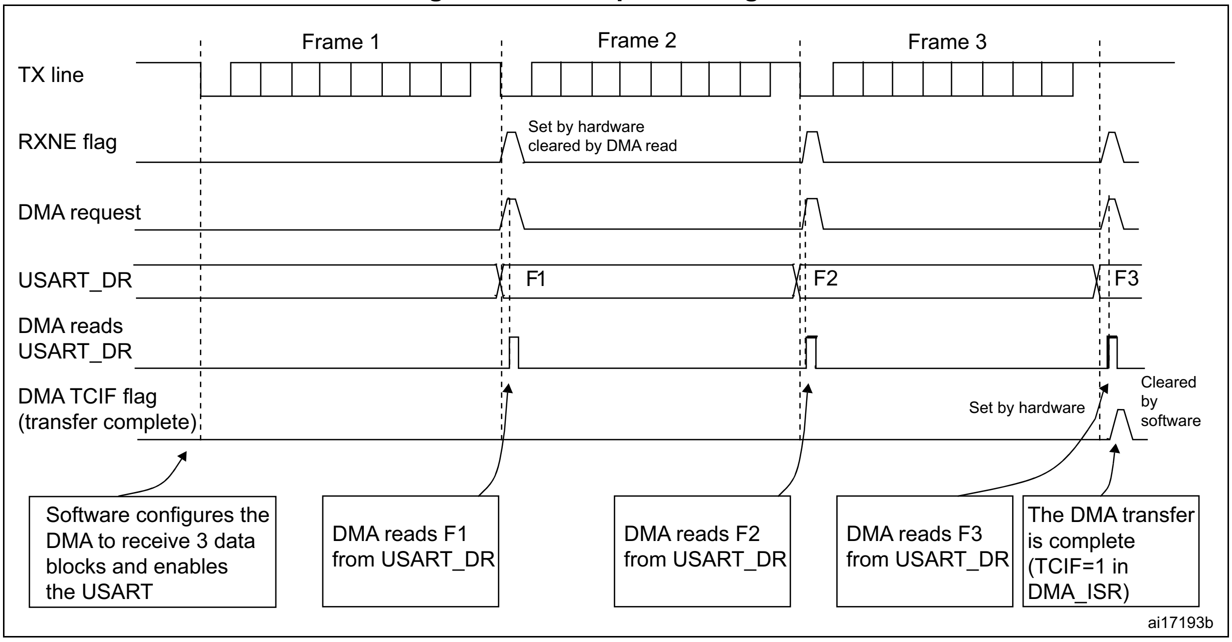
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data is loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to Section 10: DMA controller (DMA)) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 226. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

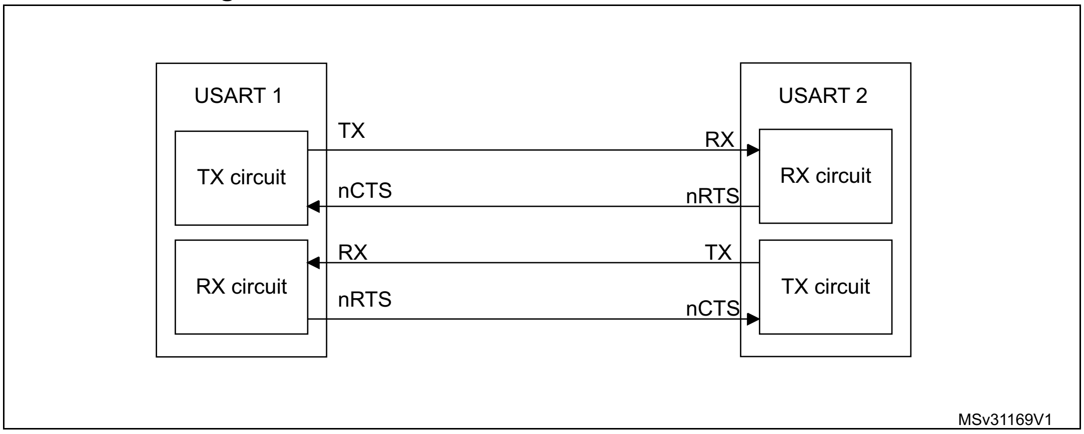
Error flagging and interrupt generation in multibuffer communication

In multibuffer communication if any error occurs during the transaction the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

26.3.10 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The *Figure 227* shows how to connect 2 devices in this mode:

Figure 227. Hardware flow control between 2 LPUARTs

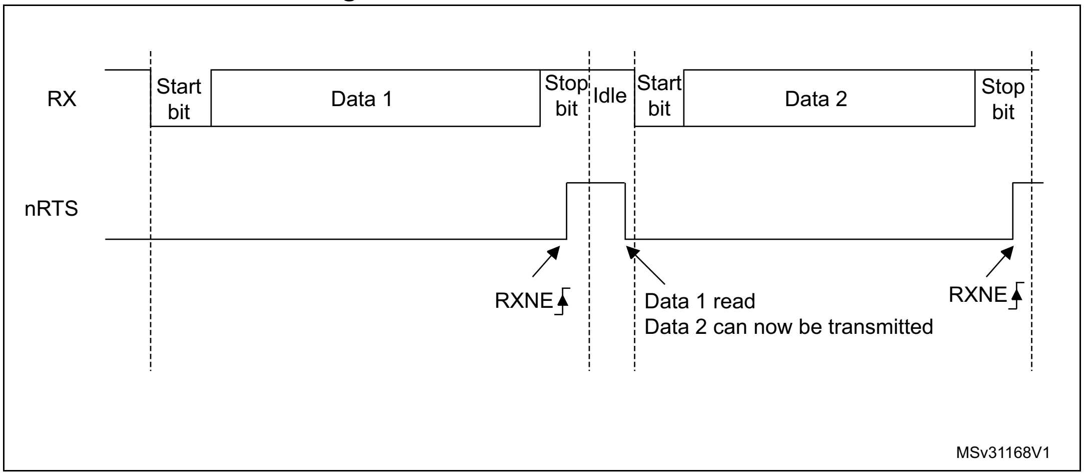


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 228* shows an example of communication with RTS flow control enabled.

Figure 228. RS232 RTS flow control



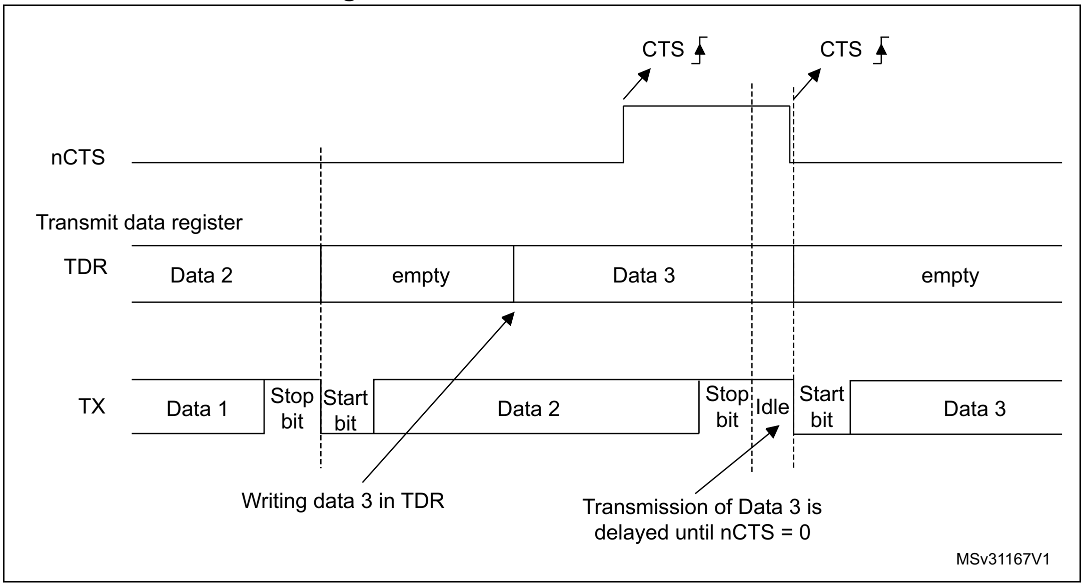
Note: When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 229* shows an example of communication with CTS flow control enabled.

Figure 229. RS232 CTS flow control



Note: For correct behavior, nCTS must be asserted at least 3 LPUART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 Driver Enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the START bit. It is programmed using the DEAT [4:0] bit fields in the LPUART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bit fields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

In LPUART, the DEAT and DEDT are expressed in LPUART clock source (f_{CK}) cycles:

- The Driver enable assertion time =
  - (1 + (DEAT x P)) x f_{CK} , if P <> 0
  - (1 + DEAT) x f_{CK} , if P = 0
- The Driver enable de-assertion time =
  - (1 + (DEDT x P)) x f_{CK} , if P <> 0
  - (1 + DEDT) x f_{CK} , if P = 0

with P = BRR[20:11]

26.3.11 Wakeup from Deepstop mode

The LPUART is able to wake up the MCU from Deepstop mode when the UESM bit is set and the LPUART clock is set to LSE (refer to the *Reset and clock control (RCC) section*.

When FIFO mode is disabled, the MCU wakeup from Deepstop mode can be done using the standard RXNE interrupt. In this case, the RXNEIE bit must be set before entering Deepstop mode.

When FIFO mode is enabled, the MCU wakeup from Deepstop mode can be done using the:

- standard RXFIFO not empty interrupt. In this case, the RXFNEIE bit must be set before entering Deepstop mode.
- standard RXFIFO full interrupt. In this case, the RXFFIE bit must be set before entering Deepstop mode.
- standard TXFIFO empty interrupt. In this case, the TXFEIE bit must be set before entering Deepstop mode. This allows sending the data in the TXFIFO during Deepstop mode. When all data are sent (i.e. TXFIFO is empty), the MCU wakes up from Deepstop mode.

In order to avoid overrun/underrun errors and transmit/receive data in Deepstop mode, the MCU wakeup from Deepstop mode can be done also using the:

- standard TXFIFO threshold interrupt.
- standard RXFIFO threshold interrupt.

An application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from Deepstop mode allows doing as many USART transfers as possible during Deepstop mode with the benefit of optimizing consumption.

Alternatively, a specific interrupt may be selected through the WUS bit fields.

In order to be able to wake up the MCU from Deepstop mode, the UESM bit in the USART_CR1 control register must be set prior to entering Deepstop mode.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUFIE bit is set.

Note: Before entering Deepstop mode, the user must ensure that the USART is not performing a transfer. BUSY flag cannot ensure that Deepstop mode is never entered during a running reception.

Note: The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in Deepstop or in an active mode.

Note: When entering Deepstop mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the USART is actually enabled.

Note: When DMA is used for reception, it must be disabled before entering Deepstop mode and re-enabled upon exit from Deepstop mode.

Note: The wakeup from Deepstop mode feature is not available for all modes. For example it doesn't work in SPI mode because the SPI operates in master mode only.

Note: When FIFO is enabled, the wakeup from Deepstop mode on address match is only possible when mute mode is enabled.

Using Mute mode with Deepstop mode

If the USART is put into Mute mode before entering Deepstop mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in Deepstop mode.
- If the wakeup from Mute mode on address match is used, then the source of wake-up from Deepstop mode must also be the address match. If the RXNE flag is set when entering the Deepstop mode, the interface remains in mute mode upon address match and wake up from Deepstop.

Note: When FIFO management is enabled,mute mode is used with wakeup from Deepstop mode without any constraints (i.e.the two points mentioned above about mute and Deepstop mode are valid only when FIFO management is disabled).

26.4 LPUART interrupts

Table 93. LPUART interrupt requests

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				lpuart_it	lpuart_wkup
Transmit data register empty	TXE	TXEIE	TXE cleared when a data is written in TDR	YES	NO
Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFNF cleared when TXFIFO is full.	YES	NO
Transmit FIFO Empty	TXFE	TXFEIE	TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.	YES	YES
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT cleared by hardware when the TXTFIFO content is less than programmed threshold	YES	YES
CTS interrupt	CTSIF	CTSIE	CTSIF cleared by software by setting CTSCF bit.	YES	NO
Transmission Complete	TC	TCIE	TC cleared when a data is written in TDR or by setting TCCF bit.	YES	NO
Receive data register not empty (data ready to be read)	RXNE	RXNEIE	RXNE cleared by reading RDR or by setting RXFRQ bit.	YES	YES
Receive FIFO Not Empty	RXFNE	RXFNEIE	RXFNE cleared when the RXFIFO is emty or by setting RXFRQ bit.	YES	YES
Receive FIFO Full	RXFF(1)	RXFFIE	RXFF cleared when the RXFIFO contains at least one data.	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT cleared by hardware when the RXFIFO content is less than programmed threshold	YES	YES
Overrun error detected	ORE	RX-NEIE/RX-FNEIE	ORE cleared by setting ORECF bit.	YES	NO

Table 93. LPUART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				lpuart_it	lpuart_wkup
Idle line detected	IDLE	IDLEIE	IDLE cleared by setting IDLECF bit.	YES	NO
Parity error	PE	PEIE	PE cleared by setting PECF bit.	YES	NO
Noise error, Overrun error and Framing Error in multi-buffer communication.	NE or ORE or FE	EIE	NE cleared by setting NECF bit. ORE cleared by setting ORECF bit. FE flag cleared by setting FECF bit.	YES	NO
Character match	CMF	CMIE	CMF cleared by setting CMCF bit.	YES	NO
Wakeup from low power mode	WUF(2)	WUFIE	WUF is cleared by setting WUCF bit.	YES	YES

1. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In DEESTOP mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
2. The WUF interrupt is active only in low power mode

26.5 LPUART registers

Refer to *Section 1.1: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

26.5.1 Control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE/ TXFNFIE	TCIE	RXNEIE / RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE** :RXFIFO Full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated when RXFF=1 in the LPUART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 30 **TXFEIE** :TXFIFO empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated when TXFE=1 in the LPUART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 29 **FIFOEN** :FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit, with bit 12 (M0) determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n Stop bit

M[1:0] = 01: 1 Start bit, 9 Data bits, n Stop bit

M[1:0] = 10: 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the LPUART is disabled (UE=0).

Note: In 7-bits data length mode, the Smarcard mode, LIN master mode and Autobaudrate (0x7F and 0x55 frames detection) are not supported.

Bit 27 Reserved, must be kept at reset value

Bit 26 Reserved, must be kept at reset value

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in UCLK (LPUART clock) clock cycles. For more details, refer to RS485 Driver Enable paragraph.

This bit field can only be written when the LPUART is disabled (UE=0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal..It is expressed in UCLK (LPUART clock) clock cycles. For more details, refer to RS485 Driver Enable paragraph.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 15 Reserved, must be kept at reset value

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit, with bit 28 (M1) determine the word length. It is set or cleared by software. See Bit 28 (M1)description.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 **TXEIE/TXFNFIE**: Transmit data regsiter empty/TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 **RXNEIE/RXFNEIE**: Receive data register not empty/RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever ORE=1 or RXNE/RXFNE=1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Note: During transmission, a “0” pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bits 1 **UESM**: LPUART enable in Deepstop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from Deepstop mode.

When this bit is set, the LPUART is able to wake up the MCU from Deepstop mode, provided that the LPUART clock selection is LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from Deepstop mode.

1: LPUART able to wake up the MCU from Deepstop mode. When this function is active, the clock source for the LPUART must be LSE (see RCC chapter)

Note: It is recommended to set the UESM bit just before entering Deepstop mode and clear it on exit from Deepstop mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low power mode

1: LPUART enabled

Note: In order to go into low power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

26.5.2 Control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:28 **ADD[7:4]**: Address of the LPUART node

This bit-field gives the address of the LPUART node or a character code to be recognized.

This is used in multiprocessor communication during Mute mode or Deepstop mode, for wakeup with 7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. It may also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.

This bit field can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the LPUART node

This bit-field gives the address of the LPUART node or a character code to be recognized.

This is used in multiprocessor communication during Mute mode or Deepstop mode, for wakeup with address mark detection.

This bit field can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

Bit 23:20 Reserved, must be kept at reset value

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)

1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: TX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the TX line.

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: RX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the RX line.

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another UART.

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 14 Reserved, must be kept at reset value

Bits 13:12 **STOP[1:0]**: STOP bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bit field can only be written when the LPUART is disabled (UE=0).

Bit 11:5 Reserved, must be kept at reset value

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE=0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

26.5.3 Control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG			RXFTIE.	RXFTCFG			Res.	TXFTIE	WUFIE	WUS[2:0]		Res.	Res.	Res.	Res.
rw			rw	rw				rw	rw	rw	rw				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DEP	DEM	DDRE	OVR DIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HD SEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
010:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Remaining combinations: Reserved.

Bit28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
010:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Remaining combinations: Reserved.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from Deepstop mode interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever WUF=1 in the LPUART_ISR register

Note: WUFIE must be set before entering in Deepstop mode.
The WUF interrupt is active only in Deepstop mode.
If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to '0'.

Bit 21:20 **WUS[1:0]**: Wakeup from Deepstop mode interrupt flag selection
This bit-field specify the event which activates the WUF (Wakeup from Deepstop mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01:Reserved.
10: WUF active on Start bit detection
11: WUF active on RXNE.
This bit field can only be written when the LPUART is disabled (UE=0).

Note: If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to '0'.

Bit 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode

This bit allows the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on Reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 : Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.

This bit can only be written when the LPUART is disabled (UE=0).

Note: This control bit allows checking the communication flow w/o reading the data.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

This bit can only be written when the LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bit 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of Single-wire Half-duplex mode
0: Half duplex mode is not selected
1: Half duplex mode is selected
This bit can only be written when the LPUART is disabled (UE=0).

Bit 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE=1 or ORE=1 or NF=1 in the LPUART_ISR register.

26.5.4 Baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000

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

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

Bits 19:0 **BRR[19:0]**

Note: It is forbidden to write values less than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be > = 0x300 and LPUART_BRR is 20 bits, a care should be taken when generating high baudrates using high fck values. fck must be in the range [3 x baudrate ..4096 x baudrate].

26.5.5 Request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000

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

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

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data is written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This allows to discard the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: In the case the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value

26.5.6 Interrupt & status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x00C0 (In case FIFO disabled)

Reset value: 0x08000C0 (In case FIFO enabled)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
					r	r		r	r	r	r	r	r	r	r

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

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the programmed threshold in TXFTCFG in LPUARTx_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO doesn’t reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the programmed threshold in RXFTCFG in LPUARTx_CR3 register i.e. the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the LPUART_CR3 register.

0: Receive FIFO doesn’t reach the programmed threshold.

1: ReceiveFIFO reached the programmed threshold.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when RXFIFO is Full.

An interrupt is generated if the RXFFIE bit =1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit =1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering Deepstop mode.

Note: If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to ‘0’.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from Deepstop mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bit field. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

The WUF interrupt is active only in Deepstop mode.

If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to ‘0’.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to '0'.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_RQR register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted

1: Break character is transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1in the LPUART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception on going

Bit 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE/TXFNF:** Transmit data register empty/TXFIFO not full

When FIFO mode is disabled, TXE is set by hardware when the content of the LPUARTx_TDR register has been transferred into the shift register. It is cleared by a write to the LPUARTx_TDR register.

When FIFO mode is enabled, TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty . After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO. (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXEIE/TXFNFIE bit =1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE/TXFF is set. An interrupt is generated if TCIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART_ICR register or by a write to the LPUART_TDR register.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 **RXNE/RXFNE**:Read data register not empty/RXFIFO not empty

RXNE bit is set by hardware when the content of the LPUARTx_RDR shift register has been transferred

to the LPUARTx_RDR register. It is cleared by a read to the LPUARTx_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUARTx_RQR register.

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXNE/RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE/RXFNEIE=1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).

If mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUARTx_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled) . It is cleared by a software, writing 1 to the ORECF, in the LPUARTx_ICR register.

An interrupt is generated if RXNEIE/ RXFNEIE=1 or EIE = 1 in the LPUARTx_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NF**: START bit Noise detection flag

This bit is set by hardware when noise is detected on the START bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NF flag is set during multi buffer communication if the EIE bit is set.

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

In Smartcard mode, in transmission, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

26.5.7 Interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w_r0			w_r0	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
						w_r0			w_r0		w_r0	w_r0	w_r0	w_r0	w_r0

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

Bit 20 **WUCF**: Wakeup from Deepstop mode clear flag

Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

Note: If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to ‘0’.

Bit 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bit 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8:7 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NCF**: Noise detected clear flag

Writing 1 to this bit clears the NF flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag

Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag

Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

26.5.8 Receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: Undefined

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

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

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 219*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

26.5.9 Transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: Undefined

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

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

Bits 8:0 **TDR[8:0]**: Transmit data value
Contains the data character to be transmitted.
The TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 219*).
When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF=1.

26.5.10 Prescaler register (LPUART_PRESC)

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000

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

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

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaing combinations: Reserved.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is «1011» i.e. input clock divided by 256.

26.5.11 LPUART register map

The table below gives the LPUART register map and reset values.

Table 94. LPUART register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1	Res.	Res.	Res.	M1	Res.	Res.	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	Res.	CMIE	MME	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value				0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		WUFIE	WUS [1:0]		Res.	Res.	Res.	Res.	DEP	DEM	DDRE		Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value										0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value										0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	Res.	ORECF	NCF	FECF	PECF
	Reset value												0			0								0			0			0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

27 Serial peripheral interface / inter-IC sound (SPI/I2S)

The STM32WL30xx/31xx/33xx embeds one SPI: SPI3.

The STM32WL31xx and STM32WL33xx embed two SPIs: SPI1, SPI3.

In the device, only SPI3 supports the I2S protocol in addition to SPI features. SPI1 does not support I2S.

27.1 Introduction

The SPI/I²S interface can be used to communicate with external devices using the SPI protocol or the I^2S audio protocol. SPI or I^2S mode is selectable by software. SPI Motorola mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

The Inter-IC sound (I^2S) protocol is also a synchronous serial communication interface.It can operate in slave or master mode with full duplex and half-duplex communication. It can address four different audio standards including the Philips I^2S standard, the MSB- and LSB-justified standards and the PCM standard.

27.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4-bit to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to $f_{PCLK}/2$.
- Slave mode frequency up to $f_{PCLK}/2$.
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- SPI TI mode support

27.3 I2S main features

- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode, overrun flag in reception mode (master and slave) and Frame Error Flag in reception and transmitter mode (slave only)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-Justified standard (Left-Justified)
  - LSB-Justified standard (Right-Justified)
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock can be output to drive an external audio component. Ratio is fixed at $256 \times F_S$ (where F_S is the audio sampling frequency)

27.4 SPI/I2S implementation

This manual describes the full set of features implemented in SPI1 (STM32WL33xx/31xx only), and SPI3.

The *Table 95* describes the SPI/I2S implementation in the STM32WL30xx/31xx/33xx.

Table 95. SPI implementation(1)

SPI Features	SPI1	SPI3
Hardware CRC calculation	X	X
Rx/Tx FIFO	X	X
NSS pulse mode	X	X
I2S mode	-	X
TI mode	X	X

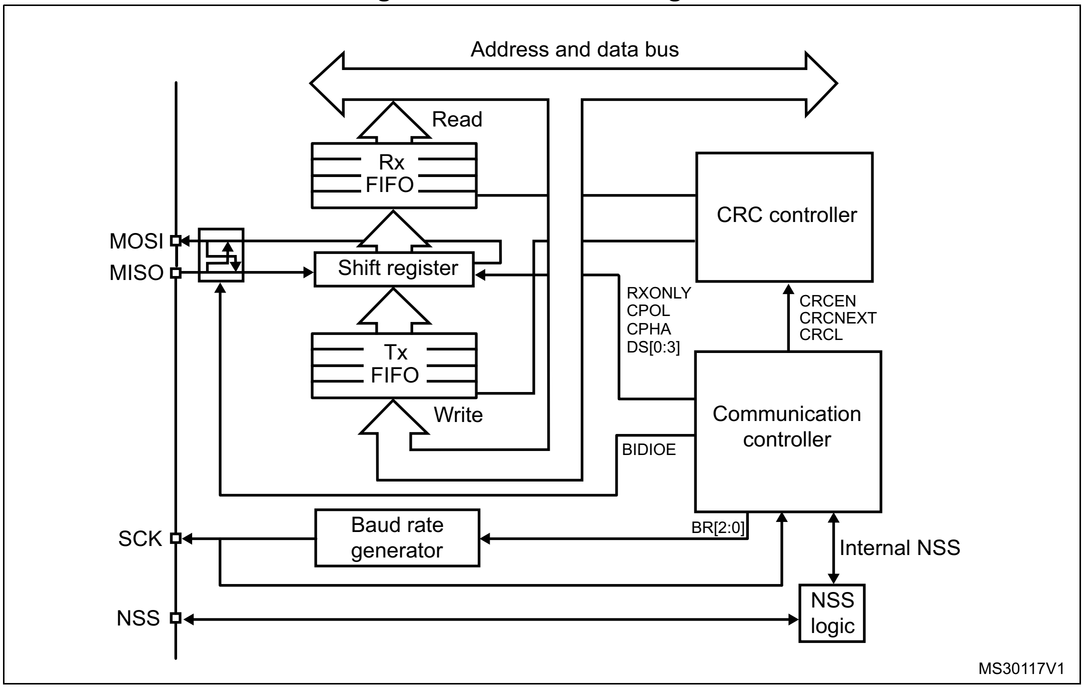
1. X = supported.

27.5 SPI functional description

27.5.1 General description

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram *Figure 230*.

Figure 230. SPI block diagram



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.

Note: If the SPI is in master mode and the internal pull-up/-down of the pad is used, the software must take care to activate the pull polarity (up or down) of the I/O to be coherent with the CPOL programming (pull-down if CPOL=0 and pull-up if CPOL=1).

- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

  See *Section 27.5.4: Slave select (NSS) pin management* for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

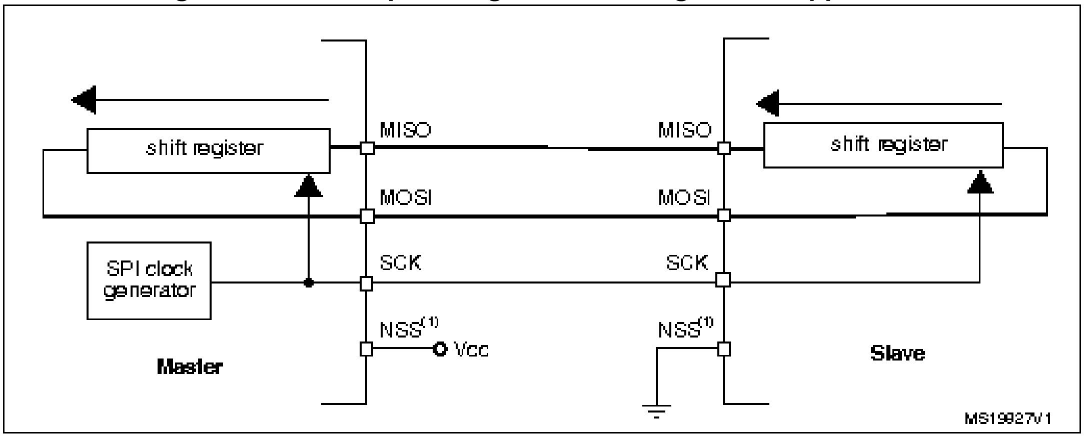
27.5.2 Communications between one master and one slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management). Communication is always initiated by the master.

Full-duplex communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 231. Full-duplex single master/ single slave application

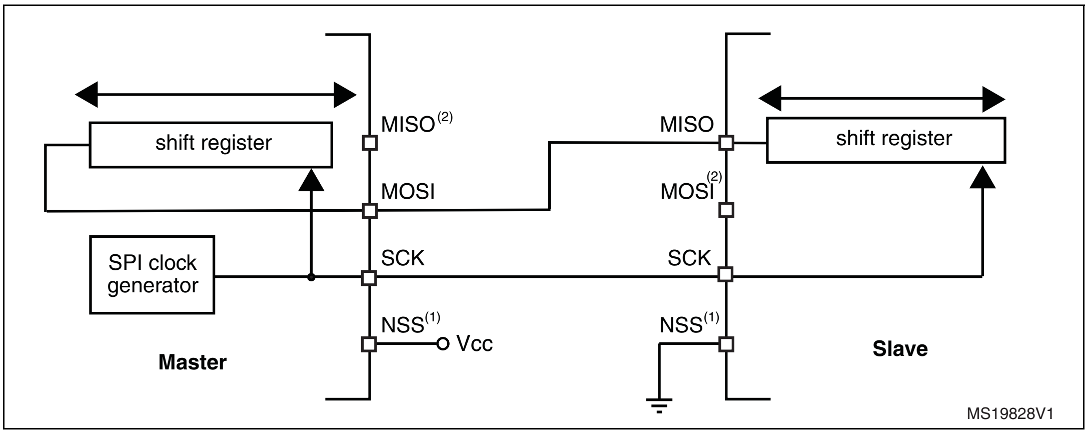


1. The NSS pin is configured as an input in this case.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx_CR1 registers. In this configuration, the master’s MISO pin and the slave’s MOSI pin are free for other application uses and act as GPIOs.

Figure 232. Half-duplex single master/ single slave application



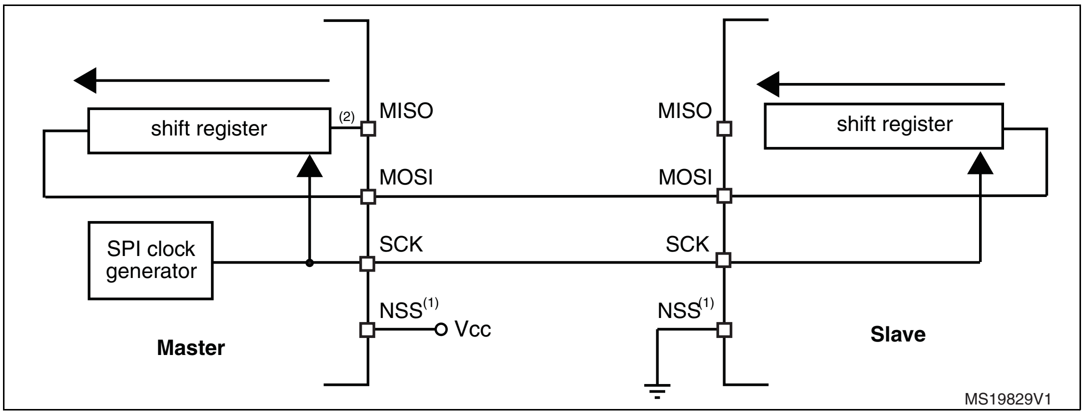
1. The NSS pin is configured as an input in this case.
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx_CR2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1)**: The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *27.5.4: Slave select (NSS) pin management*). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

Figure 233. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)



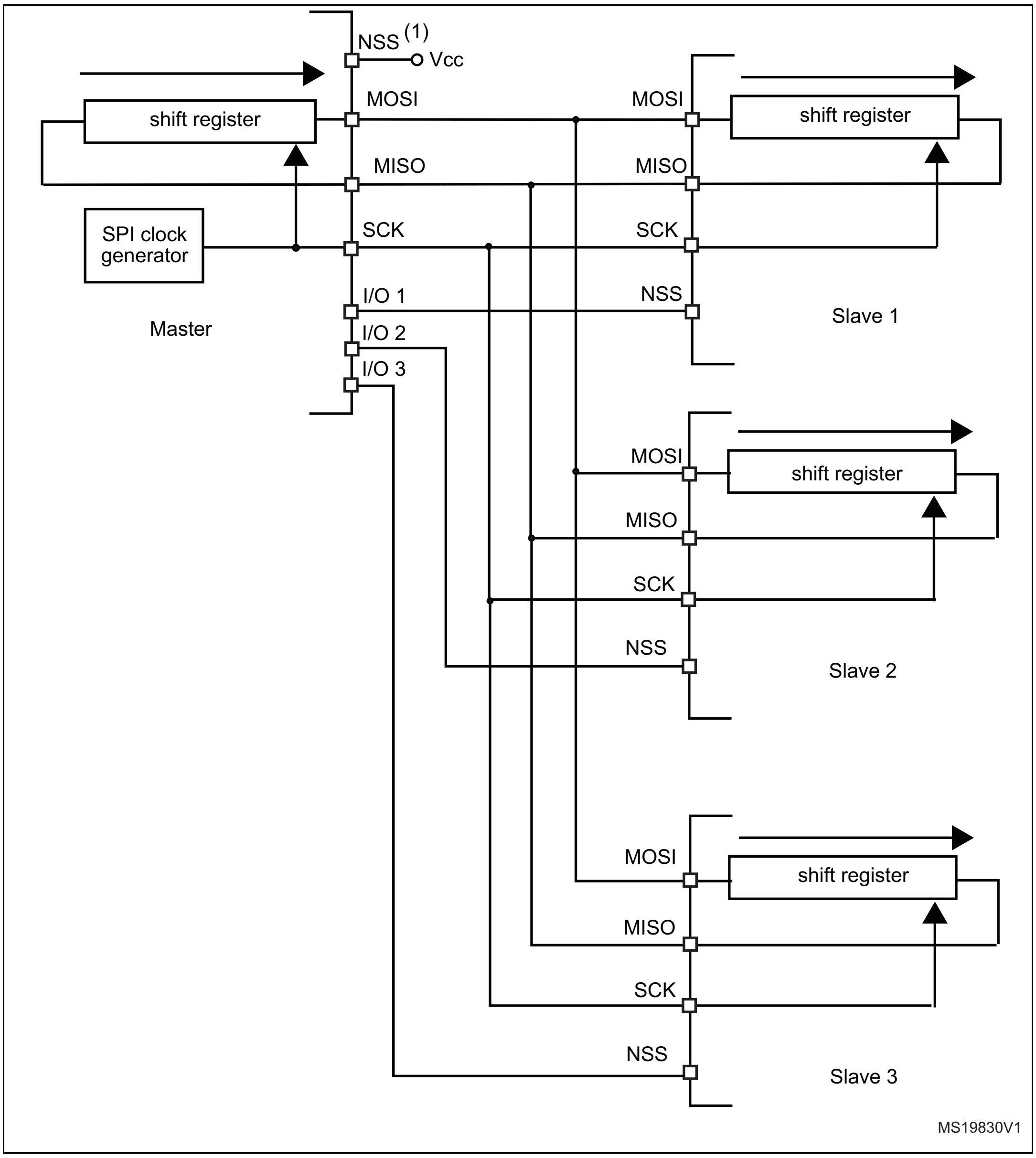
1. The NSS pin is configured as an input in this case.
2. The input information is captured in the shift register and must be ignored in standard transmit only mode (for example, OVF flag)
3. In this configuration, both the MISO pins can be used as GPIOs.

Note: Any simplex communication can be alternatively replaced by a variant of the half duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).

27.5.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see *Figure 234.*). The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input. When this is done, a standard master and dedicated slave communication is established.

Figure 234. Master and three independent slaves



1. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see *Table 8: GPIO alternate options AF0 - AF3* and *Table 9: GPIO alternate options AF4 - AF6*).

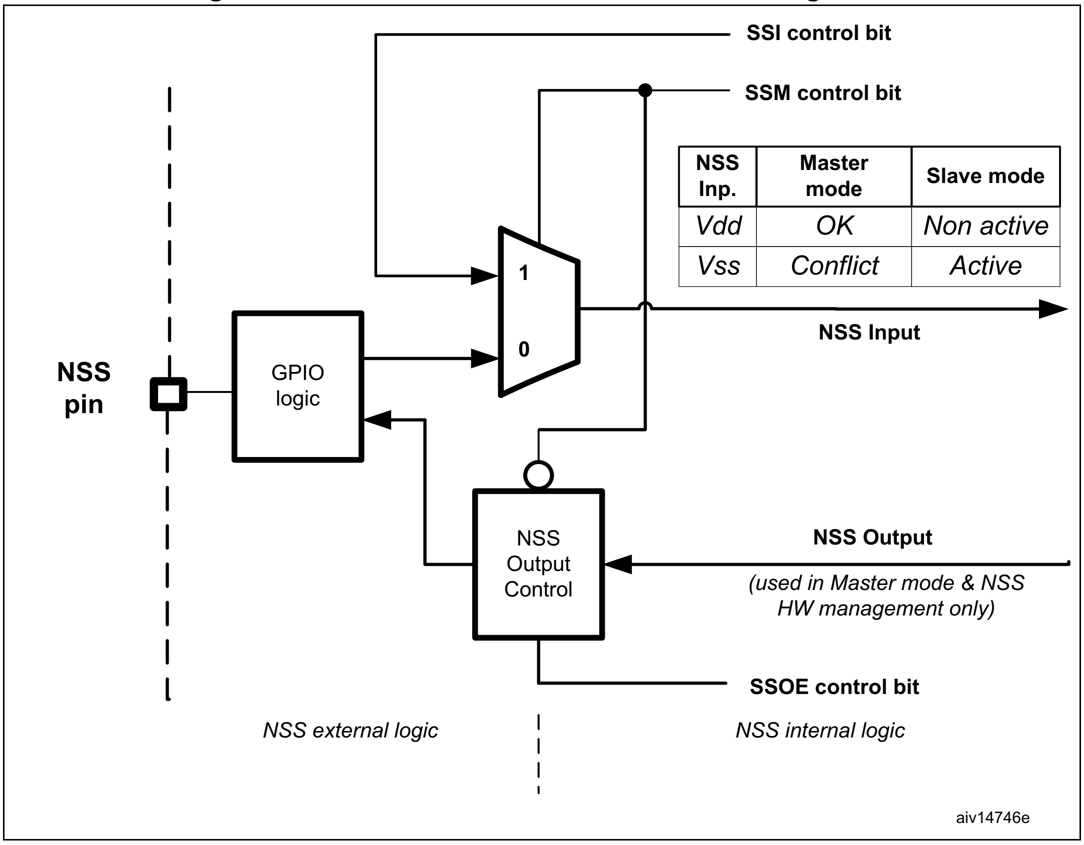
27.5.4 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

- **Software NSS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx_CR1).
  – **NSS output enable (SSM=0,SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
  – **NSS output disable (SSM=0, SSOE = 0)**: if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 235. Hardware/software slave select management



27.5.5 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

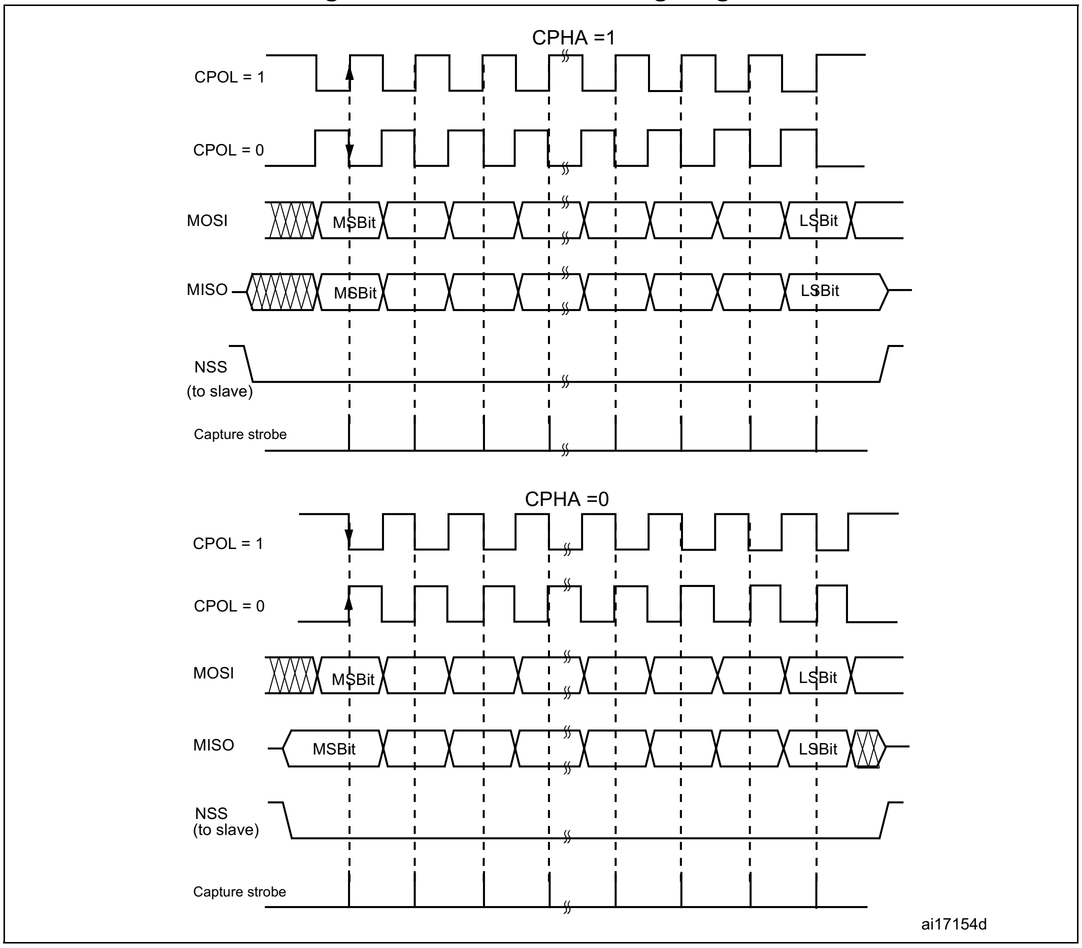
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 236, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: *Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.*

The idle state of SCK must correspond to the polarity selected in the SPIx_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

Figure 236. Data clock timing diagram

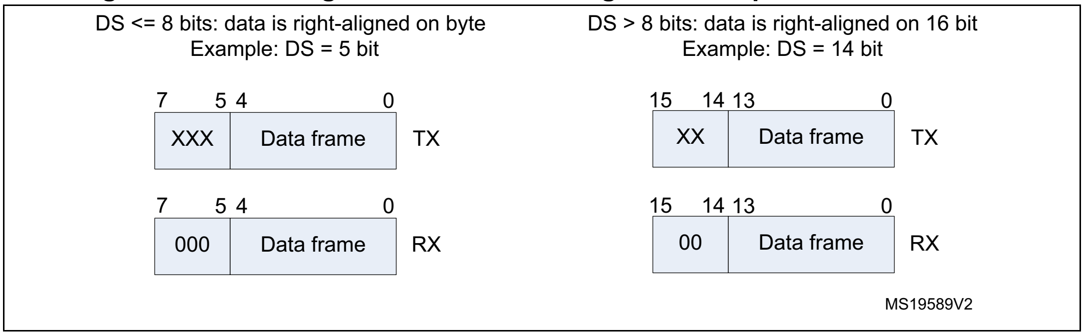


1. The order of data bits depends on LSBFIRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit. The data frame size is chosen by using the DS bits. It can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception. Whatever the selected data frame size, read access to the FIFO must be aligned with the FRXTH level. When the SPIx_DR register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte) or a half-word (see *Figure 237*). During communication, only bits within the data frame are clocked and transferred.

Figure 237. Data alignment when data length is not equal to 8-bit or 16-bit



Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 8-bit data frame size.

27.5.6 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated chapters. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI_CR1 register:
   a) Configure the serial clock baud rate using the BR[2:0] bits (Note *4*).
   b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). (Note *2*).
   c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE can't be set at the same time).
   d) Configure the LSBFIRST bit to define the frame format (Note *2*).
   e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
   f) Configure SSM and SSI (Notes *2*,*3*).
   g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI_CR2 register:
   a) Configure the DS[3:0] bits to select the data length for the transfer.
   b) Configure SSOE (*Note: 1*,*2*,*3*).
   c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
   d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CHPA and TI bits cleared in NSSP mode).
   e) Configure the FRXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx_DR register.
   f) Initialize LDMA_TX and LDMA_RX bits if DMA is used in packed mode.
4. Write to SPI_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

Note:
1. *Step is not required in slave mode.*
2. *Step is not required in TI mode.*
3. *Step is not required in NSSP mode.*
4. *The step is not required in slave mode except slave working at TI mode(1)*

27.5.7 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated chapter.

27.5.8 Data transmission and reception procedures

RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see *Section 27.5.13: CRC calculation*).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see *Section 27.5.12: TI mode*).

A read access to the SPIx_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, RXFIFO can store up to four data frames, whereas TXFIFO can only store up to three when the data frame format is not greater than 8 bits. This difference prevents possible corruption of 3x 8-bit data frames already stored in the TXFIFO when software tries to write more data in 16-bit mode into TXFIFO. Both TXE and RXNE events can be polled or handled by interrupts. See *Figure 239* through *Figure 242*.

Another way to manage the data exchange is to use DMA (see *Section 10: DMA controller (DMA)*).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at *Section 27.5.10: SPI error flags*). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see *Section 27.5.4: Slave select (NSS) pin management*).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to prevent some dummy byte exchange (refer to *Data packing* section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional "dummy" data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIiRST bits in the RCC_APBiRSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave, or
- When transactions' streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

Note: If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.

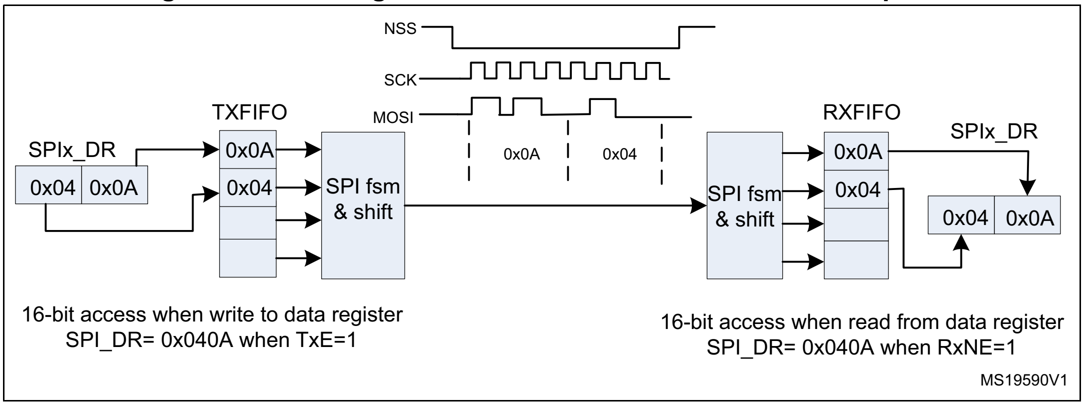
Data packing

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. *Figure 238* provides an example of data packing mode sequence handling. Two data frames are sent after the single 16-bit access the SPIx_DR register of the transmitter. This sequence can generate just one RXNE event in the receiver if the RXFIFO threshold is set to 16 bits (FRXTH=0). The receiver then has to access both data frames by a single 16-bit read of SPIx_DR as a response to this single RXNE event. The

RxFIFO threshold setting and the following read access must be always kept aligned at the receiver side, as data can be lost if it is not in line.

A specific problem appears if an odd number of such “fit into one byte” data frames must be handled. On the transmitter side, writing the last data frame of any odd sequence with an 8-bit access to SPIx_DR is enough. The receiver has to change the Rx_FIFO threshold level for the last data frame received in the odd sequence of frames in order to generate the RXNE event.

Figure 238. Packing data in FIFO for transmission and reception



Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXE or RXNE enable bit in the SPIx_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx_DR register.

See *Figure 239* through *Figure 242*.

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received is not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until FTLVL[1:0]=00 and then until BSY=0.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CR2 register, if DMA Rx is used.
2. Enable DMA streams for Tx and Rx in DMA registers, if the streams are used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CR2 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA streams for Tx and Rx in the DMA registers, if the streams are used.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CR2 register, if DMA Tx and/or DMA Rx are used.

Packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPIx_CR2 register) packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel. If the DMA channel PSIZE value is equal to 16-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. The DMA then automatically manages the write operations to the SPIx_DR register.

If data packing mode is used and the number of data to transfer is not a multiple of two, the LDMA_TX/LDMA_RX bits must be set. The SPI then considers only one data for the transmission or reception to serve the last DMA transfer (for more details refer to *Data packing*.)

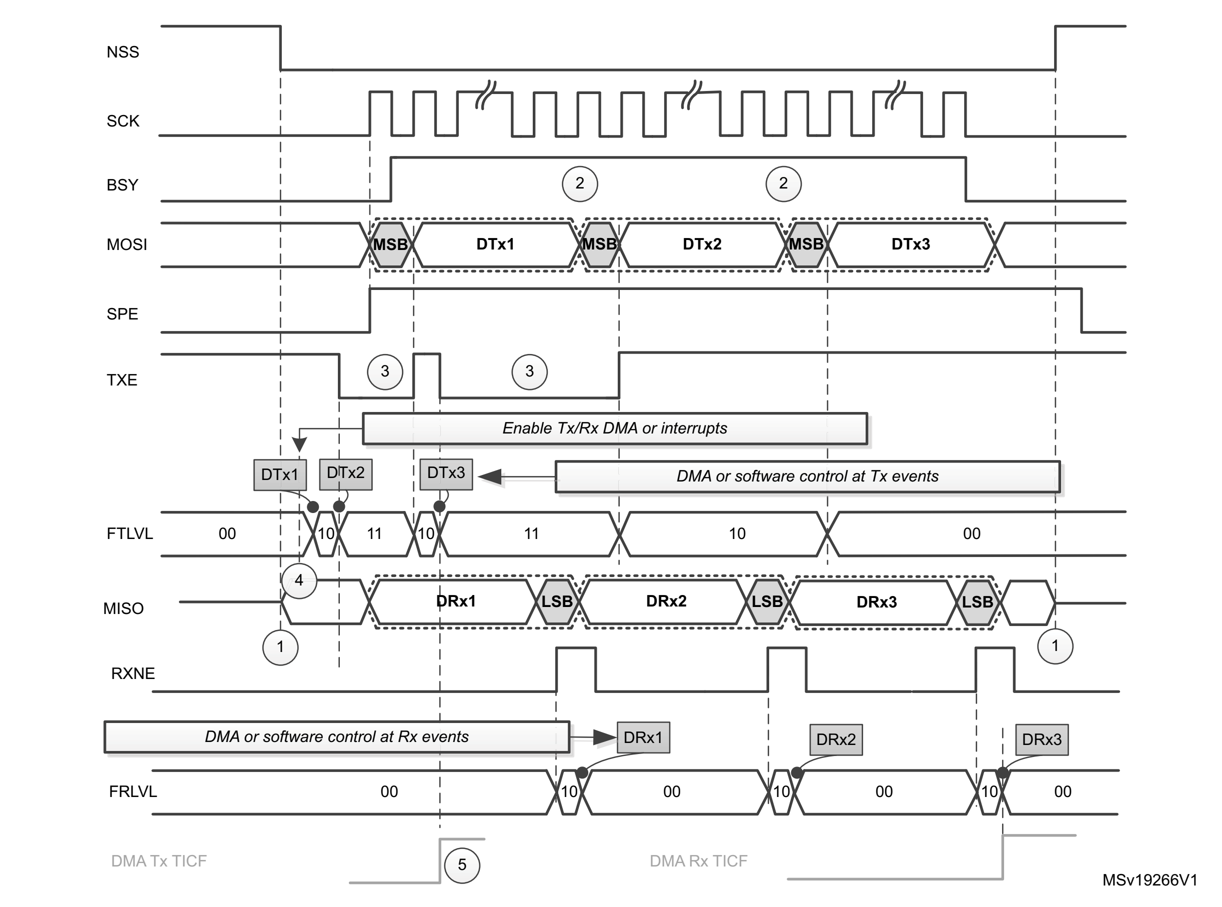
Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by pulling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for *Figure 239* through *Figure 242*.

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.
   At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depends on GPIO setting exclusively.
2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.
3. The TXE signal is cleared only if TXFIFO is full.
4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.
5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.
6. The CRC value for a package is calculated continuously frame by frame in the SPIx_TxCRCR and SPIx_RxCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).
   While the CRC value calculated in SPIx_TxCRCR is simply sent out by transmitter, received CRC information is loaded into RxFIFO and then compared with the SPIx_RxCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of RxFIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).
7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the TxFIFO is ¾ full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the TxFIFO becomes ½ full. This frame is stored into TxFIFO withan 8-bit access either by software or automatically by DMA when LDMA_TX control is set.
8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA_RX is set.

Figure 239. Master full duplex communication



Assumptions for master full duplex communication example:

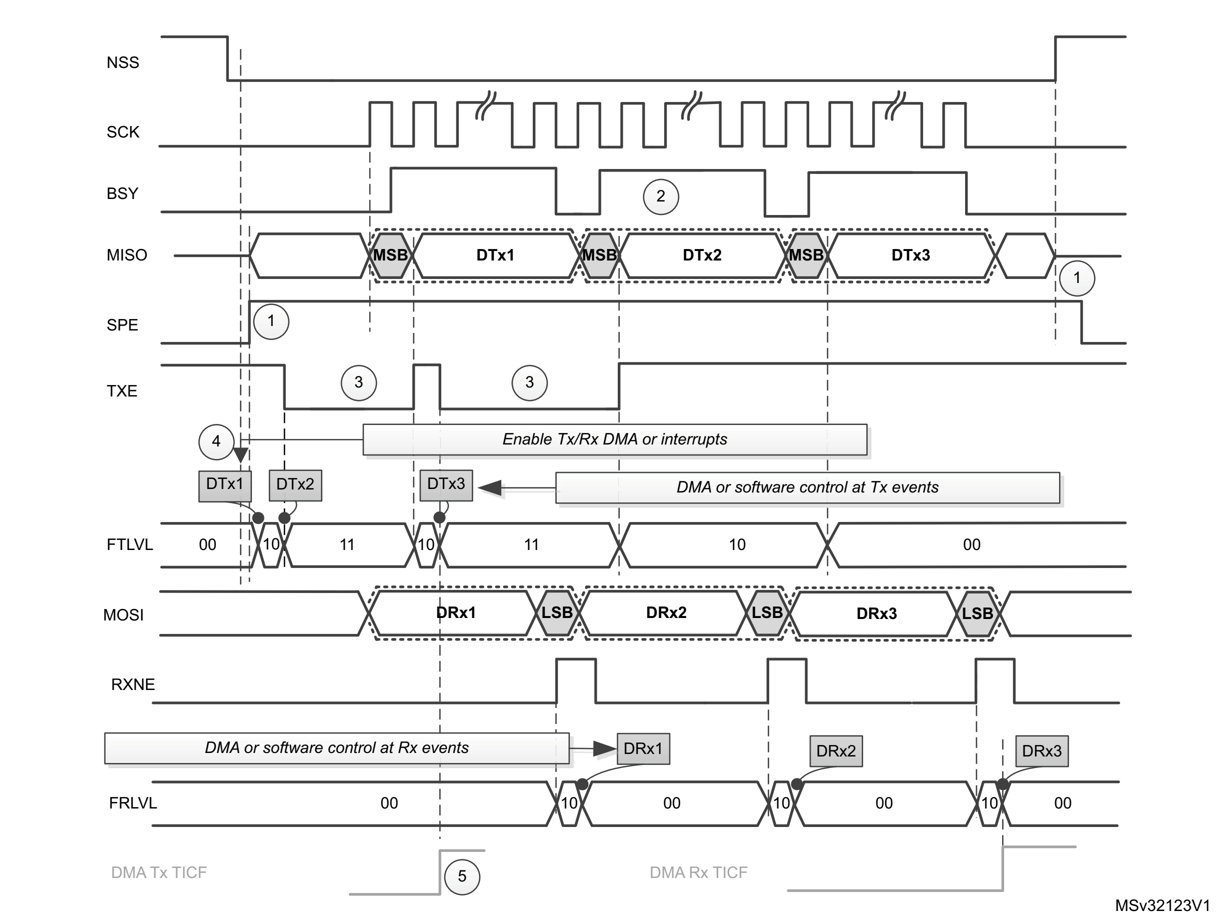
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *Communication diagrams* for details about common assumptions and notes.

Figure 240. Slave full duplex communication



Assumptions for slave full duplex communication example:

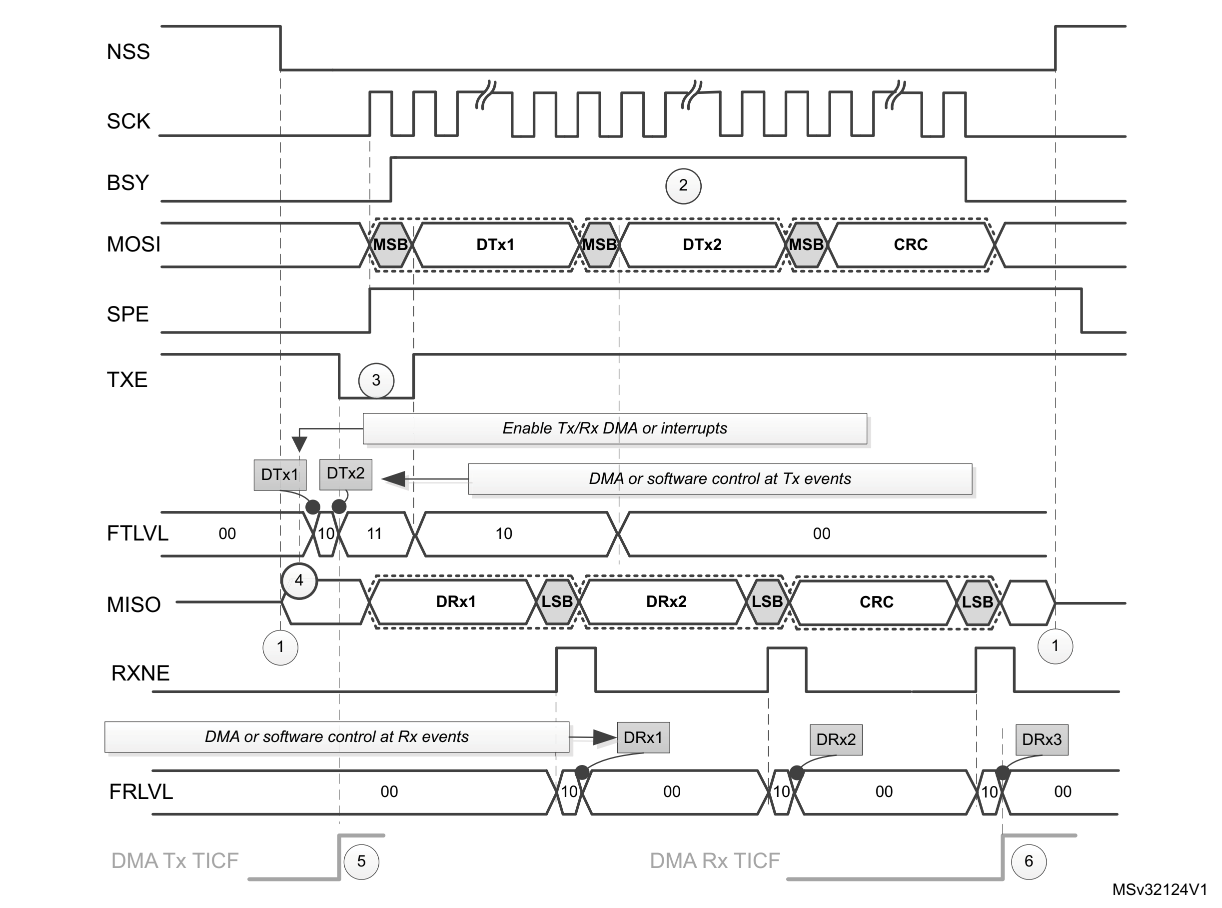
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *Communication diagrams* for details about common assumptions and notes.

Figure 241. Master full duplex communication with CRC



Assumptions for master full duplex communication with CRC example:

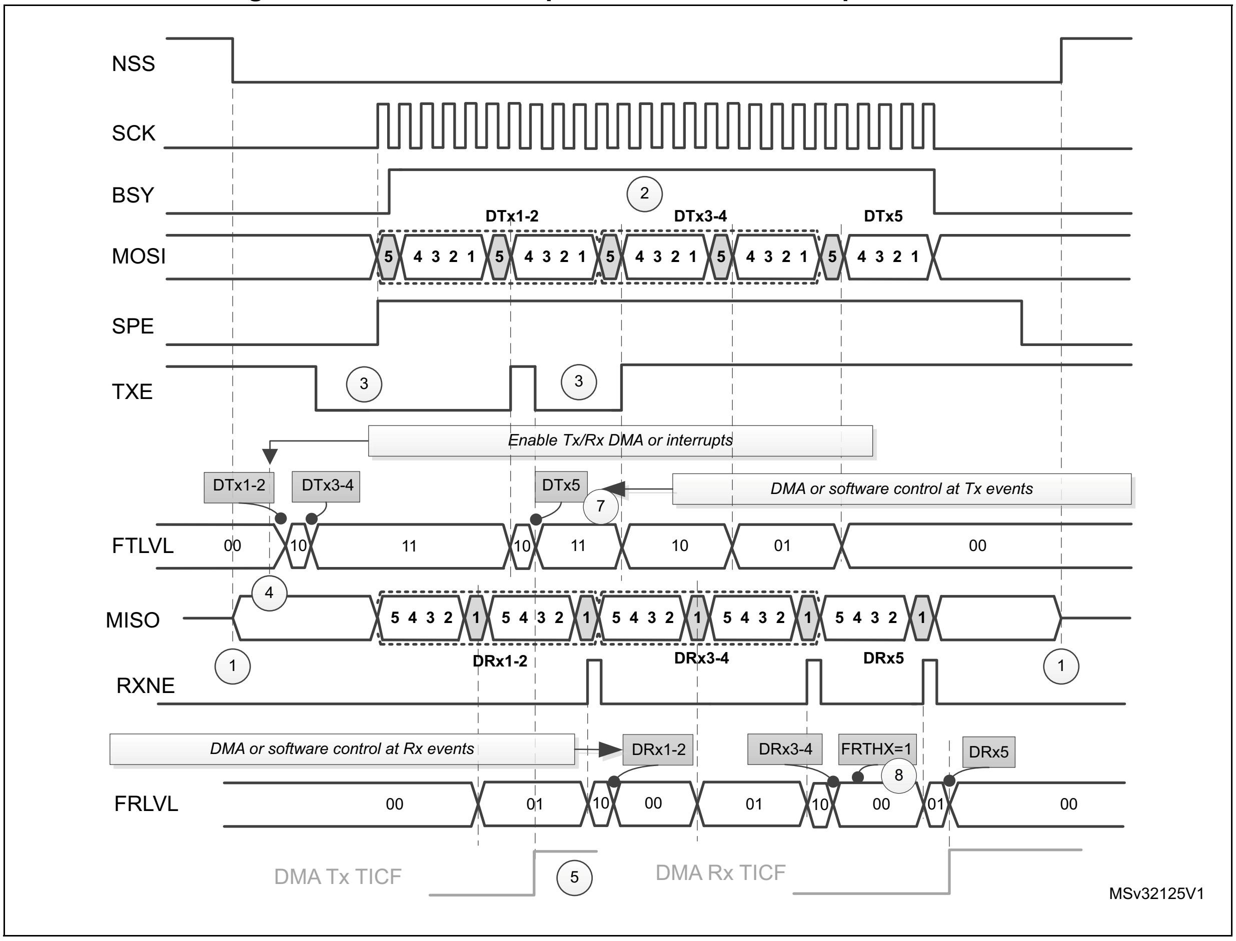
- Data size = 16 bit
- CRC enabled

If DMA is used:

- Number of Tx frames transacted by DMA is set to 2
- Number of Rx frames transacted by DMA is set to 3

See also *Communication diagrams* for details about common assumptions and notes.

Figure 242. Master full duplex communication in packed mode



Assumptions for master full duplex communication in packed mode example:

- Data size = 5 bit
- Read/write FIFO is performed mostly by 16-bit access
- FRXTH=0

If DMA is used:

- Number of Tx frames to be transacted by DMA is set to 3
- Number of Rx frames to be transacted by DMA is set to 3
- PSIZE for both Tx and Rx DMA channel is set to 16-bit
- LDMA_TX=1 and LDMA_RX=1

See also *Communication diagrams* for details about common assumptions and notes.

27.5.9 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx_CR2 register:

- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx_CR2 register is set.

The RXNE is cleared by hardware automatically when the above conditions are no longer true.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note: When the next transmission can be handled immediately by the master (e.g. if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.

27.5.10 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

Overrun flag (OVR)

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited e.g. the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see *Section 27.5.13: CRC calculation*).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI_DR register followed by a read access to the SPI_SR register.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx_SR register while the MODF bit is set.
2. Then write to the SPIx_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx_CR1 register is set. The CRCERR flag in the SPIx_SR register is set if the value received in the shift register does not match the receiver SPIx_RXCRCR value. The flag is cleared by the software.

TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

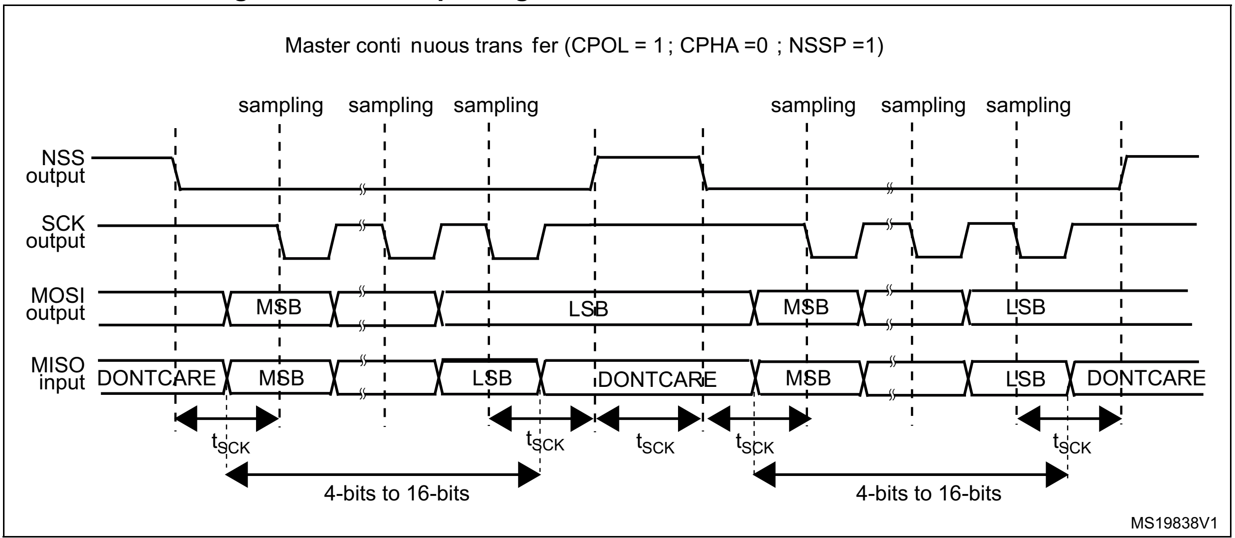
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

27.5.11 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

Figure 243 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 243. NSSP pulse generation in Motorola SPI master mode



Note: Similar behavior is encountered when CPOL = 0. In this case the sampling edge is the *rising* edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

27.5.12 TI mode

TI protocol in master mode

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx_CR1 and SPIx_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see *Figure 244*). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ ($t_{release}$) depends on internal resynchronization and on the baud rate value set in through the BR[2:0] bits in the SPIx_CR1 register. It is given by the formula:

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

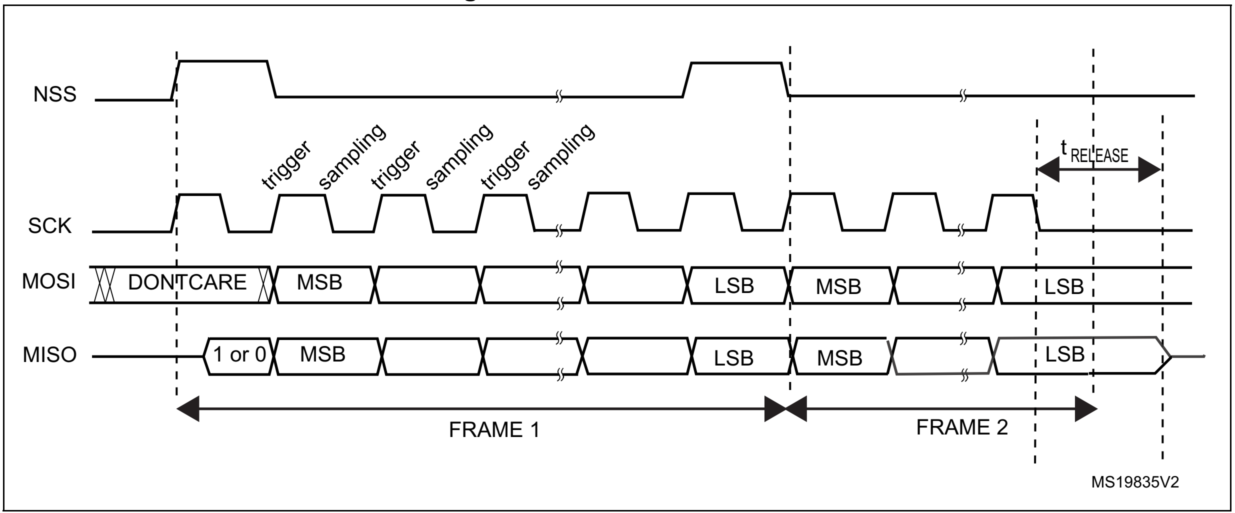
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

If the data size is equal to 4-bits or 5-bits, the master in full-duplex mode or transmit-only mode uses a protocol with one more dummy data bit added after LSB. TI NSS pulse is generated above this dummy bit clock cycle instead of the LSB in each period.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

Figure 244: TI mode transfer shows the SPI communication waveforms when TI mode is selected.

Figure 244. TI mode transfer



27.5.13 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

Note: The polynomial value should only be odd. No even values are supported.

CRC transfer managed by CPU

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx_DR register. Then CRCNEXT bit has to be set in the SPIx_CR1 register to indicate that the CRC frame transaction follows after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPIx_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx_DR register in order to clear the RXNE flag.

CRC transfer managed by DMA

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

DMA_RX = Numb_of_data + 2

In receive only mode, the DMA reception channel counter should contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA_RX bit needs managing if the number of data is odd.

Resetting the SPIx_TXCRC and SPIx_RXCRC values

The SPIx_TXCRC and SPIx_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).

If the SPI is disabled during a communication the following sequence must be followed:

1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

Note: When the SPI is in slave mode, the CRC calculator is sensitive to the SCK slave input clock as soon as the CRCEN bit is set, and this is the case whatever the value of the SPE bit. In order to avoid any wrong CRC calculation, the software must enable CRC calculation only when the clock is stable (in steady state). When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low between the data phase and the CRC phase.

27.6 SPI interrupts

During SPI communication an interrupts can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error

Interrupts can be enabled and disabled separately.

Table 96. SPI interrupt requests

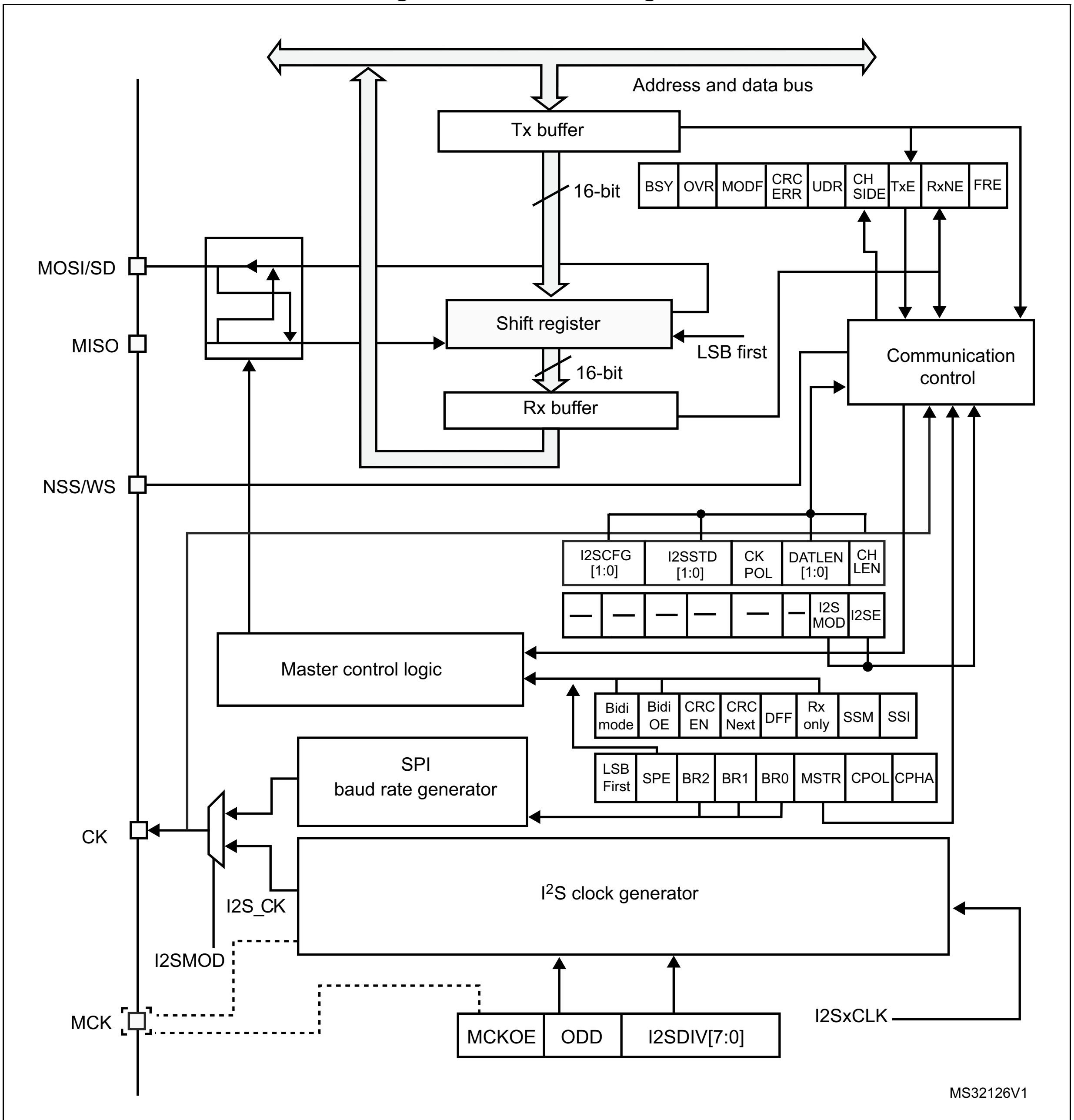
Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	

27.7 I²S functional description

27.7.1 I²S general description

The block diagram of the I²S is shown in *Figure 245*.

Figure 245. I²S block diagram



The SPI can function as an audio I²S interface when the I²S capability is enabled (by setting the I2SMOD bit in the SPIx_I2SCFGR register). This interface mainly uses the same pins, flags and interrupts as the SPI.

The I^2S shares three common pins with the SPI:

- SD: Serial Data (mapped on the MOSI pin) to transmit or receive the two time-multiplexed data channels (in half-duplex mode only).
- WS: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I^2S is configured in master mode (and when the MCKOE bit in the SPIx_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to 256 × f_S, where f_S is the audio sampling frequency.

The I^2S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I^2S mode. One is linked to the clock generator configuration SPIx_I2SPR and the other one is a generic I^2S configuration register SPIx_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPIx_CR1 register and all CRC registers are not used in the I^2S mode. Likewise, the SSOE bit in the SPIx_CR2 register and the MODF and CRCERR bits in the SPIx_SR are not used.

The I^2S uses the same SPI register for data transfer (SPIx_DR) in 16-bit wide mode.

27.7.2 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for transmission or reception. So, it is up to the software to write into the data register the appropriate value corresponding to each channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPIx_SR register. Channel left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol).

Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in a 16-bit frame
- 16-bit data packed in a 32-bit frame
- 24-bit data packed in a 32-bit frame
- 32-bit data packed in a 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPIx_DR register or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 nonsignificant bits are extended to 32 bits with 0-bits (by hardware).

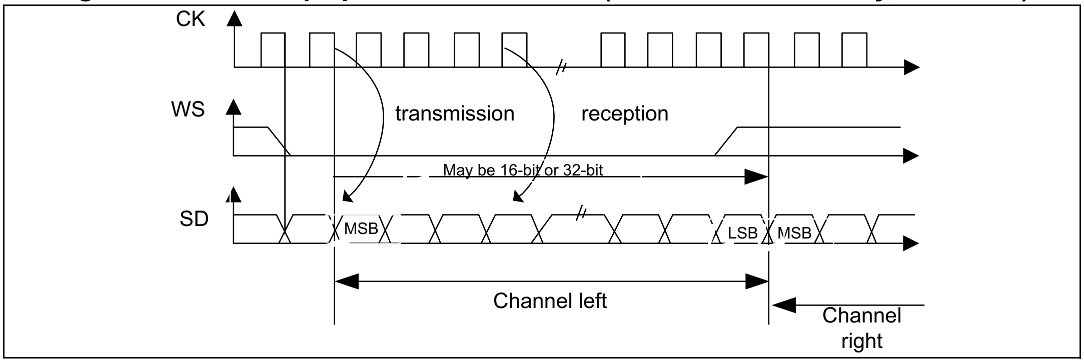
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I²S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx_I2SCFGR register.

I²S Philips standard

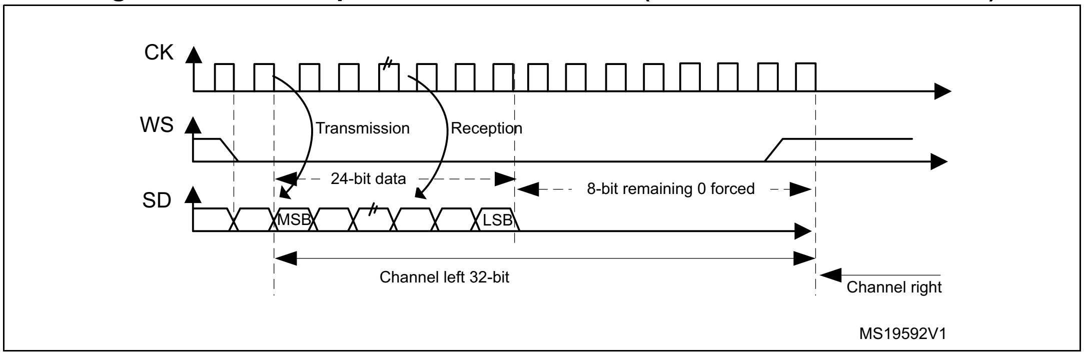
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

Figure 246. I²S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0)



Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

Figure 247. I²S Philips standard waveforms (24-bit frame with CPOL = 0)



This mode needs two write or read operations to/from the SPIx_DR register.

- In transmission mode:
  If 0x8EAA33 has to be sent (24-bit):

Figure 248. Transmitting 0x8EAA33

- In reception mode:
  If data 0x8EAA33 is received:

Figure 249. Receiving 0x8EAA33

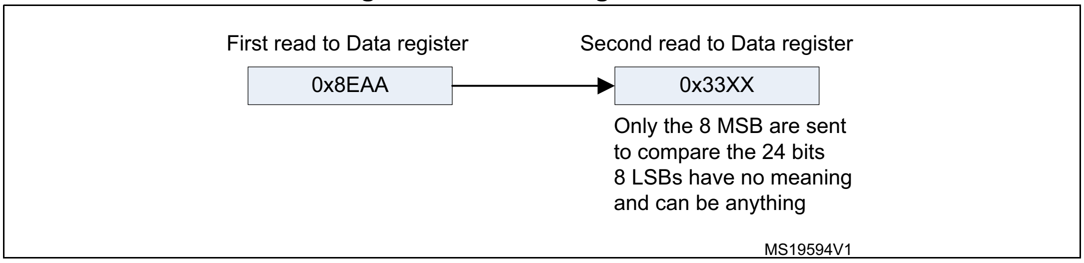
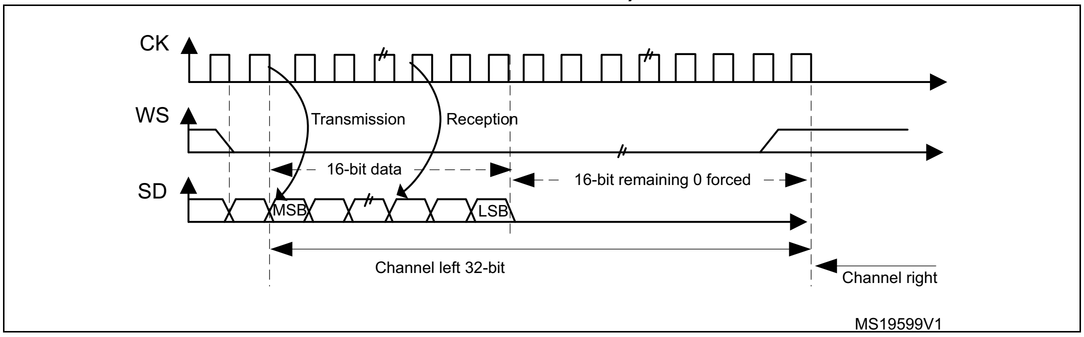


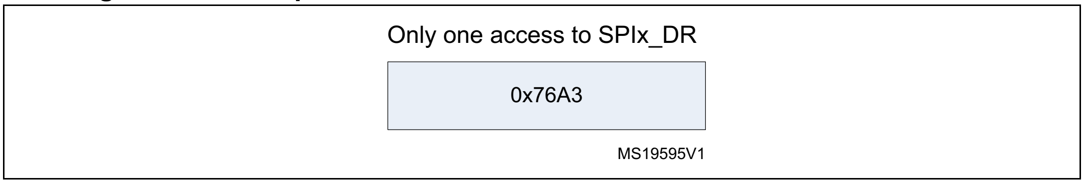
Figure 250. I²S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0)



When 16-bit data frame extended to 32-bit channel frame is selected during the I²S configuration phase, only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in *Figure 251* is required.

Figure 251. Example of 16-bit data frame extended to 32-bit channel frame



For transmission, each time an MSB is written to SPIx_DR, the TXE flag is set and its interrupt, if allowed, is generated to load the SPIx_DR register with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

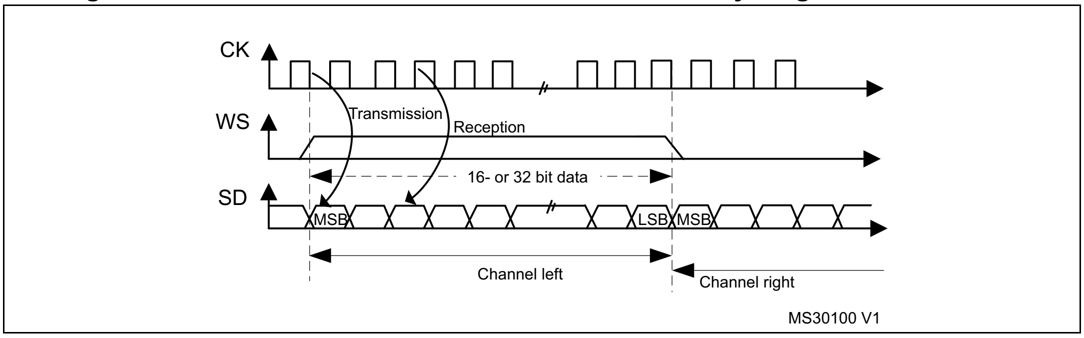
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

MSB justified standard

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

Figure 252. MSB Justified 16-bit or 32-bit full-accuracy length with CPOL = 0



Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

Figure 253. MSB justified 24-bit frame length with CPOL = 0

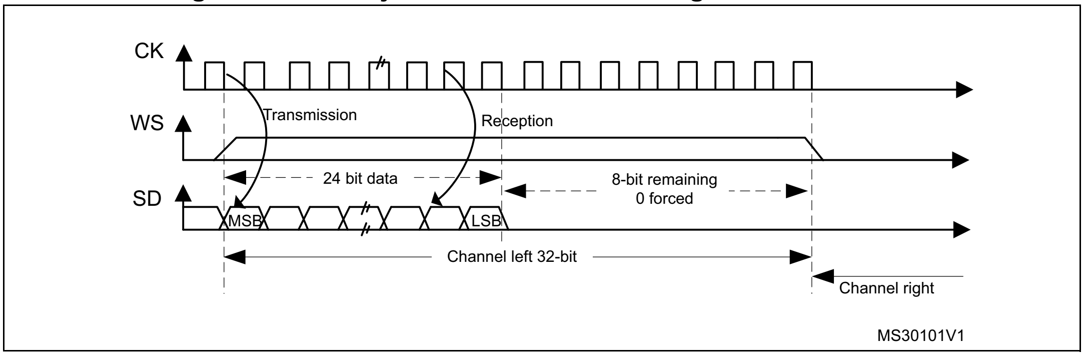
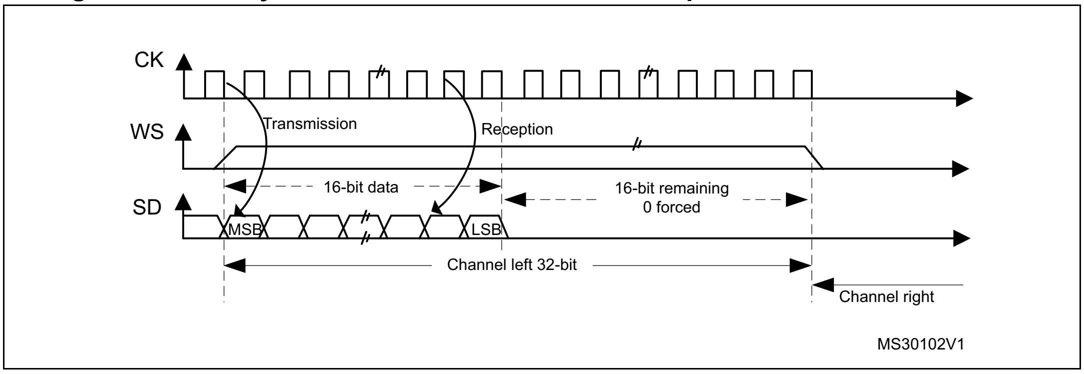


Figure 254. MSB justified 16-bit extended to 32-bit packet frame with CPOL = 0



LSB justified standard

This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

Figure 255. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0

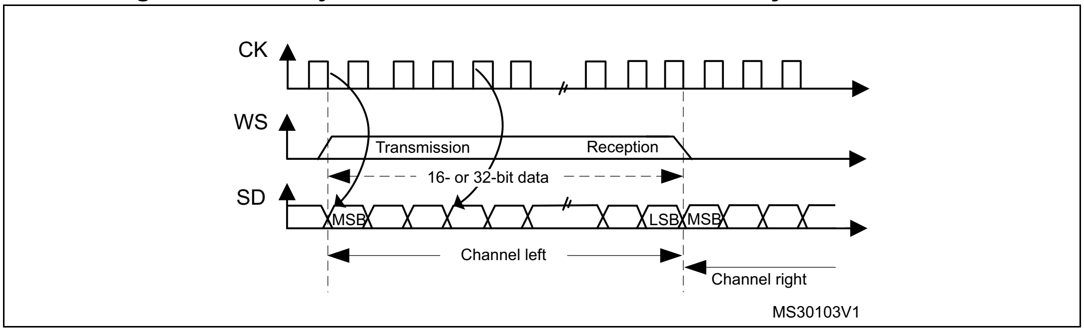
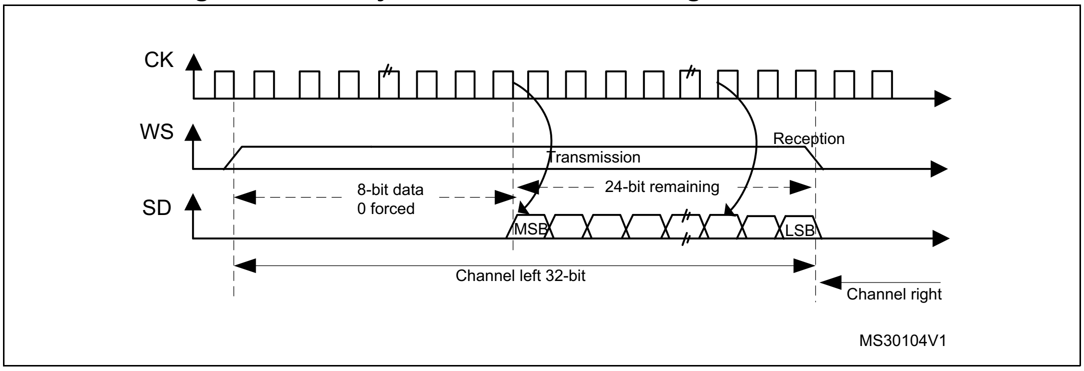


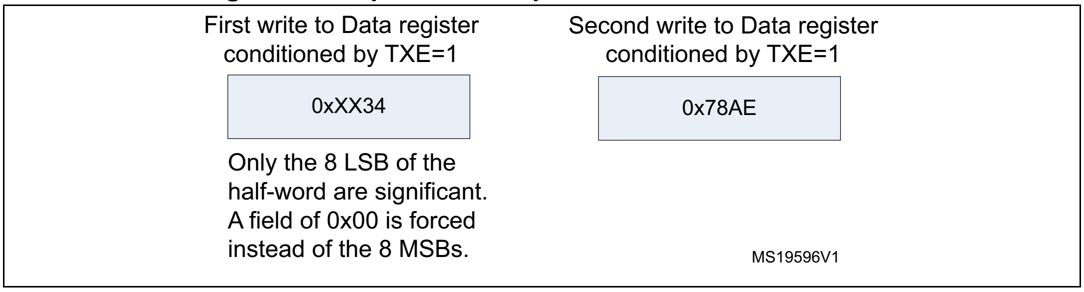
Figure 256. LSB justified 24-bit frame length with CPOL = 0



- In transmission mode:

  If data 0x3478AE have to be transmitted, two write operations to the SPIx_DR register are required by software or by DMA. The operations are shown below.

Figure 257. Operations required to transmit 0x3478AE



- In reception mode:

  If data 0x3478AE are received, two successive read operations from the SPIx_DR register are required on each RXNE event.

Figure 258. Operations required to receive 0x3478AE

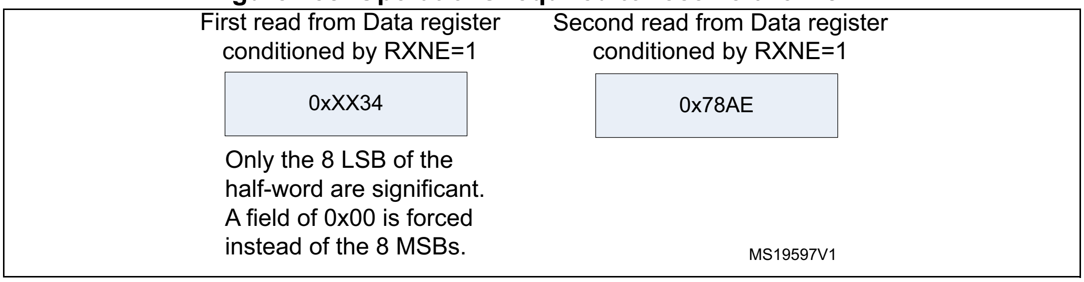
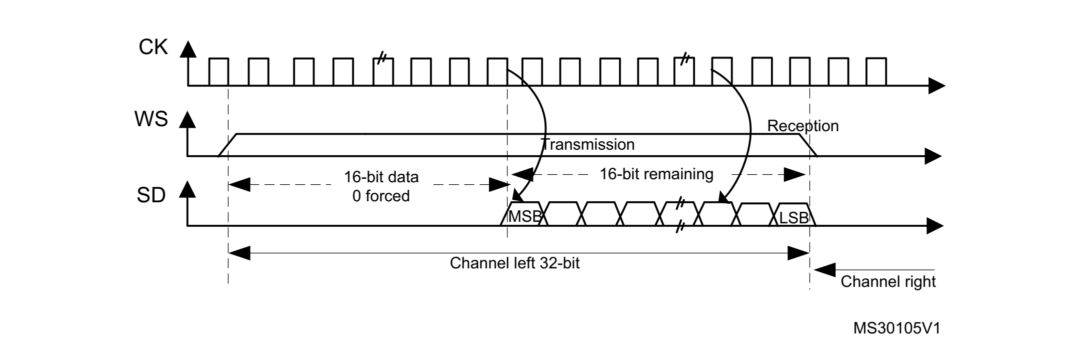


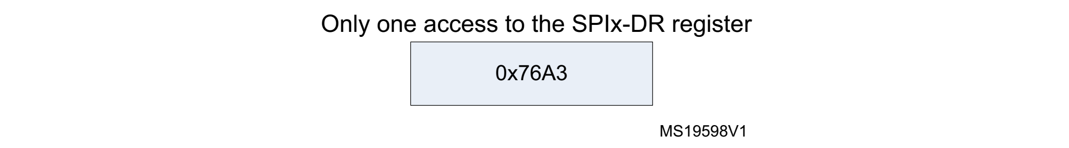
Figure 259. LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0



When 16-bit data frame extended to 32-bit channel frame is selected during the I^2S configuration phase, Only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in *Figure 260* is required.

Figure 260. Example of 16-bit data frame extended to 32-bit channel frame



In transmission mode, when a TXE event occurs, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). The TXE flag is set again as soon as the effective data (0x76A3) is sent on SD.

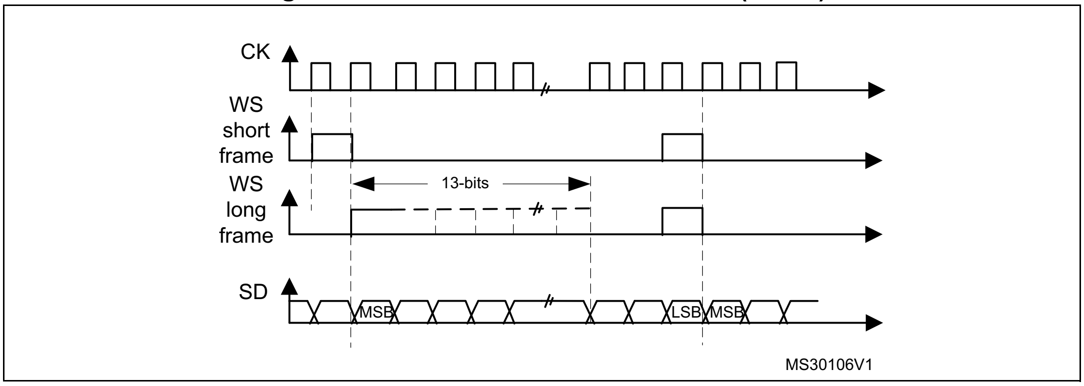
In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPIx_I2SCFGR register.

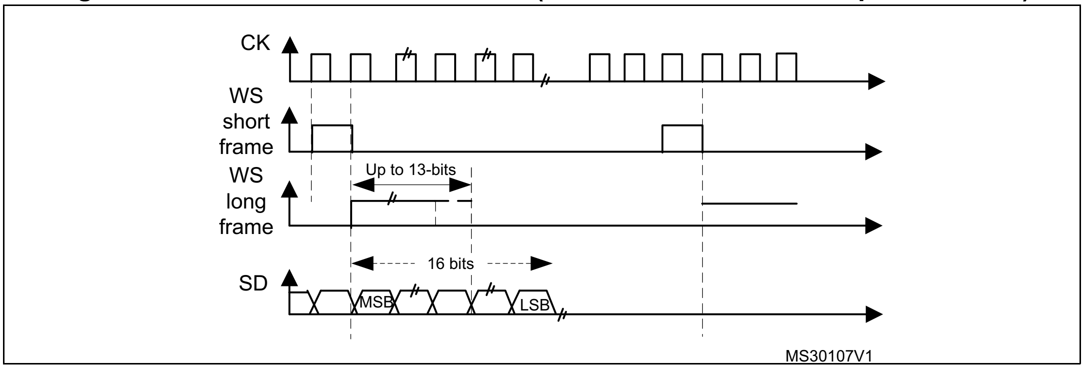
Figure 261. PCM standard waveforms (16-bit)



For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

For short frame synchronization, the WS synchronization signal is only one cycle long.

Figure 262. PCM standard waveforms (16-bit extended to 32-bit packet frame)



Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPIx_I2SCFGR register) even in slave mode.

27.7.3 Clock generator

The I^2S bitrate determines the dataflow on the I^2S data line and the I^2S clock signal frequency.

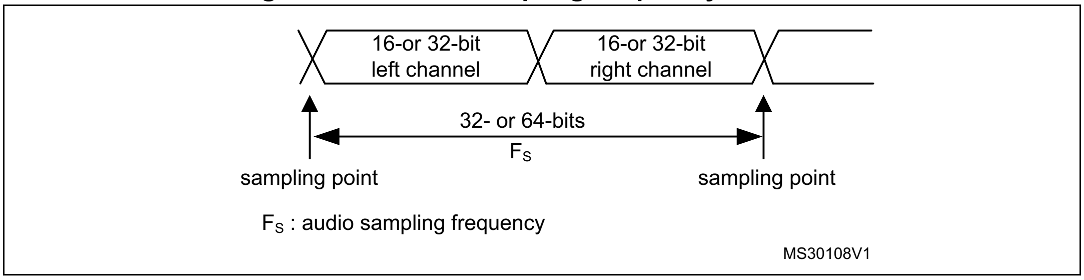
I^2S bitrate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I^2S bitrate is calculated as follows:

I^2S bitrate = $16 \times 2 \times f_S$

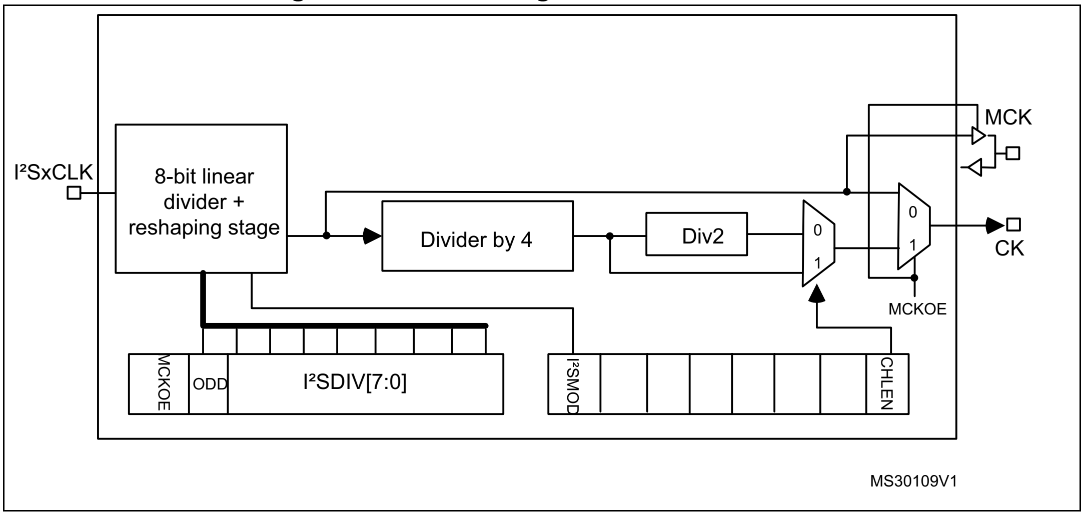
It is: I^2S bitrate = $32 \times 2 \times f_S$ if the packet length is 32-bit wide.

Figure 263. Audio sampling frequency definition



When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency.

Figure 264. I²S clock generator architecture



1. Where x can be 2 or 3.

Figure 264 presents the communication clock architecture. The I2Sx clock is always a 32 MHz frequency clock.

Warning: In addition, it is mandatory to keep I2SxCLK frequency higher or equal to the APB clock used by the SPI/I2S block. If this condition is not respected, SPI/I2S do not work.

The audio sampling frequency may be 192 kHz, 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range). In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

When the master clock is generated (MCKOE in the SPIx_I2SPR register is set):

f_S = I2SxCLK / [256 * ((2*I2SDIV)+ODD)] whatever the channel frame width (16-bit wide or 32-bit wide).

When the master clock is disabled (MCKOE bit cleared):

f_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD))] when the channel frame is 16-bit wide

f_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD))] when the channel frame is 32-bit wide

Table 97 provides example precision values for different clock configurations.

Note: Other configurations are possible that allow optimum clock precision.

Table 97. Audio frequency precision using I2SCLK = 64 MHz (continued)

I2SCLK (MHz)	Data length	I2SDIV	I2SODD	MCK	Target fs (Hz)	Real fs (KHz)	Error
64	16	5	0	No	192000	200000	4.1667%
64	32	2	1	No	192000	200000	4.1667%
64	16	10	1	No	96000	95238.1	0.7936%
64	32	5	0	No	96000	100000	4.167%
64	16	22	1	No	44100	44444.44	0.7811%
64	32	11	1	No	44100	43478.26	1.4098%
64	16	31	1	No	32000	31746.03	0.7937%
64	32	15	1	No	32000	32258.06	0.8065%
64	16	45	1	No	22050	21978.02	0.3264%
64	32	22	1	No	22050	22222.22	0.7811%
64	16	62	1	No	16000	160000	0%
64	32	31	1	No	16000	15873.032	0.7937%
64	16	90	1	No	11025	11049.72	0.2243%
64	32	45	1	No	11025	10989.01	0.3264%
64	16	125	0	No	8000	8000	0%
64	32	62	1	No	8000	8000	0%
64	16	2	1	Yes	48000	50000	4.1667%
64	32	2	1	Yes	48000	50000	4.1667%
64	16	3	0	Yes	44100	41666.667	5.5178%
64	32	3	0	Yes	44100	41666.667	5.5178%
64	16	4	0	Yes	32000	31250	2.3438%
64	32	4	0	Yes	32000	31250	2.3438%
64	16	5	1	Yes	22050	22727.27	3.0175%
64	32	5	1	Yes	22050	22727.27	3.0175%
64	16	8	0	Yes	16000	15625	2.3438%
64	32	8	0	Yes	16000	15625	2.3438%
64	16	11	1	Yes	11025	10869.57	1.4098%
64	32	11	1	Yes	11025	10869.57	1.4098%
64	16	15	1	Yes	8000	8064.516	0.8065%
64	32	15	1	Yes	8000	8064.516	0.8065%

Table 98. Audio frequency precision using I2SCLK = 32 MHz

I2SCLK (MHz)	Data length	I2SDIV	I2SODD	MCK	Target fs (Hz)	Real fs (KHz)	Error
32	16	5	0	No	96000	100000	4.1667%
32	32	2	1	No	96000	100000	4.1667%
32	16	10	1	No	48000	47619.0476	0.7936%
32	32	5	0	No	48000	50000	4.167%
32	16	11	1	No	44100	43478.261	1.410%
32	32	8	1	No	44100	45454.545	3.0715%
32	16	15	1	No	32000	32258.0645	0.806%
32	32	8	0	No	32000	31250	2.344%
32	16	22	1	No	22050	22222.22	0.781%
32	32	11	1	No	22050	21739.1304	1.410%
32	16	31	1	No	16000	15873.0159	0.794%
32	32	15	1	No	16000	16129.032	0.806%
32	16	45	1	No	11025	10989.011	0.326%
32	32	22	1	No	11025	11111.111	0.781%
32	16	62	1	No	8000	8000	0%
32	32	47	0	No	8000	7936.508	0.794%
32	16	1	1	Yes	48000	41666.667	13.194%
32	32	1	1	Yes	48000	41666.667	13.194%
32	16	1	1	Yes	44100	41666.667	5.518%
32	32	1	1	Yes	44100	41666.667	5.518%
32	16	2	0	Yes	32000	31250	2.3438%
32	32	2	0	Yes	32000	31250	2.3438%
32	16	3	0	Yes	22050	20833.333	5.5178%
32	32	3	0	Yes	22050	20833.333	5.5178%
32	16	4	0	Yes	16000	15625	2.3438%
32	32	4	0	Yes	16000	15625	2.3438%
32	16	5	1	Yes	11025	11363.6364	3.0715%
32	32	5	1	Yes	11025	11363.6364	3.0715%
32	16	8	0	Yes	8000	7812.5	2.344%
32	32	8	0	Yes	8000	7812.5	2.344%

Table 99. Audio frequency precision using I2SCLK = 16 MHz

I2SCLK (MHz)	Data length	I2SDIV	I2SODD	MCK	Target fs (Hz)	Real fs (KHz)	Error
16	16	2	1	No	96000	100000	4.1667%
16	32	2	0	No	96000	62500	34.890%
16	16	4	1	No	48000	50000	4.167%
16	32	2	1	No	48000	50000	4.167%
16	16	5	1	No	44100	45454.545	3.0715%
16	32	3	0	No	44100	41666.67	5.518%
16	16	8	0	No	32000	31250	2.344%
16	32	4	0	No	32000	31250	2.344%
16	16	11	1	No	22050	21739.13	1.410%
16	32	5	1	No	22050	22727.27	3.071%
16	16	15	1	No	16000	16129.032	0.806%
16	32	15	1	No	16000	15625	2.344%
16	16	22	1	No	11025	11111.111	0.781%
16	32	11	1	No	11025	10869.57	1.409%
16	16	31	1	No	8000	7936.51	0.794%
16	32	15	1	No	8000	8064.52	0.806%
16	16	N/A	N/A	Yes	48000	N/A	-
16	32	N/A	N/A	Yes	48000	N/A	-
16	16	N/A	N/A	Yes	44100	N/A	-
16	32	N/A	N/A	Yes	44100	N/A	-
16	16	N/A	N/A	Yes	32000	N/A	-
16	32	N/A	N/A	Yes	32000	N/A	-
16	16	3	0	Yes	22050	N/A	-
16	32	3	0	Yes	22050	N/A	-
16	16	2	0	Yes	16000	15625	2.3438%
16	32	2	0	Yes	16000	15625	2.3438%
16	16	3	0	Yes	11025	10416.67	5.518%
16	32	3	0	Yes	11025	10416.67	5.518%
16	16	4	0	Yes	8000	7812.5	2.344%
16	32	4	0	Yes	8000	7812.5	2.344%

27.7.4 I²S master mode

The I²S can be configured in master mode. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, controlled by the MCKOE bit in the SPIx_I2SPR register.

Procedure

1. Select the I2SDIV[7:0] bits in the SPIx_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPIx_I2SPR register also has to be defined.
2. Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPIx_I2SPR register if the master clock MCK needs to be provided to the external DAC (STM32WL33xx only)/ADC (STM32WL33xx/31xx only) audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to *Section 27.7.3: Clock generator*).
3. Set the I2SMOD bit in the SPIx_I2SCFGR register to activate the I²S functions and choose the I²S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the CHLEN bit. Select also the I²S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.
4. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
5. The I2SE bit in SPIx_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPIx_I2SPR is set.

Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Lets assume the first data written into the Tx buffer corresponds to the left channel data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the right channel have to be written into the Tx buffer. The CHSIDE flag indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high.

A full frame has to be considered as a left channel data transmission followed by a right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

For more details about the write operations depending on the I²S standard mode selected, refer to *Section 27.7.2: Supported audio protocols*).

To ensure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission.

To switch off the I²S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for transmission mode except for the point 3 (refer to the procedure described in *Section 27.7.4: I²S master mode*), where the configuration should set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPIx_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I²S cell.

For more details about the read operations depending on the I²S standard mode selected, refer to *Section 27.7.2: Supported audio protocols*.

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I²S, specific actions are required to ensure that the I²S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB justified mode (I2SSTD = 10)
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait 17 I²S clock cycles (using a software loop)
  c) Disable the I²S (I2SE = 0)
- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, I²S or PCM modes (I2SSTD = 00, I2SSTD = 01 or I2SSTD = 11, respectively)
  a) Wait for the last RXNE
  b) Then wait 1 I²S clock cycle (using a software loop)
  c) Disable the I²S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I²S:
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait one I²S clock cycle (using a software loop)
  c) Disable the I²S (I2SE = 0)

Note: The BSY flag is kept low during transfers.

27.7.5 I²S slave mode

For the slave configuration, the I²S can be configured in transmission or reception mode. The operating mode is following mainly the same rules as described for the I²S master configuration. In slave mode, there is no clock to be generated by the I²S interface. The clock and WS signals are input from the external master connected to the I^2S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPIx_I2SCFGR register to select I^2S mode and choose the I^2S standard through the I2SSTD[1:0] bits, the data length through the DATLEN[1:0] bits and the number of bits per channel for the frame configuring the CHLEN bit. Select also the mode (transmission or reception) for the slave through the I2SCFG[1:0] bits in SPIx_I2SCFGR register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
3. The I2SE bit in SPIx_I2SCFGR register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I^2S data register has to be loaded before the master initiates the communication.

For the I^2S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I^2S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

Note: The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

Note that the TXE flag should be checked to be at 1 before attempting to write the Tx buffer.

For more details about the write operations depending on the I^2S standard mode selected, refer to *Section 27.7.2: Supported audio protocols*.

To secure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPIx_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPIx_CR2 register, an interrupt is generated when the UDR flag in the SPIx_SR register goes high. In this case, it is mandatory to switch off the I^2S and to restart a data transfer starting from the left channel.

To switch off the I^2S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in *Section 27.7.5: I²S slave mode*), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.

Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPIx_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPIx_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer.

The CHSIDE flag is updated each time data are received to be read from the SPIx_DR register. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

For more details about the read operations depending the I²S standard mode selected, refer to *Section 27.7.2: Supported audio protocols*.

If data are received while the preceding received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I²S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

Note: The external master components should have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

27.7.6 I²S error flags

There are three error flags for the I²S cell.

Underrun flag (UDR)

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPIx_DR. It is available when the I2SMOD bit in the SPIx_I2SCFGR register is set. An interrupt may be generated if the ERRIE bit in the SPIx_CR2 register is set.
The UDR bit is cleared by a read operation on the SPIx_SR register.

Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from the SPIx_DR register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx_CR2 register.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPIx_DR register followed by a read access to the SPIx_SR register.

Frame error flag (FRE)

This flag can be set by hardware only if the I^2S is configured in Slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the synchronization is lost, the following steps are required to recover from this state and resynchronize the external master device with the I^2S slave device:

1. Disable the I^2S.
2. Enable it again when the correct level is detected on the WS line (WS line is high in I^2S mode or low for MSB- or LSB-justified or PCM modes.

Desynchronization between master and slave devices may be due to noisy environment on the SCK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by software when the status register is read.

27.7.7 DMA features

In I^2S mode, the DMA works in exactly the same way as it does in SPI mode. There is no difference except that the CRC feature is not available in I^2S mode since there is no data transfer protection system.

27.8 I^2S interrupts

Table 100 provides the list of I^2S interrupts.

Table 100. I^2S interrupt requests

Interrupt event	Event flag	Enable control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Overrun error	OVR	ERRIE
Underrun error	UDR	
Frame error flag	FRE	

27.9 SPI and I^{2}S registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI_DR in addition by can be accessed by 8-bit access.

27.9.1 SPI control register 1 (SPIx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE:** Bidirectional data mode enable. This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

Note: This bit is not used in I^{2}S mode.

Bit 14 **BIDIOE:** Output enable in bidirectional mode

This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used. This bit is not used in I^{2}S mode.

Bit 13 **CRCEN:** Hardware CRC calculation enable

0: CRC calculation disabled

1: CRC calculation Enabled

Note: This bit should be written only when SPI is disabled (SPE = ‘0’) for correct operation. This bit is not used in I^{2}S mode.

Bit 12 **CRCNEXT:** Transmit CRC next

0: Next transmit value is from Tx buffer

1: Next transmit value is from Tx CRC register

Note: This bit has to be written as soon as the last data is written in the SPIx_DR register. This bit is not used in I^{2}S mode.

Bit 11 **CRCL:** CRC length

This bit is set and cleared by software to select the CRC length.

0: 8-bit CRC length

1: 16-bit CRC length

Note: This bit should be written only when SPI is disabled (SPE = ‘0’) for correct operation. This bit is not used in I^{2}S mode.

Bit 10 **RXONLY:** Receive only mode enabled.

This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receive only mode is active.This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

0: Full duplex (Transmit and receive)

1: Output disabled (Receive-only mode)

Note: This bit is not used in I^2S mode.

Bit 9 **SSM:** Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

0: Software slave management disabled

1: Software slave management enabled

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 8 **SSI:** Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored.

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 7 **LSBFIRST***:* Frame format

0: data is transmitted / received with the MSB first

1: data is transmitted / received with the LSB first

Note: 1. This bit should not be changed when communication is ongoing.

2. This bit is not used in I^2S mode and SPI TI mode.

Bit 6 **SPE:** SPI enable

0: Peripheral disabled

1: Peripheral enabled

Note: When disabling the SPI, follow the procedure described in Procedure for disabling the SPI.

This bit is not used in I^2S mode.

Bits 5:3 **BR[2:0]:** Baud rate control

000: $f_{PCLK}/2$

001: $f_{PCLK}/4$

010: $f_{PCLK}/8$

011: $f_{PCLK}/16$

100: $f_{PCLK}/32$

101: $f_{PCLK}/64$

110: $f_{PCLK}/128$

111: $f_{PCLK}/256$

Note: These bits should not be changed when communication is ongoing.

This bit is not used in I^2S mode.

Bit 2 **MSTR:** Master selection

0: Slave configuration

1: Master configuration

Note: This bit should not be changed when communication is ongoing.

This bit is not used in I^2S mode.

Bit1 **CPOL:** Clock polarity

0: CK to 0 when idle

1: CK to 1 when idle

Note: This bit should not be changed when communication is ongoing.

This bit is not used in I^2S mode and SPI TI mode.

Bit 0 **CPHA:** Clock phase

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

Note: This bit should not be changed when communication is ongoing.

This bit is not used in I^2S mode and SPI TI mode.

27.9.2 SPI control register 2 (SPIx_CR2)

Address offset: 0x04

Reset value: 0x0700

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LDMA_TX	LDMA_RX	FRXTH	DS [3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **LDMA_TX:** Last DMA transfer for transmission

This bit is used in data packing mode, to define if the total number of data to transmit by DMA is odd or even. It has significance only if the TXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).

0: Number of data to transfer is even

1: Number of data to transfer is odd

Note: Refer to Procedure for disabling the SPI if the CRCEN bit is set.

This bit is not used in I^2S mode.

Bit 13 **LDMA_RX**: Last DMA transfer for reception

This bit is used in data packing mode, to define if the total number of data to receive by DMA is odd or even. It has significance only if the RXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).

0: Number of data to transfer is even

1: Number of data to transfer is odd

Note: Refer to Procedure for disabling the SPI if the CRCEN bit is set.

This bit is not used in I^2S mode.

Bit 12 **FRXTH**: FIFO reception threshold

FRXTH shall be set according the read access (16-bit or 8-bit) to the FIFO.

This bit is used to set the threshold of the RXFIFO that triggers an RXNE event

0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit)

1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)

Note: This bit is not used in I²S mode.

Bit 11:8 **DS [3:0]**: Data size

These bits configure the data length for SPI transfers:

0000: Not used
0001: Not used
0010: Not used
0011: 4-bit
0100: 5-bit
0101: 6-bit
0110: 7-bit
0111: 8-bit
1000: 9-bit
1001: 10-bit
1010: 11-bit
1011: 12-bit
1100: 13-bit
1101: 14-bit
1110: 15-bit
1111: 16-bit

If software attempts to write one of the “Not used” values, they are forced to the value “0111”(8-bit).

Note: This bit is not used in I²S mode.

Bit 7 **TXEIE:** Tx buffer empty interrupt enable

0: TXE interrupt masked

1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE:** RX buffer not empty interrupt enable

0: RXNE interrupt masked

1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE:** Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode and UDR, OVR, and FRE in I^{2}S mode).

0: Error interrupt is masked

1: Error interrupt is enabled

Bit 4 **FRF**: Frame format

0: SPI Motorola mode

1 SPI TI mode

Note: This bit must be written only when the SPI is disabled (SPE=0).
This bit is not used in I^{2}S mode.

Bit 3 **NSSP**: NSS pulse management

This bit is used in master mode only. it allow the SPI to generate an NSS pulse between two consecutive data when doing continuous transfers. In the case of a single data transfer, it forces the NSS pin high level after the transfer.

It has no meaning if CPHA = ’1’, or FRF = ’1’.

0: No NSS pulse

1: NSS pulse generated

Note: 1. This bit must be written only when the SPI is disabled (SPE=0).

2. This bit is not used in I^2S mode and SPI TI mode.

Bit 2 **SSOE:** SS output enable

0: SS output is disabled in master mode and the SPI interface can work in multimaster configuration

1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multimaster environment.

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 1 **TXDMAEN:** Tx buffer DMA enable

When this bit is set, a DMA request is generated whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

Bit 0 **RXDMAEN:** Rx buffer DMA enable

When this bit is set, a DMA request is generated whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

27.9.3 SPI status register (SPIx_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTLVL[1:0]		FRLVL[2:0]		FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
			r	r	r	r	r	r	r	r	rc_w0	r	r	r	r

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:11 **FTLVL[1:0]:** FIFO Transmission Level

These bits are set and cleared by hardware.

00: FIFO empty
01: 1/4 FIFO
10: 1/2 FIFO
11: FIFO full (considered as FULL when the FIFO threshold is greater than 1/2)

Note: These bits are not used in I²S mode.

Bits 10:9 **FRLVL[1:0]**: FIFO reception level

These bits are set and cleared by hardware.

00: FIFO empty
01: 1/4 FIFO
10: 1/2 FIFO
11: FIFO full

Note: These bits are not used in I²S mode and in SPI receive-only mode while CRC calculation is enabled.

Bits 8 **FRE**: Frame format error

This flag is used for SPI in TI slave mode and I^2S slave mode. Refer to *Section 27.5.10: SPI error flags* and *Section 27.7.6: I^2S error flags*.

This flag is set by hardware and reset when SPIx_SR is read by software.

0: No frame format error
1: A frame format error occurred

Bit 7 **BSY:** Busy flag

0: SPI (or I2S) not busy
1: SPI (or I2S) is busy in communication or Tx buffer is not empty
This flag is set and cleared by hardware.

Note: The BSY flag must be used with caution: refer to Section 27.5.9: SPI status flags and Procedure for disabling the SPI.

Bit 6 **OVR:** Overrun flag

0: No overrun occurred
1: Overrun occurred
This flag is set by hardware and reset by a software sequence. Refer to *I^2S error flags* for the software sequence.

Bit 5 **MODF:** Mode fault

0: No mode fault occurred
1: Mode fault occurred

This flag is set by hardware and reset by a software sequence. Refer to *Mode fault (MODF)* for the software sequence.

Bit 4 **CRCERR:** CRC error flag

0: CRC value received matches the SPIx_RXCRCR value
1: CRC value received does not match the SPIx_RXCRCR value

This flag is set by hardware and cleared by software writing 0.

Bit 3 **UDR:** Underrun flag

0: No underrun occurred
1: Underrun occurred

This flag is set by hardware and reset by a software sequence. Refer to *I^2S error flags*for the software sequence.

Note: This bit is not used in SPI mode.

Bit 2 **CHSIDE:** Channel side

0: Channel Left has to be transmitted or has been received
1: Channel Right has to be transmitted or has been received

Note: This bit is not used in SPI mode. It has no significance in PCM mode.

Bit 1 **TXE:** Transmit buffer empty

0: No more empty space in Tx buffer. (software shall not write data to the Tx buffer).
1: At least one empty space in Tx buffer. (software may write data to the Tx buffer).

Bit 0 **RXNE:** Receive buffer not empty

0: Rx buffer empty
1: Rx buffer not empty

27.9.4 SPI data register (SPIx_DR)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:0 **DR[15:0]:** Data register

Data received or to be transmitted

The data register serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses TxFIFO (See Section 27.5.8: Data transmission and reception procedures).

Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.

27.9.5 SPI CRC polynomial register (SPIx_CRCPR)

Address offset: 0x10

Reset value: 0x0007

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

Bits 15:0 **CRCPOLY[15:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0007h) is the reset value of this register. Another polynomial can be configured as required.

Note: The polynomial value should be odd only. No even value is supported.

27.9.6 SPI Rx CRC register (SPIx_RXCRCR)

Address offset: 0x14

Reset value: 0x0000

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

Bits 15:0 **RXCRC[15:0]:** Rx CRC register

When CRC calculation is enabled, the RxCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPIx_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

A read to this register when the BSY Flag is set could return an incorrect value.

27.9.7 SPI Tx CRC register (SPIx_TXCRCR)

Address offset: 0x18

Reset value: 0x0000

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

Bits 15:0 **TxCRC[15:0]:** Tx CRC register

When CRC calculation is enabled, the TxCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPIx_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY flag is set could return an incorrect value.
These bits are not used in I^2S mode.

27.9.8 SPIx_I^{2}S configuration register (SPIx_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ASTR TEN	I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
			rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved: Forced to 0 by hardware

Bit 11 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S mode is selected
Note: This bit should be configured when the SPI is disabled.

Bit 10 **I2SE**: I2S enable
0: I^{2}S peripheral is disabled
1: I^{2}S peripheral is enabled
Note: This bit is not used in SPI mode.

Bits 9:8 **I2SCFG**: I2S configuration mode
00: Slave - transmit
01: Slave - receive
10: Master - transmit
11: Master - receive
Note: These bits should be configured when the I^{2}S is disabled.
They are not used in SPI mode.

Bit 7 **PCMSYNC**: PCM frame synchronization
0: Short frame synchronization
1: Long frame synchronization
Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used).
It is not used in SPI mode.

Bit 6 Reserved: forced at 0 by hardware

Bits 5:4 **I2SSTD**: I^{2}S standard selection
00: I^{2}S Philips standard.
01: MSB justified standard (left justified)
10: LSB justified standard (right justified)
11: PCM standard
For more details on I^{2}S standards, refer to *Section 27.7.2: Supported audio protocols*
Note: For correct operation, these bits should be configured when the I^{2}S is disabled.
They are not used in SPI mode.

Bit 3 **CKPOL**: Steady state clock polarity

0: I^2S clock steady state is low level

1: I^2S clock steady state is high level

Note: For correct operation, this bit should be configured when the I^2S is disabled.
It is not used in SPI mode.

Bits 2:1 **DATLEN**: Data length to be transferred

00: 16-bit data length

01: 24-bit data length

10: 32-bit data length

11: Not allowed

Note: For correct operation, these bits should be configured when the I^2S is disabled.
They are not used in SPI mode.

Bit 0 **CHLEN**: Channel length (number of bits per audio channel)

0: 16-bit wide

1: 32-bit wide

The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in.

Note: For correct operation, this bit should be configured when the I^2S is disabled.
It is not used in SPI mode.

27.9.9 SPIx_I^{2}S prescaler register (SPIx_I2SPR)

Address offset: 0x20

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV							
						rw	rw	rw							

Bits 15:10 Reserved: Forced to 0 by hardware

Bit 9 **MCKOE**: Master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Note: This bit should be configured when the I^{2}S is disabled. It is used only when the I^{2}S is in master mode.

It is not used in SPI mode.

Bit 8 **ODD**: Odd factor for the prescaler

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2)+1

Refer to *Section 27.7.3: Clock generator*

Note: This bit should be configured when the I^{2}S is disabled. It is used only when the I^{2}S is in master mode.

It is not used in SPI mode.

Bits 7:0 **I2SDIV**: I^{2}S linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to *Section 27.7.3: Clock generator*

Note: These bits should be configured when the I^{2}S is disabled. They are used only when the I^{2}S is in master mode.

They are not used in SPI mode.

27.10 SPI/I2S register map

Table 101 shows the SPI/I2S register map and reset values.

Table 101. SPI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPIx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPIx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	Reset value																		0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Table 101. SPI register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	**SPIx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	1	0
0x0C	**SPIx_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPIx_CRCPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	**SPIx_RXCRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPIx_TXCRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**SPIx_I2SCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASTRTEN	I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
	Reset value																				0	0	0	0	0	0		0	0	0	0	0	0
0x20	**SPIx_I2SPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV							
	Reset value																							0	0	0	0	0	0	0	0	1	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

28 Low power autonomous wakeup radio IP (LPAWUR)

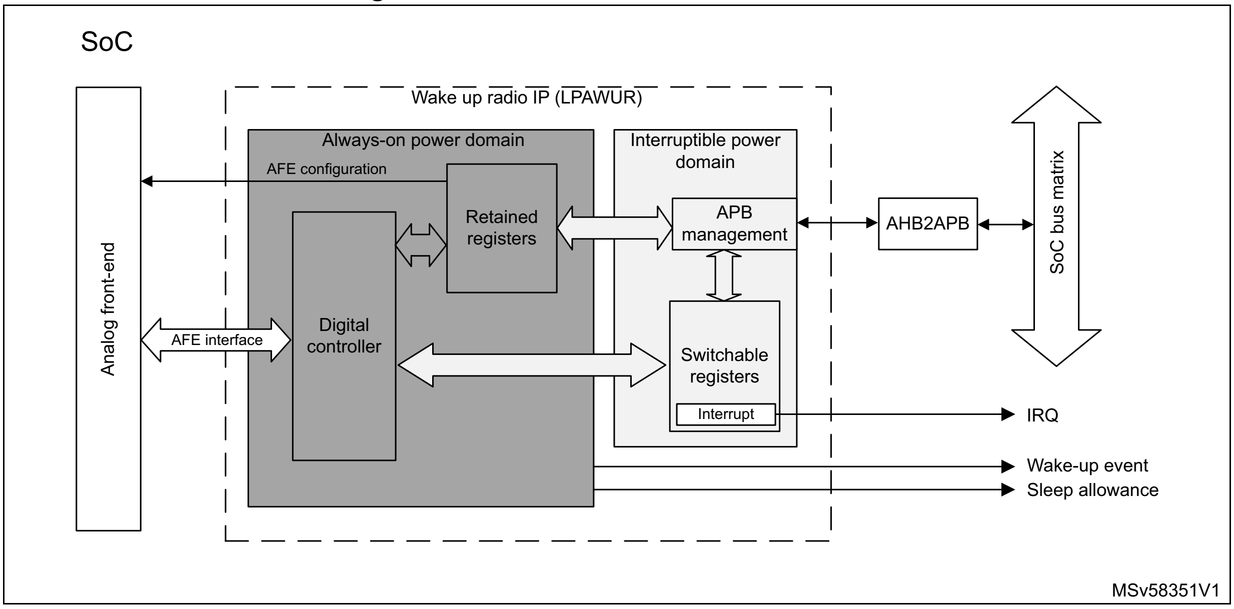
The STM32WL33xx embeds a Low power wakeup radio (LPAWUR) IP. LPAWUR is not available in the STM32WL30xx and STM32WL31xx.

The Low power wakeup radio (LPAWUR) IP is a digital low-power RF IP tat controls the RF wakeup analog front-end (AFE). It uses the reception of a specific frame to trigger the wake up of a system.

The LPAWUR is a small RF IP that is dedicated to this wake-up feature only.

28.1 LPAWUR architecture overview

Figure 265. LPAWUR architecture overview



As shown in *Figure 265*, the LPAWUR contains:

- an always-on power domain (dark grey):
  - The digital Controller interfacing with the AFE and both registers blocks
  - The retained registers
- an Interruptible power-domain (light grey):
  - The switchable registers
  - An APB management interfacing with the Soc Bus Matrix and the registers blocks.

28.2 Interfacing with the LPAWUR

The LPAWUR interfaces with several blocks in the SoC in which it is integrated:

- CPU through interrupts
- power controller block (PWRC) to manage the power supplies
- reset and clock controller block (RCC) to manage the clocks and their dedicated resets.

28.2.1 LPAWUR interrupt lines to the CPU

The LPAWUR IP provides a single interruption line to the CPU, issued by the Registers block.

This interrupt line is dedicated to the wakeup framing, it allows the RFIP to indicate to the SW what the framing detection/ completion status is.

The management of the interrupts is done through the IRQ_ENABLE switchable register which allows to enable an interrupt among the different existing status detailed in the STATUS switchable register.

28.2.2 LPAWUR interface with the power controller (PWRC)

The RFIP_CONFIG retained register defines a WAKEUP_LEVEL[1:0] bit field to allow a wakeup event depending on different root causes.

28.2.3 LPAWUR interface with the reset and clock controller (RCC)

The LPAWUR IP receives two Clocks from the SoC:

- an RFIP system clock at 16 MHz
- a slow clock (named 32 kHz clock in this document) that needs to be in the always-on power domain as it is the clock used when the system is in low power mode.

The LPAWUR IP is mainly used while the system is in low power mode. This means that the slow clock is used most of the time. The 16 MHz system clock is only used for interfacing with the CPU.

Note: The LPAWUR IP supports only 16 MHz as its system clock. The SoC must manage the prescaling on the APB path if the SoC system runs faster (that is, at 32 MHz or 64 MHz). Configuration with a SoC system clock slower than the RFIP 16 MHz is not supported.

Slow clock limitation

The slow clock internal RC oscillator frequency varies from 24 kHz to 49 kHz.

The RF wakeup analog front-end provides information on the rising edge of clk_slow_32k at around 2 kHz, meaning that the information is repeated:

- for 12 clock cycles when clk_slow_32k frequency is 24000 Hz
- for 16 clock cycles when clk_slow_32k frequency is 32000 Hz
- for 24 clock cycles when clk_slow_32k frequency is 48000 Hz

In order to catch the data properly, the LPAWUR IP embeds a retained register which stores the number of expected cycles per encoded bit FRAME_CONFIG0[25:21].SLOW_CLK_CYCLE_PER_BIT_CNT. This information is provided by the SoC once the slow clock has been measured by means of the dedicated MR_SubG feature.

The slow-clock internal oscillator frequency deviation over conditions (temperature and so on) is lower than 10%.

28.3 Packet and protocol construction

The LPAWUR IP is in charge of triggering a wakeup event to the SoC once a specific frame is received and decoded.

The Wakeup frame is a specific frame that can be configured with a set of registers. The following sections detail the packet format, and the registers to be configured.

This IP is not only able to manage the specific Wakeup frame; it can also decode another kind of Wakeup frame with the same format (Bit sync, Frame Sync, Payload and CRC), but with a variable bit width and/or data rate.

28.3.1 Data modulation and encoding

The LPAWUR IP uses OOK modulation only (on/off keying modulation). This modulation is simple:

- on the air, a bit ‘1’ results in the presence of the carrier frequency
- on the air, a bit ‘0’ results in the absence of the carrier frequency

In addition, Manchester encoding is applied to the data that is received by the LPAWUR IP. This encoding consists of representing each bit of the packet as a **transition** between two **symbols**:

- A raw bit ‘0’ results in a transition from low to high
- A raw bit ‘1’ results in a transition from high to low

Note: By default throughout this document, and unless otherwise indicated, the packet (number of bits, data rate) is described without considering Manchester encoding. A prefix “raw” or “decoded” means that we do not consider the Manchester encoding. This is also explicitly indicated to avoid any misunderstanding.

When a bit, data, a frame, or a packet must be considered with Manchester encoding, “with Manchester encoding” is indicated.

28.3.2 Data rate

The default data rate, for a typical slow clock of 32 kHz is of 2 kbit/s seen by the antenna for a packet with Manchester encoding. Because a Manchester encoded data is a couple of two symbols at a different level, this means that a data rate of 2 kbit/s with Manchester encoding represents a data rate of 1 kbit/s for the raw data.

28.3.3 Frame format

The frame format is a specific frame as described below, with a few bit fields that are configurable in order to expand the use of the LPAWUR IP.

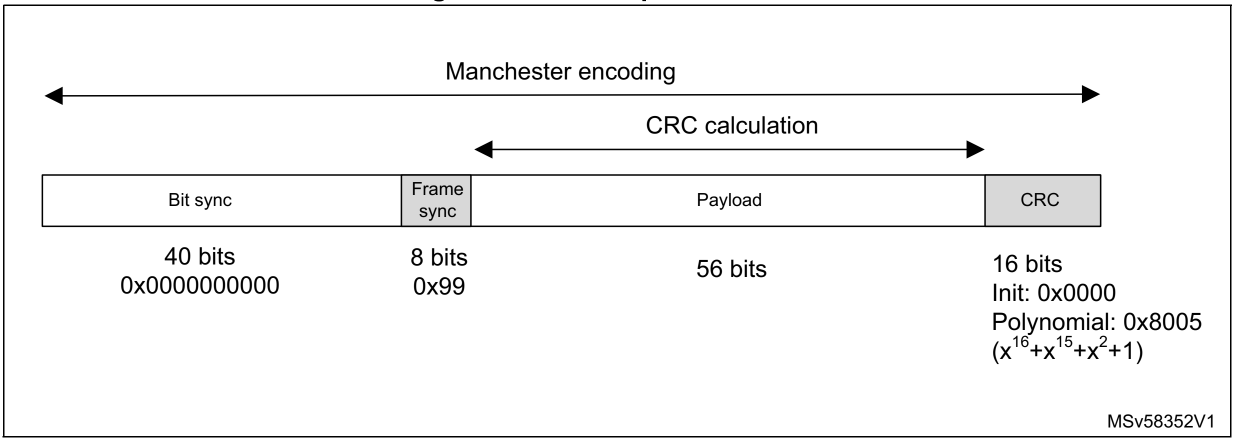
The specific frame is composed of:

- a Bit Sync of 40 bits defined at ‘0’
- a Frame Sync of 8 bits corresponding to the pattern 0x99
- a Payload of 56 bits
- a CRC defined on 16 bits, and calculated on the payload only

This specific Wakeup frame format is “static”, it represents a total of **120 bits**. The whole frame is then encoded with a Manchester algorithm and sent over the air, to be received by the LPAWUR IP. This means that from 120 bits, the complete encoded frame reaches 240 Manchester symbols, thus all 240 symbols are received by the IP (not only 120).

With a typical data rate of 2 kbit/s (Manchester OOK), it represents a duration of 120 ms to receive the 240 symbols.

Figure 266. Wakeup frame format



The frame is sent (and thus received) in the same order as shown in *Figure 266*, with bit sync first, then Frame sync, Payload and CRC last.

Each field of the frame is transmitted over-the-air, and thus received by the LPAWUR IP, with the most-significant byte (MSB) first.

In addition, each byte is transmitted over-the-air, and is thus received by the LPAWUR IP, with the Most Significant bit (MSb) first.

This means that from the LPAWUR IP reception point-of-view, once the bit sync is detected, the **first bit** seen is the most-significant bit of the most-significant byte of the **Frame Sync**. The last bit seen is the least-significant bit (LSb) of the least-significant byte (LSB) of the CRC.

Bit Sync

The Bit Sync is a static field in the specific Wakeup frame, it is a sequence of ‘0’s defined on 40 bits. This bit field is not configurable, for any frame it is always this 40-bit sequence of ‘0’s.

Once Manchester coded, the Bit Sync is seen by the antenna as an 80-bit sequence of alternate ‘0’s and ‘1’s. (This is a classic preamble in RF systems.)

Frame Sync

The Frame Sync is a static field in the specific Wakeup frame, it is defined on 8 bits and always has the value 0x99. It is the default frame if the user does not manage any register to configure a proprietary frame.

The frame sync length is configured through the SYNC_LENGTH[8] bit field of the *FRAME_CONFIG0 register (FRAME_CONFIG0)* retained register:

- ‘0’ (default value) - standard Wakeup frame sync on 8 bits before encoding
- ‘1’ frame sync defined on 16 bits before encoding

The frame sync by default of 0x99 becomes 0x9696 once Manchester encoded. This means that the Sync detection, which is done before the decoding, compares the income frame sync to 0x9696.

This comparison value is not hard written, it is located in the *FRAME_SYNC_CONFIG register (FRAME_SYNC_CONFIG)* retained register, in order to be free to change this sync word. This register is composed of two bit fields:

[15:0]: FRAME_SYNC_PATTERN_L. Contains lower part of the frame sync pattern value stored by the SW (default value is 0x9696)

[31:16]: FRAME_SYNC_PATTERN_H. Contains the higher part of the frame sync pattern value stored by the SW (default value is 0x0000).

By default, only FRAME_SYNC_PATTERN_L is used because the frame sync is defined on 8 bits (16 after encoding). However, if the user decides to use a frame sync length of 16 bits by configuring SYNC_LENGTH = 1, the pattern, once encoded, becomes a pattern on 32 bits, thus the extra 16 higher bits have to be compared to a value which is stored in FRAME_SYNC_PATTERN_H.

Payload

The payload of the specific Wakeup frame has a static length of **maximum** 8 bytes (56 bits by **default**), this is the default frame if the user does not manage a register to configure a proprietary frame.

The payload length is configured through the PAYLOAD_LENGTH[19:16] bit field of the *FRAME_CONFIG0 register (FRAME_CONFIG0)* retained register.

The length is provided in number of bytes. The default value is 7, meaning an expected payload of 56 bits, once decoded.

Note: Only values from 1 to 8 included are allowed in the PAYLOAD_LENGTH[19:16] bit field. Any other configuration must not be used, as the LPAWUR *IP behavior would no longer be guaranteed.*

The payload is received as a Manchester-encoded Bit sync and Frame sync. Once received by the LPAWUR RX block, it is first decoded, and the number of payload bits received is then compared to the expected length configured in the previous register.

Once decoded, the data is stored bit per bit into the PAYLOAD_x retained registers with the Least Significant Byte First.

- If the payload length is less than, or equal to 32 bits, only the PAYLOAD_0 retained register is filled
- If the payload is greater than 32 bits, the PAYLOAD_1 retained register is also filled.

CRC

The CRC of the Wakeup frame is defined on 16 bits, it is only calculated on the payload (excluding Bit Sync and Frame Sync). This CRC uses the CRC-16/ARC algorithm with the following polynomial: 0x8005 ($x^{16} + x^{15} + x^2 + 1$).

The LPAWUR RX block computes each decoded payload bit to generate the CRC at reception in order to compare it with the received CRC.

The CRC starts being received once the expected number of payload bytes, PAYLOAD_LENGTH[19:16] is received.

Once the complete CRC is received, it is compared to the computed CRC to determine whether the frame is valid, or if there is an error:

- If the comparison is TRUE, then the ERROR_F[31:30] bit field of the STATUS switchable register is equal to “00”, meaning that no error occurred. Thus the received frame is valid.
- If the comparison is FALSE, then the ERROR_F[31:30] bit field of the STATUS switchable register is equal to “11” to indicate that a CRC error occurred. Thus the received frame is not valid.

28.4 LPAWUR IP register overview

The register block of the LPAWUR IP manages the different sets of registers used to configure the wakeup feature, to store the received payload and to generate the interrupts dedicated to the CPU.

All the registers are accessible through an APB interface to allow the SW to control the LPAWUR.

The LPAWUR IP configuration is done through the registers, before the wakeup feature is enabled.

28.4.1 Register organization

The registers of the LPAWUR IP are divided in two categories:

- **Retained** registers: these registers keep their content when the SoC is in low power mode. They are used to configure the LPAWUR IP, enable it once it is ready to be used, and to store the payload.
- **Switchable** registers: these registers are solely dedicated to the interactions between the LPAWUR IP and the Software, meaning that they are only used when the CPU is running, that is, when the SoC is in active mode. There are both status registers, which are “read / write 1 to clear”, and the interrupt enables, which allow the generation of CPU interrupts.

Table 102 provides the offset value for the retained and switchable registers:

Table 102. Register sub-group base address

Registers sub-group	Offset versus the register base address
Retained registers	0x00
Switchable registers	0x40

IRQ management

The LPAWUR IP contains one IRQ line, which is fully managed by the switchable register block. The switchable registers are located in the switchable power domain, meaning that they are only supplied when the SoC is in Run mode, and with the 16 MHz clock in order to communicate between the CPU and the registers.

There are four existing interrupts which are described in the *IRQ_ENABLE register (IRQ_ENABLE)* switchable register, with the according status defined in the *STATUS register (STATUS)* switchable register:

- BIT_SYNC_DETECTED_E/F to inform that the Bit Sync (or preamble) have been correctly detected.
- FRAME_SYNC_COMPLETE_E/F to inform that the frame sync has been detected, without any information on the payload.
- FRAME_COMPLETE_E/F to inform that the payload and CRC have been received but with an error in the framing or in the CRC. The detail is provided by the ERROR_F status.
- FRAME_VALID_E/F to inform that the payload and CRC have been received without any error (CRC check is valid).

The ERROR_F status allows more detailed reporting of any error in the RX chain:

- “00”: No Error. The frame has been correctly received, the payload length is valid, and the CRC has been checked.
- “01”: Sync Error. The bit sync has correctly been received but there is an issue on the reception of the sync word, which is either not received in the timeout, or does not match with the expected.
- “10”: Framing Error. The received payload or CRC is not valid because of a Manchester decoding issue, or the number of bits is not correct.
- “11”: CRC Error. The received CRC is different than the CRC which has been computed with the received payload.

All the fields of the *STATUS register (STATUS)* register are Read/Write-1-to-clear, meaning that they are readable at any moment, and it is necessary to Write a ‘1’ in the interrupt source we want to clear.

28.5 LPAWUR retained register descriptions

The **RETAINED_REG_BLOCKBaseAddress** keyword is used as the base address of all Retained registers. Each register address is built by adding an offset to this base address. It is defined by the SoC when integrating the LPAWUR IP.

Note: RETAINED_REG_BLOCKBaseAddress is 0x4900_1000 in the STM32WL33xx product

Table 103. LPAWUR retained register list

Offset	Register Name	Register Title	Reset
0x0000	FRAME_CONFIG0	FRAME_CONFIG0 register	0x0207 4012
0x0004	FRAME_CONFIG1	FRAME_CONFIG1 register	0x0002 4669
0x0008	FRAME_SYNC_CONFIG	FRAME_SYNC_CONFIG register	0x0000 9696
0x000C	RFIP_CONFIG	RFIP_CONFIG register	0x0000 0006
0x0010	RF_CONFIG	RF_CONFIG register	0x0001 33EE
0x0014	AGC_CONFIG	AGC_CONFIG register	0x0000 0000
0x001C	PAYLOAD_0	PAYLOAD_0 register	0x0000 0000
0x0020	PAYLOAD_1	PAYLOAD_1 register	0x0000 0000

28.5.1 FRAME_CONFIG0 register (FRAME_CONFIG0)

Address offset: 0x0000

Reset value: 0x0207 4012

FRAME_CONFIG0 register

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

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:21 **SLOW_CLK_CYCLE_PER_BIT_CNT[4:0]**: The number of expected slow clock cycle per each manchester coded bit.

Default value 0x10 for 16 slow clock cycles. Allowed values are in the range from 0x0C up to 0x18 (respectively for a slow clock at 24 kHz and 48 kHz).

Bit 20 Reserved, must be kept at reset value.

Bits 19:16 **PAYLOAD_LENGTH[3:0]**: The number of data bytes in the payload (decoded).

Only the values 1 to 8 included are supported, other values are not supported and shall not be used.

Default value is 7 bytes / 56 bits

Bits 15:10 **SYNC_THRESHOLD_COUNT[5:0]**: detection threshold when receivng the Frame sync (Manchester encoded). The frame sync detection uses a correletor to detect the address.

if SYNC_LENGTH=0, recommended threshold is 16 (0x10), if SYNC_LENGTH=1, recommended threshold is 32 (0x20)

Default value is 16

Bit 9 Reserved, must be kept at reset value.

Bit 8 **SYNC_LENGTH**: Frame sync pattern length (Manchester encoded).

- If sync_len=0, a 16-bits frame sync is used, the Low word (FRAME_SYNC_PATTERN_L) is describing the frame sync pattern value. (default)

- If sync_len=1, a 32-bits frame sync is used, both Low (FRAME_SYNC_PATTERN_L) and High (FRAME_SYNC_PATTERN_H) words are storing the pattern value.

Bits 7:0 **PREAMBLE_THRESHOLD_COUNT[7:0]**: The number of transitions for preamble detection when receiving the manchester encoded preamble.

Default value is 18

28.5.2 FRAME_CONFIG1 register (FRAME_CONFIG1)

Address offset: 0x0004

Reset value: 0x0002 4669

FRAME_CONFIG1 register

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **TREC_LOOP_ALGO_SEL**: Timing recovery loop algorithm selection.

0: The timing recovery loop algorithm is a second order loop with a defined leakage (0.98) (default)

1: The timing recovery loop algorithm is with two steps. First order loop until preamble detected and second order loop after

Bit 17 **PREAMBLE_ENABLE**: Preamble detection enable

1: WAKEUP_LEVEL The sync detection starts after the preamble found, (default)

0: The sync detection starts when the LPAWUR RX is enabled

The timeout works coherently. i.e if pd_en is set to 1 after the pd_found the timeout start to count the symbols. If pd_en is set to 0,the timeout parameters must be set to 0 which means no count

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **FRAME_SYNC_COUNTER_TIMEOUT[7:0]**: The timeout in manchester encoded bits for the Frame Sync,it represents the number of samples after which in case the frame sync is not detected a sync_error is raised. The default value is 70 (0x46)

Bits 7:4 **KP[3:0]**: kp gain value for the timing recovery loop. (default value = 6)

Bits 3:0 **KI[3:0]**: ki gain value for the timing recovery loop. (default value = 9), (Note that if Ki gain is set to 0 the loop is configured as a first order loop)

28.5.3 FRAME_SYNC_CONFIG register (FRAME_SYNC_CONFIG)

Address offset: 0x0008

Reset value: 0x0000 9696

FRAME_SYNC_CONFIG register

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

Bits 31:16 **FRAME_SYNC_PATTERN_H[15:0]**: The value of the frame sync pattern, High word, manchester encoded, used only when the frame sync length is 32 bits (default 0x0000)

Bits 15:0 **FRAME_SYNC_PATTERN_L[15:0]**: The value of the frame sync pattern, Low word, Manchester encoded, used when the frame sync length is 16 bit (default 0x9696 which represents a frame sync value of 0x99)

28.5.4 RFIP_CONFIG register (RFIP_CONFIG)

Address offset: 0x000C

Reset value: 0x0000 0006

RFIP_CONFIG register

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

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

Bits 2:1 **WAKEUP_LEVEL[1:0]**:
- 00: the bit Sync has been detected
- 01: Frame sync detected
- 10: Frame_complete detected
- 11: Frame_Valid detected (default)

Bit 0 **LPAWUR_ENABLE**:
Enable (start) or Disable (stop) the LPAWUR feature (0: disabled by default)

28.5.5 RF_CONFIG register (RF_CONFIG)

Address offset: 0x0010

Reset value: 0x0001 33EE

RF_CONFIG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPF3_CAL	ED_ICAL[2:0]			AGC_HIGH_LVL[3:2]	
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AGC_HIGH_LVL[1:0]		ED_DC_CTRL	AGC_LOW_LVL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CLKDIV[3:0]				ED_SWITCH
rw	rw	rw	rw	rw							rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **LPF3_CAL**: None

Bits 20:18 **ED_ICAL[2:0]**: Current versus VBAT calibration for ED
- 000: VBAT=1.7 .. 2.0 V (default)
- 001: VBAT=2.0 .. 2.25V
- 010: VBAT=2.25 .. 2.5V
- 011: VBAT=2.5 .. 2.75V
- 100: VBAT=2.75 .. 3.0 V
- 101: VBAT=3.0 V .. 3.25V
- 110: VBAT=3.25V ..3.5V
- 111: VBAT=3.5..3.7V

Bits 17:14 **AGC_HIGH_LVL[3:0]**: AGC level (High) (default value: 0x4)

Bits 12:11 **AGC_LOW_LVL[1:0]**: AGC level (Low) (default value: 0x2)

Bit 13 **ED_DC_CTRL**: DC current subtraction enabling signal (default value: 0x1)

Bits 10:5 Reserved, must be kept at reset value.

Bits 4:1 **CLKDIV[3:0]**: Calibrate 4 kHz clock (programmable divider used by the analog part of the IP)
0000: Divide by 1
...
0111: Divide by 8 (default)
...
1110: Divide by 15
1111: Off

Bit 0 **ED_SWITCH**:
0: Normal operation (default)
1: Use ED as input switch (test mode)

28.5.6 AGC_CONFIG register (AGC_CONFIG)

Address offset: 0x0014

Reset value: 0x0000 0000

AGC_CONFIG register

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

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

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

Bit 3 **AGC_RESET_MODE**: The AGC reset behavior when the AGC is working in ON or HOLD mode

0: reset the AGC after the complete frame detection (default)

1: the AGC is never reset

Bit 2 **AGC_HOLD_MODE**: The behavior when the AGC is ON and is working in HOLD mode

0: hold after the preamble detection (default)

1: hold after the sync detection

Bits 1:0 **AGC_MODE[1:0]**: Define the working mode of the AGC:

00: AGC MAX: the AGC is ON and is always operating at maximum value (default)

01: AGC OFF: the AGC is OFF

10: AGC ON: the AGC is ON

11: AGC HOLD: the AGC is ON and its value is frozen during the frame detection, depending of the AGC_HOLD_MODE configuration after a preamble or after the frame sync.

28.5.7 PAYLOAD_0 register (PAYLOAD_0)

Address offset: 0x001C

Reset value: 0x0000 0000

PAYLOAD_0 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAYLOAD_0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PAYLOAD_0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PAYLOAD_0[31:0]**: First part of the payload (Least significant Byte First)

28.5.8 PAYLOAD_1 register (PAYLOAD_1)

Address offset: 0x0020

Reset value: 0x0000 0000

PAYLOAD_1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAYLOAD_1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PAYLOAD_1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PAYLOAD_1[31:0]**: Second part of the payload (Least significant Byte First)

28.5.9 LPAWUR retained register summary

Table 104. LPAWUR retained register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RETAINED_REG_BLOCK_FRAME_CONFIG0	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_CLK_CYCLE_PER_BIT_CNT[4:0]					Res.	PAYLOAD_LENGTH[3:0]				SYNC_THRESHOLD_COUNT[5:0]						Res.	SYNC_LENGTH	PREAMBLE_THRESHOLD_COUNT[7:0]							
	Reset value							1	0	0	0	0		0	1	1	1	0	1	0	0	0	0		0	0	0	0	1	0	0	1	0
0x004	RETAINED_REG_BLOCK_FRAME_CONFIG1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREC_LOOP_ALGO_SEL	PREAMBLE_ENABLE	Res.	FRAME_SYNC_COUNTER_TIMEOUT[7:0]								KP[3:0]				KI[3:0]			
	Reset value														0	1		0	1	0	0	0	1	1	0	0	1	1	0	1	0	0	1
0x008	RETAINED_REG_BLOCK_FRAME_SYNC_CONFIG	FRAME_SYNC_PATTERN_H[15:0]																FRAME_SYNC_PATTERN_L[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0
0x00C	RETAINED_REG_BLOCK_RFIP_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_LEVEL[1:0]		LPAWUR_ENABLE
	Reset value																														1	1	0
0x010	RETAINED_REG_BLOCK_RF_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPF3_CAL	ED_ICAL[2:0]			AGC_HIGH_LVL[3:0]				ED_DC_CTRL	AGC_LOW_LVL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CLKDIV[3:0]				ED_SWITCH
	Reset value											0	0	0	0	0	1	0	0	1	1	0							0	1	1	1	0

Table 104. LPAWUR retained register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	RETAINED_REG_BLOCK_AGC_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_RESET_MODE	AGC_HOLD_MODE	AGC_MODE[1:0]	
	Reset value																													0	0	0	0
0x018	Reserved																																
0x01C	RETAINED_REG_BLOCK_PAYLOAD_0	PAYLOAD_0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	RETAINED_REG_BLOCK_PAYLOAD_1	PAYLOAD_1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

28.6 LPAWUR switchable register descriptions

The SWITCHABLE_REG_BLOCKBaseAddress keyword is used as the base address of the whole Switchable registers, each register address is built by adding an offset to this base address. It is defined by the SoC when integrating the LPAWUR IP.

Note: SWITCHABLE_REG_BLOCKBaseAddress is 0x4900_1040 in the STM32WL33xx product.

The 0x40 offset is due to the fact the Switchable registers mapping is following the Retained registers mapping, which has a size of 0x40.

Table 105. LPAWUR register list

Offset	Register Name	Register Title	Reset
0x0000	RFIP_VERSION	RFIP_VERSION register	0x0000 1100
0x0004	IRQ_ENABLE	IRQ_ENABLE register	0x0000 0000
0x0008	STATUS	STATUS register	0x0000 0000

28.6.1 LPAWUR switchable register descriptionsRFIP_VERSION register (RFIP_VERSION)

Address offset: 0x0000

Reset value: 0x0000 1100

RFIP_VERSION register

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

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

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

Bits 15:12 **PRODUCT[3:0]**: Used for major upgrades (new protocols support / new features)

Bits 11:8 **VERSION[3:0]**: Version of the RFIP (to be used for cut upgrades)

Bits 7:4 **REVISION[3:0]**: Revision of the RFIP to be used for metal fixes)

Bits 3:0 Reserved, must be kept at reset value.

28.6.2 IRQ_ENABLE register (IRQ_ENABLE)

Address offset: 0x0004

Reset value: 0x0000 0000

IRQ_ENABLE register

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

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

Bit 3 **FRAME_VALID_E**: Frame (payload + CRC) received wthout error (the CRC has been checked and is matching with the received CRC).

The content of the PAYLOAD_X registers is valid.

Bit 2 **FRAME_COMPLETE_E**: Frame (payload + CRC) received, the content of the PAYLOAD_X registers is valid.

Note: the frame may have been received with CRC or Framing error.

Bit 1 **FRAME_SYNC_COMPLETE_E**: Frame Sync has been detected, the content of the PAY-LOAD_X registers is not yet valid.

Bit 0 **BIT_SYNC_DETECTED_E**: Preamble has been detected, the content of the PAYLOAD_X reg-isters is not yet valid.

28.6.3 STATUS register (STATUS)

Address offset: 0x0008

Reset value: 0x0000 0000

STATUS register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERROR_F[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rc_w1	rc_w1														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRAME_VALID_F	FRAME_COMPLETE_F	FRAME_SYNC_COMPLETE_F	BIT_SYNC_DETECTED_F
												rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:30 **ERROR_F[1:0]**:

11: CRC error

10: Framing error

01: Sync Error

00: No Error

Bits 29:4 Reserved, must be kept at reset value.

Bit 3 **FRAME_VALID_F**: Frame (payload + CRC) received wthout error (the CRC has been checked and is matching with the received CRC).

The content of the PAYLOAD_X registers is valid.

Bit 2 **FRAME_COMPLETE_F**: Frame (payload + CRC) received, the content of the PAYLOAD_X registers is valid.

Note: the frame may have been received with CRC or Framing error.The ERROR_F is describing the status of the reception.

Bit 1 **FRAME_SYNC_COMPLETE_F**: Frame Sync has been detected, the content of the PAYLOAD_X registers is not yet valid.

Bit 0 **BIT_SYNC_DETECTED_F**: Preamble has been detected, the content of the PAYLOAD_X registers is not yet valid.

28.6.4 LPAWUR register summary

Table 106. LPAWUR register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SWITCHABLE_REG_BLOCK_ RFIP_VERSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRODUCT[3:0]				VERSION[3:0]				REVISION[3:0]				Res.	Res.	Res.	Res.
	Reset value																	0	0	0	1	0	0	0	1	0	0	0	0				
0x004	SWITCHABLE_REG_BLOCK_ IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRAME_VALID_E	FRAME_COMPLETE_E	FRAME_SYNC_COMPLETE_E	BIT_SYNC_DETECTED_E
	Reset value																													0	0	0	0
0x008	SWITCHABLE_REG_BLOCK_ STATUS	ERROR_F[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRAME_VALID_F	FRAME_COMPLETE_F	FRAME_SYNC_COMPLETE_F	BIT_SYNC_DETECTED_F
	Reset value	0	0																											0	0	0	0

28.7 Radio controller

The radio controller is in charge of managing the correct behavior of the whole LPAWUR IP, it is the sequencer of the IP.

Its main tasks are to:

- enable and control the analog front-end (AFE)
- enable the LPAWUR RX and get feedback to raise status flags
- notify the SoC about the availability of the
- LPAWUR IP to go in sleep mode.

28.7.1 Radio controller sequencing

The radio controller is triggered by the software who has to use the LPAWUR_ENABLE bit from the *RFIP_CONFIG register (RFIP_CONFIG)* retained register.

Once the LPAWUR IP enabled, the radio controller reads the *RF_CONFIG register (RF_CONFIG)* and *AGC_CONFIG register (AGC_CONFIG)* retained registers to check the configurations of the AFE.

The analog part is enabled first in order to allow time for the precharge. This ensures that the radio is ready to receive some data over-the-air.

Once the precharge done, the radio controller enables the LPAWURRX block.

At this moment, each block works autonomously, the radio controller only waits for input signals to determine how to handle the next operations.

It may choose to trigger a wakeup event, or even a sleep allowance, as described in *Section 28.7.2: Wakeup event management* and *Section 28.7.3: Sleep management.*

However, it may also choose to restart a reception, or even to internally shut down the LPAWUR RX block.

- If the WAKEUP_LEVEL is defined as “bit sync level’ (“00”), and the LPAWUR RX sent an error report, such as a sync error, the detection is disabled.
- If the WAKEUP_LEVEL is different of “bit sync level’ (“00”), and the LPAWUR RX sent an error report, such as a sync error, the detection is restarted internally
- If the WAKEUP_LEVEL is different of “frame valid level (“11”), and the LPAWUR RX indicated an end of frame report, the detection is disabled.
- If the WAKEUP_LEVEL is equal to “frame valid level (“11”), and the LPAWUR RX raised a FRAME_COMPLETE without any error, the wakeup frame detection is stopped
- If the WAKEUP_LEVEL is equal to “frame valid level (“11”), and the LPAWUR RX raised a FRAME_COMPLETE with an error, the frame detection is restarted from the beginning.

28.7.2 Wakeup event management

The wakeup event consists in rising a wakeup pulse to the PWRC (Power and Reset Controller) of the SoC to indicate that the LPAWUR IP requires the SoC to be in Run mode. The IP does not know in what mode the SoC is running:

- If the SoC is already in Run mode, nothing happens. This means that the 16 MHz clock is already available, and the SoC is probably performing an operation
- If the SoC is in SLEEP mode, the PWRC wakes up the whole system, but the RFIP does not see any feedback from the SoC.

Basically, a wakeup event may be desired when a wakeup frame has been received and is valid. This means that a transmitter wanted to establish a connection, and probably expects a feedback from the LPAWUR IP.

Different wakeup levels are defined in the WAKEUP_LEVEL bitfield of the *RFIP_CONFIG register (RFIP_CONFIG)* retained register:

- WAKEUP_LEVEL = “00” (Bit sync detected)
  – the wakeup event is raised once the preamble is detected
- WAKEUP_LEVEL = “01” (Frame sync detected)
  – the wakeup event is raised once the frame sync is detected
- WAKEUP_LEVEL = “10” (Frame complete)
  – the wakeup event is raised once the whole frame is detected, considering both the payload plus the CRC
- WAKEUP_LEVEL = “11” (Frame valid)
  – the wakeup event is raised once the whole frame is detected without any error, meaning that the CRC has been checked and is matching with the received CRC.

Note: The wakeup events on a bit sync detected and a frame sync detected are not intended to be used in a functional point of view, it is for test purpose only.

28.7.3 Sleep management

Most of the time, the LPAWUR IP works in sleep mode of the SoC, with the always-on power domain only to use the retained registers or to receive and decode a frame.

However, there is a wakeup event which allows the switchable power domain to be enabled, typically for accessing the APB registers. Thus, there is also a sleep allowance signal which allows the IP to warn the SoC that it is currently working or that is ready to return to a sleep mode.

When the sleep allowance is set at 1, the RFIP is allowing the system to enter in sleep mode, when the sleep allowance is set at 0, the IP does not allow the system to enter sleep mode.

- By default, the **sleep is allowed**, so it is also the case when the LPAWUR IP is reset.
- When a preamble is detected by the LPAWUR RX, the **sleep allowance** signal is cleared
- When a Sync Error is detected by the LPAWUR RX while it is trying to detect a frame sync, the **sleep allowance** signal is asserted
- When an end of frame is detected by the LPAWUR RX while it is waiting for receiving a payload, the **sleep allowance** signal is asserted.

This signal may only be active when the SoC is in Run mode, thus it is based on the **clk_sys_16m** and not the slow clock as the other signals.

Note: A wakeup event may have been sent at the same time as a sleep allowed information. In this case, it is the role of the PWRC (Power and Reset Controller) of the SoC to manage the point.

28.7.4 Recommended register values

This section gives some recommended values to be programmed for the LPAWUR IP registers in order to optimize the wakeup frame detection performance.

The following register default values should be modified by the Firmware:

- *FRAME_CONFIG1 register (FRAME_CONFIG1)* retained register. The recommended value of this register is: 0x6466A
- *RF_CONFIG register (RF_CONFIG)* retained register. The recommended value of this register is: 0x2123EE

29 Sub-GHz radio IP (MRSUBG)

29.1 Overview

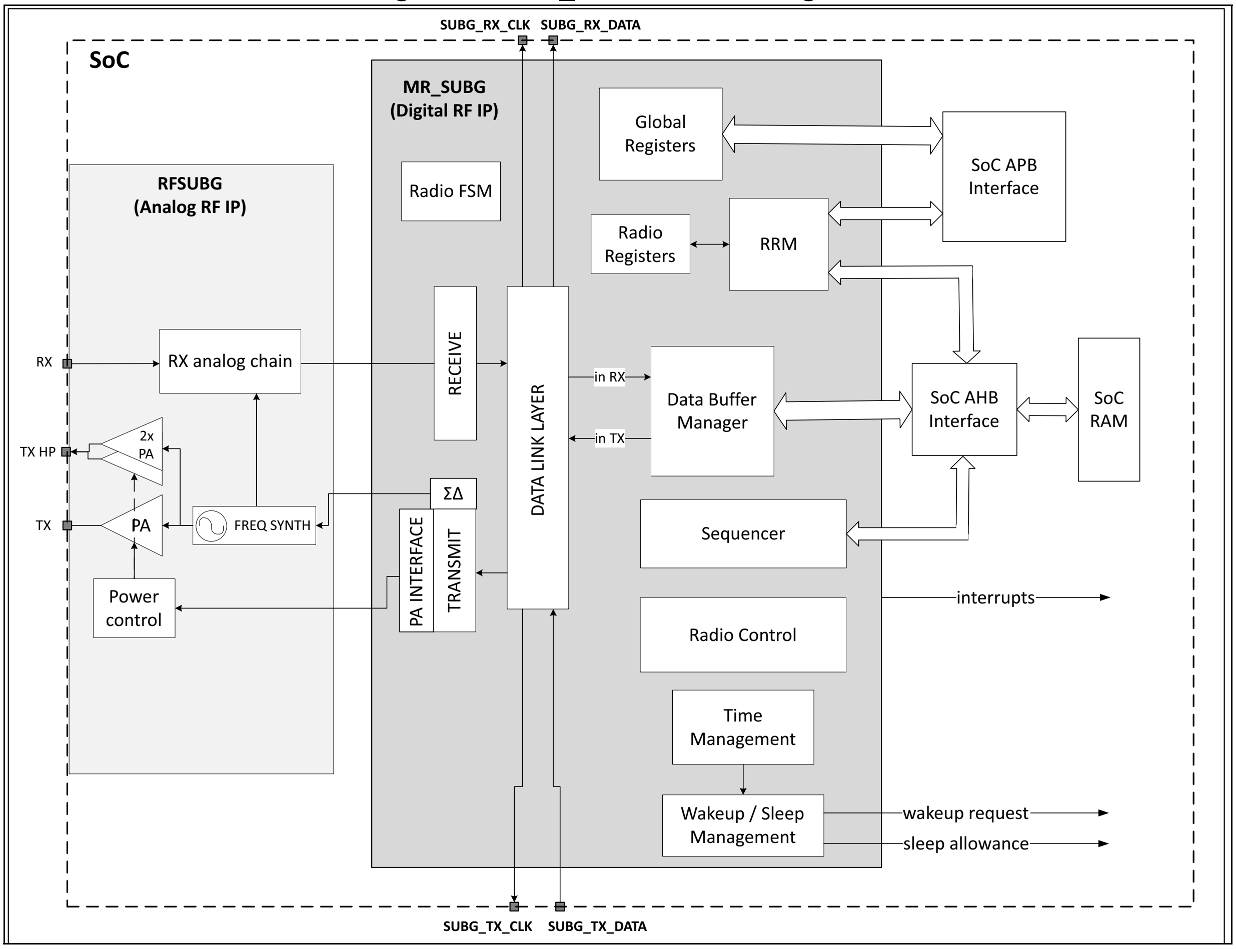
The MR_SubG IP is the digital part of a Sub-GHz radio working in association with the analog RFSUBG IP.

The MR_SubG IP:

- controls the analog RF block
- offers the possibility to create customized protocols by choosing modulation type, data rate, frequency deviation, packet format, and so on
- embeds a sequencer, allowing an applicative scenario to be built with minimum interference of the CPU.

29.1.1 MR_SubG IP overview

Figure 267. MR_SubG IP block diagram



As shown in *Figure 267: MR_SubG IP block diagram*, the MR_SubG IP contains:

- Global registers block
- A Radio controller
- A RRM (Radio Registers Manager) block containing:
  - a UDRA: mainly used after a low power mode exit to restore some Radio registers (if needed) through a command link list located in retention RAM

Note: This sub-block is an AHB master

  - a Direct Radio Register Access allowing read/write access to radio registers through APB.
- A Radio register block
  - to configure the analog radio parameters
  - to configure some digital modulator/demodulator parameters

Note: Most of the radio registers contained the correct settings by default.

- A Radio FSM controlling the RFSUBG analog radio.
- a transmitter block containing:
  - the framer
  - the modulator
- a receive block containing:
  - the demodulator
  - the deframer
  - the AGC controller
  - the I/Q compensation
  - the AFC controller
- a data buffer manager to manage read / write of data to transmit / received data in the SoC RAM
- a sequencer
- a wakeup block managing sleep allowance and wakeup request, including a wakeup timer
- Some small blocks used for calibrations, PA control, PLL control, and RX data path interface (AGC, FIFO ADC) and so on.

The MR_SubG IP supports only one system clock frequency: 16 MHz.

29.1.2 Miscellaneous feature: control of an external components through pads of the SoC

Control an external power amplifier (PA)

The MR_SubG IP is able to control an external Power Amplifier (located on the board to increase the Tx power) through a signal provided to the SoC (TX_SEQUENCE) to be connected directly on an external GPIO or to be managed with additional logical mechanism. The external PA can be useful to extend the TX power.

Hence this provided signal is raised when the transmitter device starts putting power on the antenna (Radio FSM is EN_PA and TX states).

Control an external LNA

In the same idea, the MR_SubG IP is able to control an external LNA through a signal provided to the SoC (RX_SEQUENCE) to be connected directly on an external GPIO or to be managed with additional logical mechanism.

The timing / sequence to raise this signal has been taken from the existing RX state information provided by the S2-LP product.

So this provided signal is raised when the receiver device starts the analog RX chain (Radio FSM is EN_RX, EN_LNA and RX states), knowing the capacity to decode received information is only available from state RX.

Control an antenna switching block (2 antennas)

The MR_SubG IP embeds an antenna selection feature. When enabled, the RX_Top tries two antennas and select the one with the higher measured RSSI. To select one antenna or the other, the MR_SubG IP provides an rfip_antenna_switch signal to the SoC to be connected directly on an external GPIO (ANTENNA_SWITCH in *the STM32WL30xx/31xx/33xx*) or to be managed with additional logical mechanism.

Refer to chapter *Section 29.11.6: Antenna switching* for details on the feature.

29.2 Interfacing with the MR_SubG IP

The MR_SubG IP interfaces with several blocks in the system:

- CPU through interrupts
- RAM through AHB interface either initiated by the RRM, the DBM or the sequence
- Power controller block and clock and reset controller block

29.2.1 Interrupt lines to the CPU

The MR_SubG IP provides several interruption lines to the CPU, issued by different sub-blocks:

- 1 interrupt line dedicated to the RF sequences grouping all status flag from the global RFSEQ_IRQ_STATUS register (active only if the associated bit(s) in RFSEQ_IRQ_ENABLE register is also set)
  - this interrupt is linked to Radio IP transfers and actions
  - In STM32WL30xx/31xx/33xx devices: mapped on interrupt line 21
- 1 interrupt line from the wakeup block for CPU wakeup event:
  - this interrupt is raised when the wakeup timer reaches the programmed value in CPU_WAKEUPTIME register
  - In STM32WL30xx/31xx/33xx devices: mapped on interrupt line 23
- 1 interrupt line from the wakeup block for RFIP wakeup event
  - In STM32WL30xx/31xx/33xx devices: mapped on interrupt line 24

The interrupt features associated to TX_SEQUENCE, RX_SEQUENCE, rfip_busy and rfip_ahb_busy are managed at system controller (SYSCFG) level.

29.2.2 Interface with the RAM embedded in the SoC

The MR_SubG IP is a multi AHB masters on the bus matrix as the CPU or the DMA.

Note: At SoC level, an intermediate 4-to-1 bus matrix is used so that there is only one AHB master for the MR_SubG IP on the SoC bus matrix

The AHB masters accessing the RAM of the system are:

- the RRM UDRA block to get commands structure in RAM, used to program the Radio registers while the CPU is not available (potentially rebooting after a low power mode),
- the DBM to read data to transmit or write received data

29.2.3 Interface with the power, clock and reset controllers

The MR_SubG IP receives two clocks from the SoC:

- an RFIP system clock at 16 MHz
- a slow clock (called 32 kHz clock in this document), available in the always-on power domain as it corresponds to the clock used to wake-up the device from low power mode.

The fast clock (16 MHz) must be accurate when the radio is used (for the transmission/reception) which means low ppm. The SW is responsible to configure the SoC clock tree on the accurate clock to use the radio.

The SW is also responsible to configure properly the clock ratio between the system and the digital radio IP and to program the AHB UP/DOWN converters through the RCC of the SoC.

The MR_SubG IP is implemented into two power domains:

- an always-on 1.2 V power domain: reduced to 1.0 V during low power mode
- a switchable 1.2 V power domain: OFF when the SoC is in low power mode.

Only the Wakeup block and the retained registers are in the always-on power domain. The rest of the radio is in the switchable area.

The MR_SubG IP communicates with the power controller of the system:

- to indicate when it allows the device going into low power modes
- to request a wakeup of the system.

A SOC_WAKEUP_OFFSET[7:0] bit field allows programming a delay to anticipate the SoC wakeup request versus the associated CPU interrupt. This delay is supposed to let the power and clock tree being established before to generate the event on the CPU.

29.2.4 Interface with the SoC for busy information

The MR_SubG IP provides 2 status signals to indicate when it is busy. Those signals can be then used by the soc different ways (to generate interrupts on falling/rising/both edges, make the signal value readable in a register). There are two signals:

o_rfip_busy:

- this signal indicates when the RFIP can be considered as busy meaning not IDLE
- the RFIP is considered as busy when at least one of the conditions below is true:
  - the Radio FSM is not in IDLE mode
  - an RRM UDRA command is under execution

o_rfip_ahb_busy:

- this signal is a bit more restrictive than the general busy previously described as it is set only when the RFIP is in conditions that could lead to AHB master accesses
- this signal is set if at least one of the conditions below is true:
  - the Radio FSM is in RX state or in TX state
  - an RRM UDRA command is under execution,
  - an AHB DBM access is ongoing
    Note: the Radio FSM RX/TX state info is not enough to cover the activity period of

the DBM due to an early RAM prefetch at the very beginning of a TX sequence done by the DBM/

- In STM32WL30xx/31xx/33xx devices, this signal is used to generate interruption, programmable in the SYSCFG block of the SoC.

29.2.5 Signals connected to SoC GPIOs

The MR_SubG IP provides some signals that can be connected to pads in the SoC.

Some signals are mandatorily routed to SoC pads to use the “Direct through GPIO” TX/RX modes (see sections *Direct through GPIO* and *Direct through GPIO mode* for details):

- When the transmission uses the Direct through GPIO mode
  - TX_DATA input signal to provide serialized bits to be modulated on the antenna by the radio
  - TX_CLOCK output signal providing a data rate clock to indicate when a new TX_DATA bit must be provided
- When the reception uses the Direct through GPIO mode
  - RX_DATA output signal providing the serialized demodulated bits
  - RX_CLOCK output signal providing a data rate clock to sample the RX_DATA

Some signals can be optionally provided on SoC GPIOs or kept only internally for status and/or interrupt purpose:

- TX_SEQUENCE: set when the Radio FSM is in EN_PA and TX states (see *Control an external power amplifier (PA)* for details)
- RX_SEQUENCE: set when the Radio FSM is in EN_RX, EN_LNA and RX states (see *Control an external LNA* for details)
- ANTENNA_SWITCH: used to control an external antenna switching component, this output signal toggles during the antenna selection process and then stays on selected one (see *Control an antenna switching block (2 antennas)* for details)

Note: In STM32WL30xx/31xx/33xx devices, all of these signals are available on GPIOs, if the corresponding Alternate Function is programmed.

29.3 Packet and protocol construction

The MR_SubG IP and the associated analog IP provides a great flexibility in protocol building and support by offering a lot of programmable parameters such as:

- the packet format
- the modulation type
- the data rate

The aim of this product is to build customized / home-made protocols. In parallel, it has been tuned to support few standards like:

- Sub-GHz 802.15.4
- W-MBUS
- Sigfox™ (Transmitter only)

29.3.1 Packet structure

Packet elements

In any protocol, a packet contains a structure with many common elements.

PREAMBLE

The MR_SubG IP offers several format and length of PREAMBLE to the user:

- the PREAMBLE length is programmable from 0 to 2046 bits, with a granularity of 2-bit
- the PREAMBLE pattern can be chosen among a selection of pattern:

Table 107. PREAMBLE pattern selection

PREAMBLE_SEQ[1:0]	In 2-(G)FSK or ASK/OOK	In 4-(G)FSK
00	0101..	0111..
01	1010..	0010..
10	0011.. (not recommended)	1101..
11	1100.. (not recommended)	1000..

Note: The pattern is transmitted as leftmost bit first

2 patterns are mentioned as “not recommended” for 2(-G)FSK and ASK/OOK as they are not the most convenient for the receiver.

Both length and pattern choice are programmable in the Global static register called PCKT_CONFIG (see *PCKT_CONFIG register (PCKT_CONFIG)* for details).

The preamble is essential for locking the frequency and timing recovery algorithms. At least 8/16 preamble bits are required for reliable detection (more bits are better).

SYNC (synchronization word)

The MR_SubG IP can support up to two synchronization words: a primary (SYNC) and a secondary (SEC_SYNC) word. All configuration parameters (except the value of the word itself) are shared by both on SYNC and SEC_SYNC.

Even if the programming allows selecting no SYNC in the frame, a SYNC word is expected to be present in Normal buffer mode.

The synchronization words can be configured as follow:

- indicate if a synchronization word is present or not in the frame using the SYNC_PRESENT bit,
- define the size (in bit) of the SYNC word,
- the SECONDARY_SYNC_SEL bit can be set to indicate a secondary SYNC word has to be managed. When set:
  - in TX mode: the SEC_SYNC is sent instead of the SYNC on the frame
  - in RX mode: the receiver enables the SEC_SYNC detection in parallel of the SYNC word (so frames using one or the other SYNC word are detected by the receiver).

Note: In reception, whatever the SECONDARY_SYNC_SEL bit value:
- *the SYNC word is always detected,*
- *when the packet format is defined as 802.15.4, the secondary SYNC word is automatically searched.*

The size of the SYNC word(s) is defined with a 1-bit granularity in the SYNC_LEN[3:0] bit field and can be programmed from 1-bit long to 32-bit long .

Note: The (SEC_)SYNC is transmitted as leftmost bit first (SEC_)SYNC[31] bit first.

All bit fields to configure the synchronization words are programmable in the Global static register called PCKT_CONFIG (see *PCKT_CONFIG register (PCKT_CONFIG)* for details).

Note: If the SYNC word is too close to the PREAMBLE pattern, the receiver may not manage to detect a valid SYNC word.

LENGTH

The length of the packets can be configured as follow:
- Length is defined in bytes unit (in the *PCKTLEN_CONFIG register (PCKTLEN_CONFIG)*
- the length can be defined in a bit field with a size of 1 or 2 bytes (through the LEN_WIDTH bit field)
- 2 length modes can be selected using the FIX_VAR_LEN bit:
  - FIXED length mode:

    no LENGTH field added inside the transmitted frame and the receiver does no try to decode a LENGTH field on received frame.

    Both receiver and transmitter use the value defined in the PCKTLEN_CONFIG register to know the length of the frame.
  - VARIABLE length mode:

    in TX mode, the device adds the LENGTH information inside the transmitted frame using the PCKTLEN and the LEN_WIDTH bit fields information.

    in RX mode, the device decodes the LENGTH information in the received frame using the LEN_WIDTH bit field information.

The packet length information located in the PCKTLEN_CONFIG register has different meanings depending on the selected RX mode or TX mode.

In TX mode:
- when Normal Buffer mode (both FIXED and VARIABLE length modes): the PCKTLEN bit field represents the number of payload bytes to transmit.
- when Direct through buffer mode: the PCKTLEN bit field represents the number of bytes to transmit before to stop the transmission.

Note: If PCKTLEN=0, then the transmission continues until a SABORT command is issued by the SW.

In RX mode:

- when Normal Buffer mode:
  - if FIXED length mode: the PCKTLEN bit field represents the number of payload bytes
  - if VARIABLE length mode:

    the PCKTLEN bit field represents the maximum receive size in bytes.

    If the receiver decodes in the frame a LENGTH value greater than the PCKTLEN bit field value, it stops the on-going reception.

For all other RX/TX modes, the PCKTLEN bit field is not used.

Note: In a special case of a transmission in Normal mode with PCKTLEN=0, the TX_Top fetches one byte in RAM at the beginning of the transmission:

- *if the Data Buffer size is not null, then the RAM is prefetched early (at the TX command start) knowing the read byte is not used*
- *if the Data Buffer size is null, no RAM access is done in the SoC but the DBM raises a FIFO error flag as the TX_Top requests a data not available inside the DBM.*

PAYLOAD

The payload represents the data to be provided by the SW through the Data Buffers when the Normal Buffer mode is used for transmission or reception.

In the direct through buffer, direct through GPIO or I/Q sampling RX modes, the framer / deframer of the MR_SubG IP does not insert/decode any PREAMBLE, SYNC word, LENGTH, CRC, and so on, and use only the provided data as raw data.

The payload size corresponds to the PCKTLEN value. It can be from 0 to 65 535 bytes.

CRC

The MR_SubG IP implements error detection by means of Cyclic Redundancy Check codes.

The CRC is calculated over all the frame fields except the PREAMBLE, the SYNC word.

The MR_SubG IP supports several CRC length and polynomials through CRC_MODE[2:0] bit field in *PCKT_CONFIG register (PCKT_CONFIG)* static register:

- mode 0: no CRC
- mode 1: CRC 8-bit with poly x^8+x^2+x+1 (0x07)
- mode 2: CRC 16-bit with poly x^16+x^15+x^2+1 (0x8005)
- mode 3: CRC-16-bit with poly x^16+x^12+x^5+1 (0x1021) (802.15.4 compliant)
- mode 4: CRC 24-bit with poly x^24+x^23+x^18+x^17+x^14+x^11+x^10+x^7+x^6+x^5+x^4+x^3+x+1) (0x864CFB)
- mode 5: CRC 32-bit with poly x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x+1 (0x04C11DB7) (802.15.4 compliant)

For all CRC modes:

- the CRC initialization is programmable through the *CRC_INIT register (CRC_INIT)* register,
- no reflection is applied (RefIn = RefOut = false)
- no XOR on output (XORout = all 0s)
  - except for CRC mode5 when PCKT_CTRL[0] = PCKT_FORMAT = 1 (802.15.4): XORout = all 1s so CRC is inverted versus CRC mode5 with PCKT_FORMAT = 0 =Basic

Note: CRC versus coding:

- the CRC is calculated before whitening,
- the CRC is calculated on clear data when Manchester and FEC coding are selected,
- the CRC is calculated on the encoded data when 3oo6 coding is selected.

POSTAMBLE

The packet POSTAMBLE allows inserting a certain number of ‘01’ or ‘10’ bit pairs at the end of the data packet. The number of POSTAMBLE pairs of bits can be set from 0 to 63 by the POSTAMBLE_LENGTH bit field of the *PCKT_CONFIG register (PCKT_CONFIG)*. The type of POSTAMBLE bit sequence (‘01’ or ‘10’) is selected by the POSTAMBLE_SEQ bit field of the PCKT_CONFIG register according on the following values:

- 00: POSTAMBLE sequence = 01 01 01 01 01 ……
- 01: POSTAMBLE sequence = 10 10 10 10 10 ……
- 10: POSTAMBLE sequence depends on last payload/CRC bit:
  - last payload/CRC bit is 0: POSTAMBLE sequence = ‘10’
  - last payload/CRC bit is 0: POSTAMBLE sequence = ‘01’

The POSTAMBLE configuration concerns only the transmission to add a POSTAMBLE at the end of the frame (as required by some protocols like the W-MBUS). The POSTAMBLE is not considered when in reception (the receiver stops the reception after the CRC without considering additional postamble bits to be received).

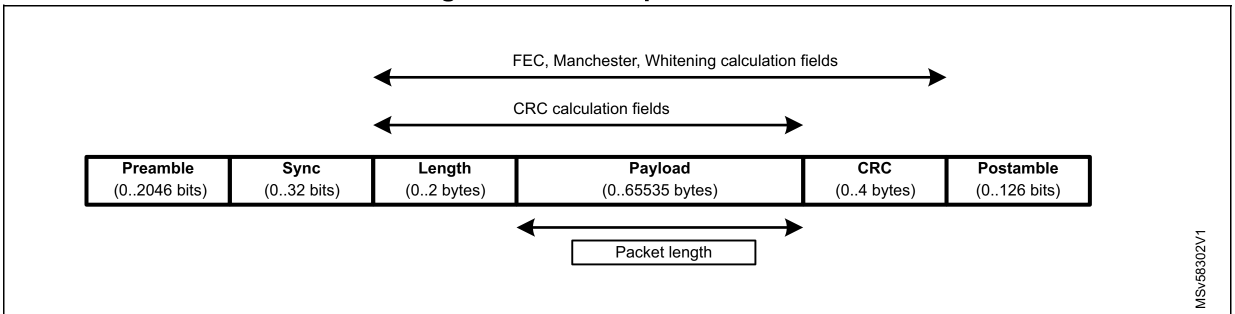
Packet formats

At HW level, the MR_SubG IP supports 2 sorts of packet:

Basic packet

The Basic format packet can be used to build any home-made protocol by playing with all options of configuration for all bit fields.

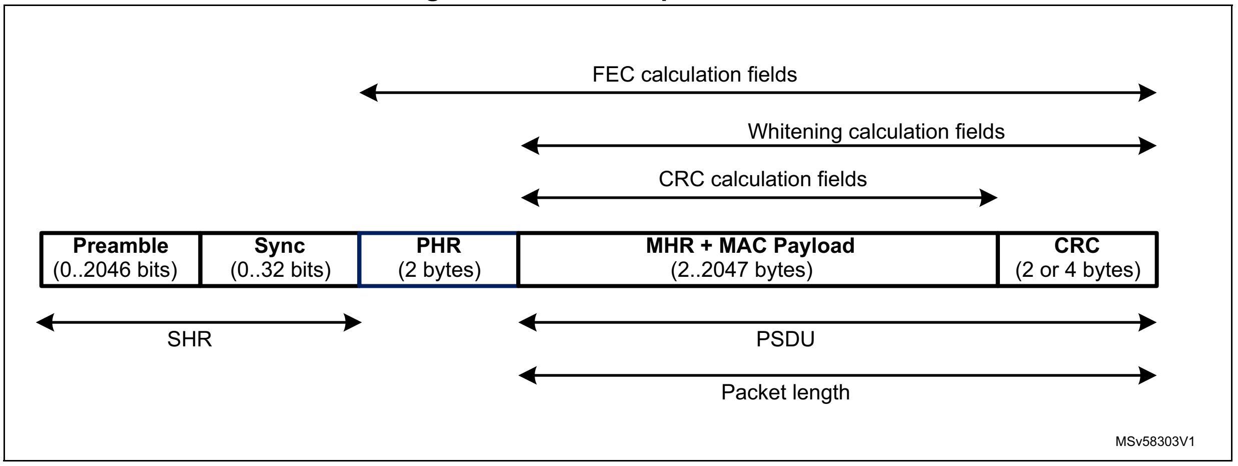
Figure 268. Basic packet format



802.15.4 packet

The 802.15.4 packet format allows activating few additional HW treatment / options to fit the 802.15.4 standard.

Figure 269. 802.15.4 packet format



The PREAMBLE and SYNC word (primary and secondary) are managed according to the programmed configuration.

Refer to section *802.15.4* for details on registers configurations to manage an 802.15.4 standard RF transfer with the MR_SubG IP.

PHR: The PHR (physical header) field is specific for the 802.15.4g packet format and is automatically built by the packet handler block based on current register configuration.

Table 108. PHR frame

Bit string index	0	1-2	3	4	5-15
Bit mapping	MS	R_1-R_0	FCS	DW	L_{10}-L_0
Field name	Mode Switch	Reserved	FCS type	Data Whitening	Frame Length

In particular:

- MS is always set to 0b (mode switch not supported)
- R_1-R_0 are always set to 00b
- FCS is set to:
  - 1b if CRC mode 3 is selected
  - 0b if CRC mode 5 is selected
- DW is set to:
  - 0b if whitening is disabled (WHIT_EN bit = 0 in the *PCKT_CTRL register (PCKT_CTRL)*)
  - 1b if whitening is enabled (WHIT_EN bit = 1 in the *PCKT_CTRL register (PCKT_CTRL)*)
- L_{10}-L_0:
  - in Transmission: represents the PCKTLEN[11:0] bit field of the *PCKTLEN_CONFIG register (PCKTLEN_CONFIG)*
  - in Reception: decoded on the fly to know the packet length and stored in the RX_PCKTLEN_OUT[15:0] bit field in the RX_INFO_REG status register.

Note: The packet length represents MHR + MAC Payload + CRC.

The CRC is calculated on MHR + MAC Payload bit fields and appended to the end of the payload in transmission or calculated and compared to the CRC part of the frame in reception.

As mentioned previously, the CRC is included in the length of the packet. For this reason:

- If CRC mode 3 (2-bytes):
  - the MR_SubG IP uses packet length – 2 bytes from/to the RAM Data Buffers. The 2-bytes CRC are built on the fly and appended to the frame in transmission and decoded/checked in reception but not written into RAM)
- If CRC mode 5 (4-bytes):
  - the MR_SubG IP uses packet length – 4 bytes from/to the RAM Data Buffers. The 4-bytes CRC are built on the fly and appended to the frame in transmission and decoded/checked in reception but not written into RAM)
  - if the payload length is less than 4 bytes, the payload is zero-padded to reach a minimum length of 4-bytes. The padded bits are internally used to compute the CRC but not transmitted on the air. The reverse operation is performed on receiver side.
- If CRC mode 0 (no CRC): the MR_SubG IP uses packet length bytes from/to the RAM Data Buffers. In this case no CRC calculation nor management is done at HW level and it is the responsibility of MAC layer to process it.

29.3.2 Sequence transformation and data coding

The MR_SubG IP embeds several options that impact the payload format.

Some options modify the sequence to help building a home-made protocol (byte swap) or to make the transfer more robust (whitening).

Other options are linked to coding algorithms. The MR_SubG IP supports 3 exclusive coding options.

Whitening

The data whitening algorithm aims at preventing repeating runs of 0's or 1's on the antenna, which would create spectral lines which may complicate symbol tracking at the receiver or interfere with other transmissions.

Data whitening is implemented with a maximum length LFSR generating a pseudo-random binary sequence used to XOR data before entering the encoding chain. The length of the LSFR is set to 9 bits.

The pseudo-random sequence is initialized with the value present in the WHIT_INIT[8:0] bit field of the *PCKT_CTRL register (PCKT_CTRL)*.

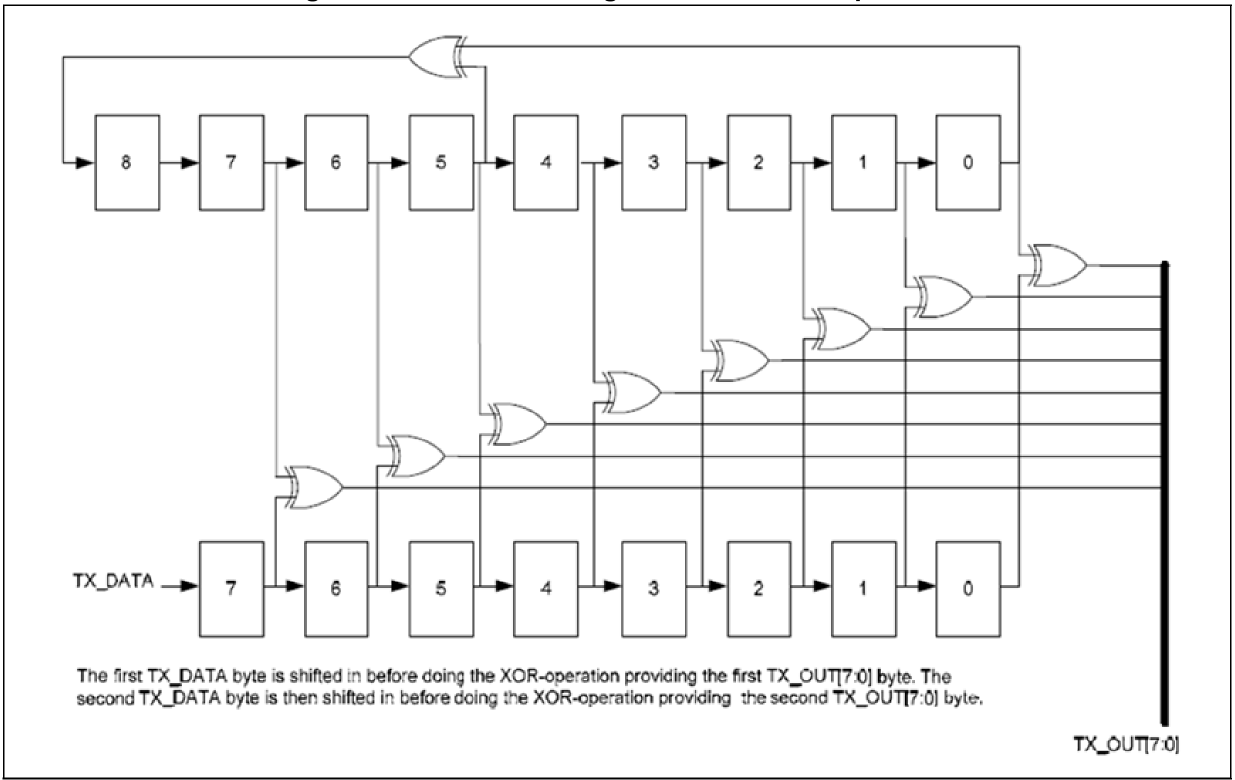
When enabled through WHIT_EN bit in the PCKT_CTRL register, the data are scrambled before being transmitted in such a way that long sequences of zeros or ones become very unlikely and physical layer algorithms perform better.

At the receiver end, the data are XOR-end with the same pseudo-random sequence. Whitening is applied according to the following LFSR implementation. Data whitening is always recommended.

For Basic packet format:

- the data whitening is applied on all fields excluding the PREAMBLE, the SYNC word and the POSTAMBLE
- The data whitening is enabled / disabled through the WHIT_EN bit for both transmission and reception
- the scheme is the following:

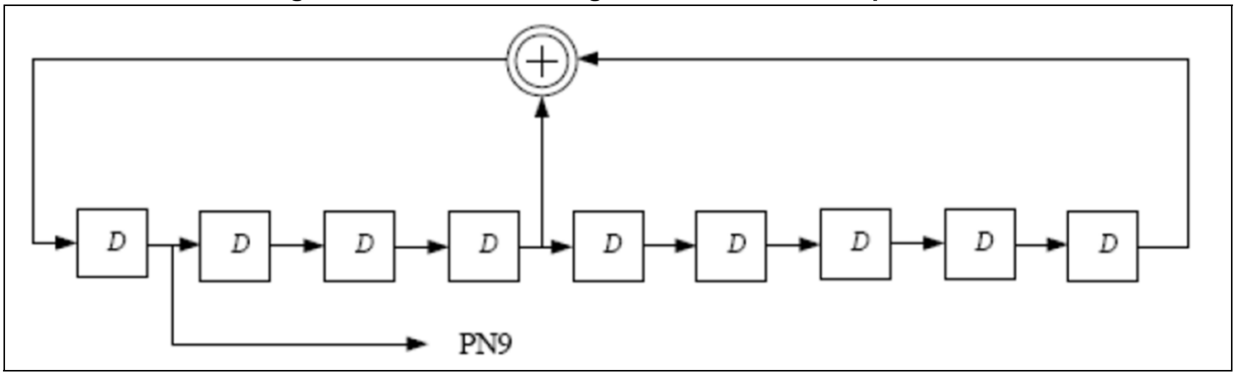
Figure 270. Data whitening scheme for Basic packets



For 802.15.4 packet format:

- the data whitening is applied to all fields following the PHR field
- the data whitening is enabled through the WHIT_EN bit in transmission but decoded on inside the PHR field of the received frame in reception
- the scheme is the following:

Figure 271. Data whitening scheme for 802.15.4 packets



Byte swapping

This feature allows swapping the MSbit-LSbit endianness inside the data to transmit.

This feature when enabled applies only on the PAYLOAD and LENGTH (when VARIABLE length mode is used) part of the frame. PREAMBLE, SYNC, CRC and POSTAMBLE are not impacted by the swapping action.

Note: Msbit-Lsbit inversion is performed on a single byte basis for the PAYLOAD, while it is performed on a full-field basis for the LENGTH. This means the swap transmits the 16 bit value LENGTH[15]...LENGTH[0] as LENGTH[0]..LENGTH[15] when the LENGTH field is 2 bytes long.

The feature is managed through the BYTE_SWAP bit in the *PCKT_CTRL register (PCKT_CTRL)*:

- 0: byte swap is disabled, bytes are transmitted MSB first
- 1: byte swap is enabled, bytes are transmitted LSB first

Note: This feature is dedicated to Basic format packet and MUST NOT be enabled for the configuration that selects the 802.15.4 format packet.

Coding

The MR_SubG IP supports several types of coding that can be selected through the CODING_SEL[1:0] bit field in the *PCKT_CTRL register (PCKT_CTRL)* (static register):

- 00: none
- 01: convolutional encoding (FEC) in TX / Viterbi decoder in RX
- 10: 3-out-of-6 coding (applies only on payload)
- 11: Manchester coding / decoding

Note: As those coding types are exclusive, the bit field to program for selection allows enabling only one at a time.

FEC coding

The MR_SubG IP provides hardware support for error correction and detection.

The error correction can be enabled or disabled according to the link reliability and power consumption needs.

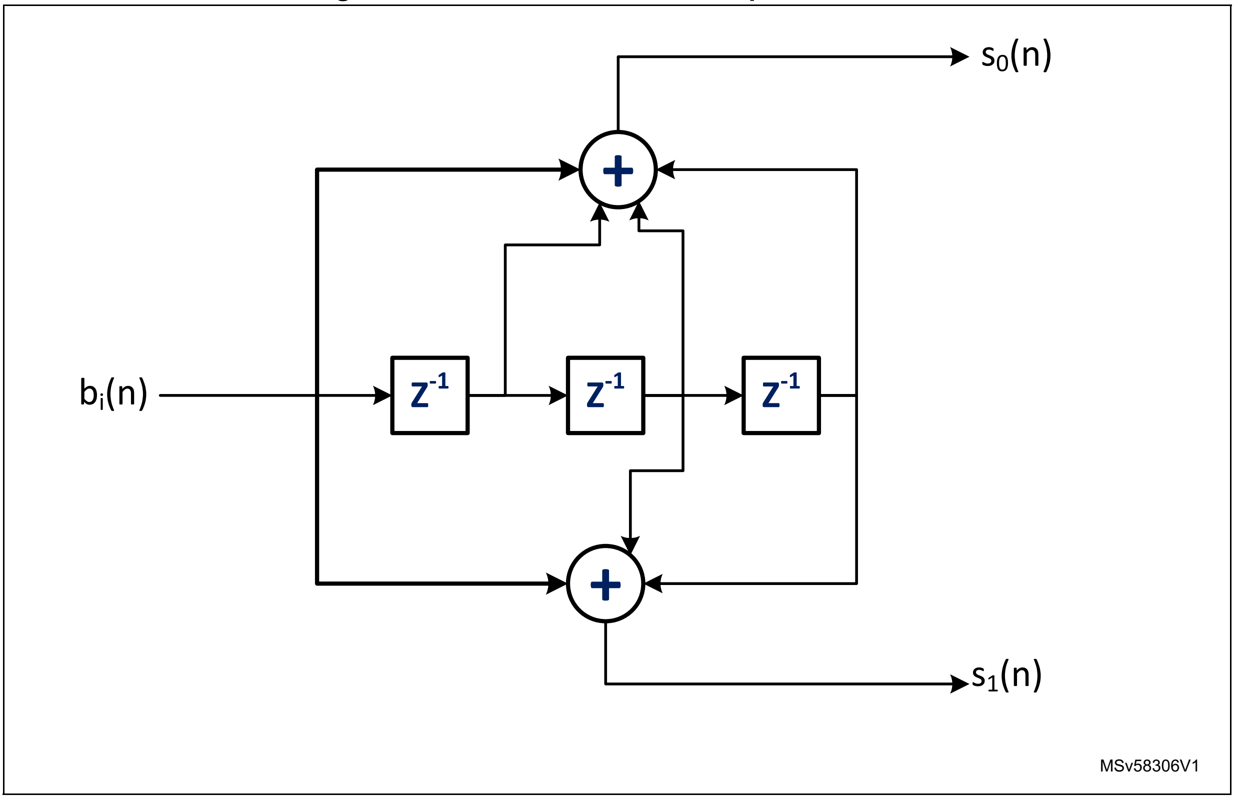
Convolution coding (rate 1/2) and interleaving can optionally be applied to the data.

FEC / Viterbi can be enabled by setting “01” in the CODING_SEL[1:0] bit field of the PCKT_CTRL register. When FEC is enabled the number of transmitted bits is roughly doubled hence the on-air packet duration in time is also roughly doubled.

Convolutional coding for Basic packet format:

- the FEC is applied to LENGTH (when present), PAYLOAD and CRC (when present). The PREAMBLE, the SYNC word and the POSTAMBLE are excluded.
- the convolutional coding scheme is the following:

Figure 272. FEC scheme for Basic packet format



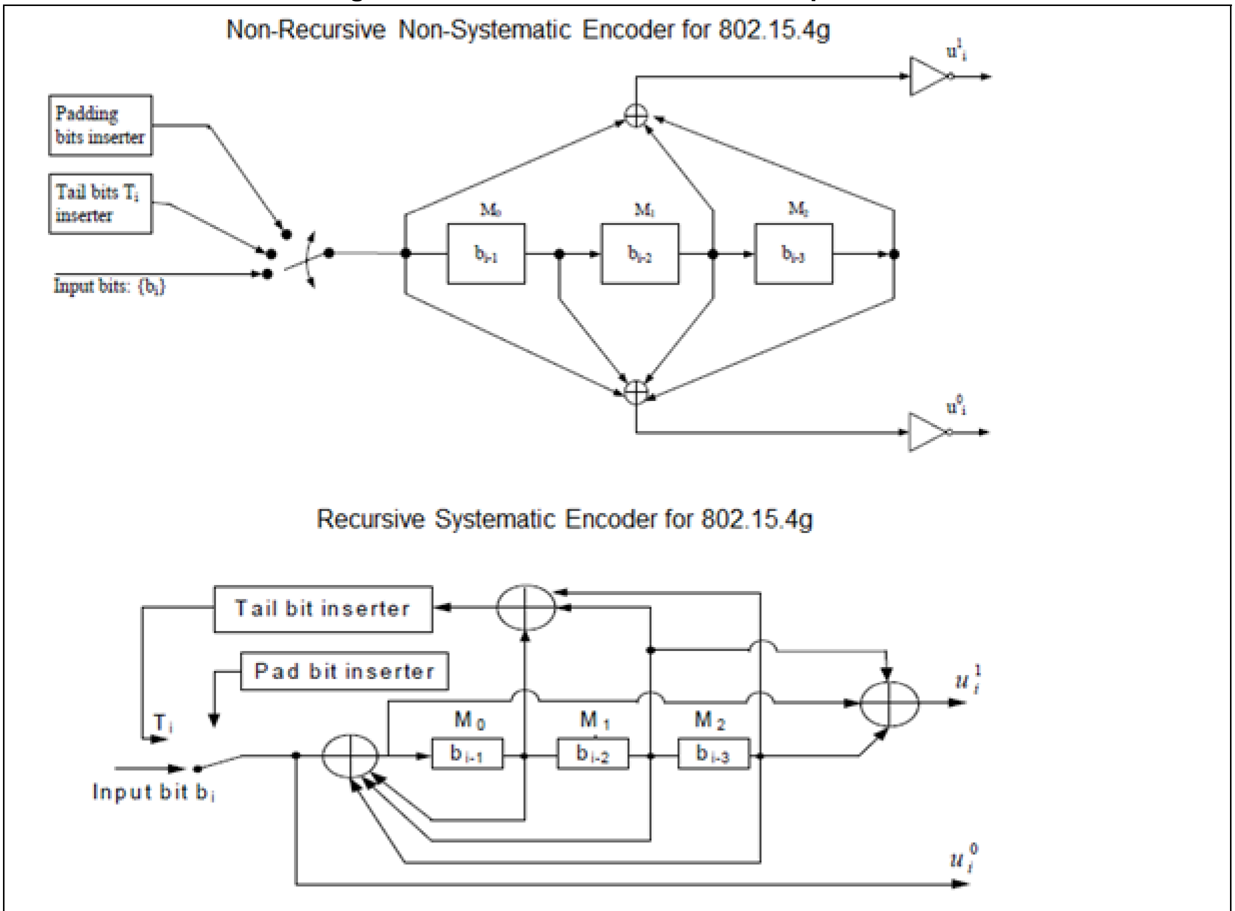
Convolutional coding for 802.15.4 packet format:

- the FEC is applied to all fields except the PREAMBLE and the SYNC word
- two different coding schemes can be selected depending on the FEC_TYPE_4G bit value in the PCKT_CTRL register (shown in *Figure 273: FEC schemes for 802.15.4 packets*):
  – Non-Recursive Non-Systematic Coding (NRNSC) when FEC_TYPE_4G = 0

Note: The NRNSC encoder for the 802.15.4 packet is the same as the one used for Basic packet with logical inversion of the output symbols.

   – Recursive Systematic Coding (RSC) when FEC_TYPE_4G = 1
- when FEC is enabled:
   – in transmission: the secondary SYNC word is automatically transmitted
   – in reception: the detection of the SEC_SYNC word automatically enables the FEC coding regardless the CODING_SEL[1:0] bit field value (and regardless the SECONDARY_SYNC_SEL bit value)

Figure 273. FEC schemes for 802.15.4 packets



Interleaving:

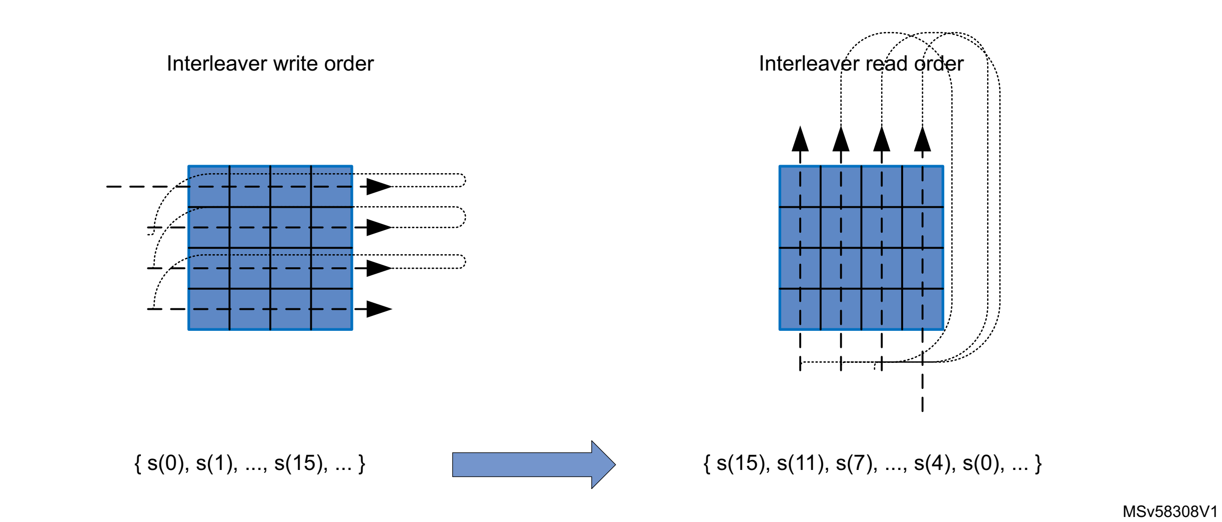
In order to improve the effectiveness of the convolutional encoding, an interleaving matrix can be applied to the encoded data at the output of the convolutional encoder.

As shown on the figure below, the symbols from the output of the encoder are:

- written raw-wise into a 4 x 4 matrix buffer starting from the upper-left cell
- read column-wise starting from the lower-right cell

Each pair of encoded symbols (corresponding to one single encoded bit) is packed into a single matrix cell. For each encoded symbol pair, $s_0(n)$ is transmitted first on the air and $s_1(n)$ is transmitted second.

Figure 274. Interleave scheme



Note: For Basic packet: interleave is always enabled together with FEC

for 802.15.4 packet: interleave is enabled / disabled using the INT_EN_4G bit in the PCKT_CTRL register (PCKT_CTRL).

Manchester coding

The Manchester coding can be enabled by setting "11" in the CODING_SEL[1:0] bit field of the PCKT_CTRL register.

Note: The SW is not supposed to select this coding:

For the 802.15.4 packets (planned to work only with Basic packet format).

For 4-(G)FSK modulation type.

The Manchester coding type is configurable through the MANCHESTER_TYPE bit in the PCKT_CTRL register:

- MANCHESTER_TYPE = 0:
  - bit '1' is transmitted as '10"
  - bit '0' is transmitted as '01"
- MANCHESTER_TYPE = 1:
  - bit '1' is transmitted as '01"
  - bit '0' is transmitted as '10"

The Manchester encoding is applied to LENGTH (when present), PAYLOAD and CRC (when present). The PREAMBLE, the SYNC word and the POSTAMBLE are excluded.

3 out of 6 coding

The three-out-of-six coding can be selected by setting "10" in the CODING_SEL[1:0] bit field of the PCKT_CTRL register.

Note: The SW is not supposed to select this coding for the 802.15.4 packets. It is planned to work only with Basic packet format for compatibility with W-MBUS protocol.

The 3oo6 cannot be used in parallel of whitening HW option.

Note: From cut 2:
- *the 3oo6 coding can be used with 4-(G)FSK*
- *the 3oo6 coding applies on LENGTH + PAYLOAD + CRC instead of PAYLOAD only which means it can be used whatever the CRC_MODE and FIX_VAR_LEN bit fields values.*

The coding is done according to *Table 109*:

Table 109. 3-out-of-6 coding table

4-bit not coded pattern		6-bit code		Number of transitions
In hexadecimal	In decimal	In hexadecimal	In decimal	
0000	0	010110	22	4
0001	1	001101	13	3
0010	2	001110	14	2
0011	3	001011	11	3
0100	4	011100	28	2
0101	5	011001	25	3
0110	6	011010	26	4
0111	7	010011	19	3
1000	8	101100	44	3
1001	9	100101	37	4
1010	10	100110	38	2
1011	11	100011	35	2
1100	12	110100	52	3
1101	13	110001	49	2
1110	14	110010	50	3
1111	15	101001	41	4

29.3.3 Protocols

Frequency modulations

The user can select the wanted modulation through the MOD0_CONFIG[22:20] = MOD_TYP[2:0] bit field:
- MOD_TYP = 000: 2-FSK
- MOD_TYP = 001: 4-FSK
- MOD_TYP = 010: 2-GFSK
- MOD_TYP = 011: 4-GFSK

Figure 275. Transmission path for 2-(G)FSK modulation

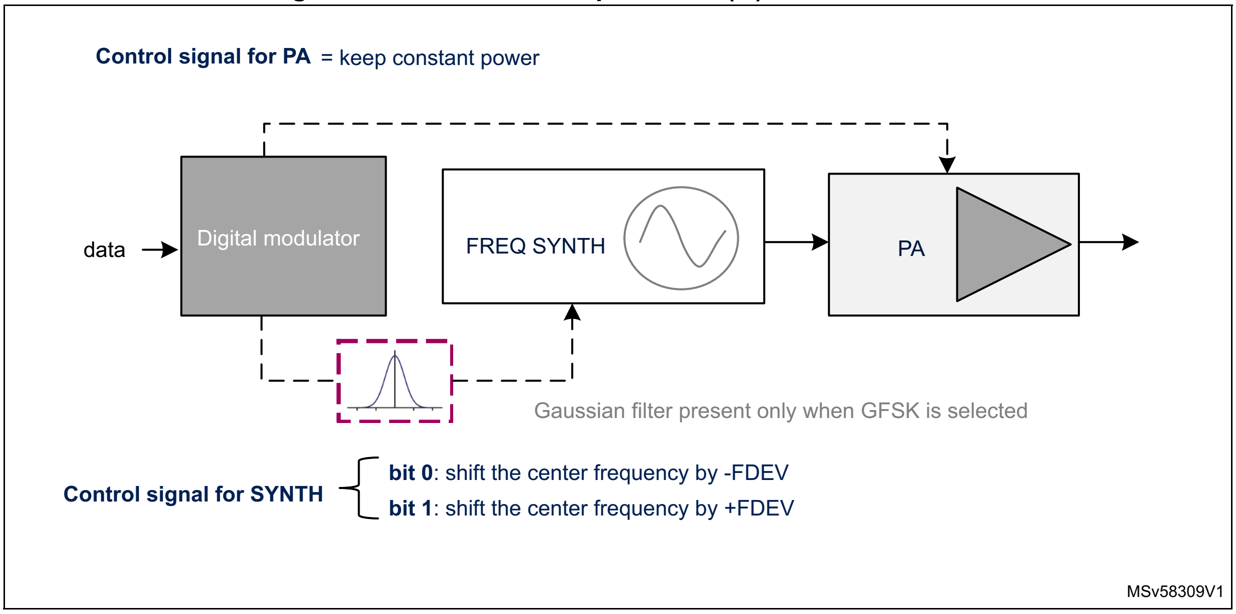
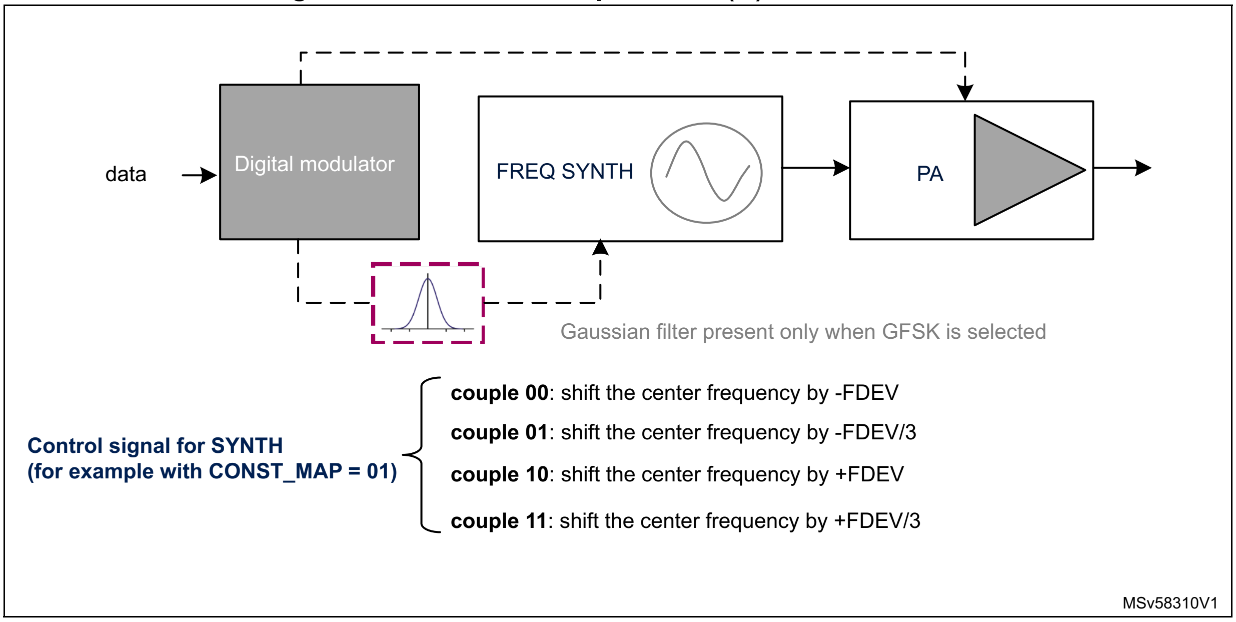


Figure 276. Transmission path for 4-(G)FSK modulation



In transmission the user must configure:

- PA_MODE = 00 or 10 (in the *PA_CONFIG register (PA_CONFIG)* register)
- (Recommended) PA_CONFIG[14] = PA_RAMP_EN to have a power ramp-up at the beginning of the Tx and a PA ramp down at the end of the TX

Note: To achieve reliable performance in reception, the PREAMBLE and the SYNC word should always be mapped on outer constellation points (+FDEV / -FDEV). Refer to the Symbol mapping paragraph concerning SYNC word additional option.

Frequency deviation

For frequency modulation 2-(G)FSK and 4-(G)FSK, the frequency deviation can be tuned in wide range that depends on the f_{xo} (XTAL frequency) according the following formula:

Equation 1 - Frequency deviation in (G)FSK modulation

$$f_{dev} = \frac{f_{xo}}{2^{19}} \times \frac{\mathrm{round}\left(FDEV\_M \times \frac{B}{8}\right)}{B} \quad \text{if FDEV\_E = 0}$$

$$f_{dev} = \frac{f_{xo}}{2^{19}} \times \frac{\mathrm{round}\left((256 + FDEV\_M) \times 2^{(FDEV\_E - 1)} \times \frac{B}{8}\right)}{B} \quad \text{if FDEV\_E > 0}$$

with:

B:

- 4 for the high band (corresponding to SYNTH_FREQ[30] = BS = 0)
- 8 for the low band (corresponding to SYNTH_FREQ[30] = BS = 1)

FDEV_E and FDEV_M are bit fields of the dynamic *MOD1_CONFIG register (MOD1_CONFIG)*.

Note: Recommendation for FDEV range (to provide a rough rule to help the user):

- *for data rate (DR) ≥ 3kbit/s, use FDEV = DR / 2*
- *for data rate (DR) < 3kbit/s, use FDEV = DR*

Symbol mapping

The frequency deviation programmed corresponds to the deviation of the outer constellation symbols.

For the 4-(G)FSK modulation, the deviation of the inner symbols is 1/3 and 4 options are available.

Furthermore, since the payload is normally arranged in bytes, the arrangement can change the mapping for both 2-(G)FSK and 4-(G)FSK modulations, by using the FOUR_GFSK_CONST_MAP (in the *MOD0_CONFIG register (MOD0_CONFIG)* register), in the following way:

Table 110. Constellation mapping for 2-(G)FSK

Format	Symbol	FOUR_GFSK_CONST_MAP			
		0	1	2	3
2-(G)FSK	0	-FDEV	-FDEV	+FDEV	+FDEV
	1	+FDEV	+FDEV	-FDEV	-FDEV

Table 111. Constellation mapping for 4-(G)FSK

Format	Symbol	FOUR_GFSK_CONST_MAP			
		0	1	2	3
4-(G)FSK	00	-FDEV/3	-FDEV	+FDEV/3	+FDEV
	01	-FDEV	-FDEV/3	+FDEV	+FDEV/3
	10	+FDEV/3	+FDEV	-FDEV/3	-FDEV
	11	+FDEV	+FDEV/3	-FDEV	-FDEV/3

This mapping is not optimal since the resulting SYNC word is very short and this may cause several false detection problems. For this reason, a specific control bit allows selecting a 2-(G)FSK like treatment on SYNC word mapping by using only +/-Fdev, keeping the rest of the frame in 4-(G)FSK mapping. This option is activated by setting the PCKT_CTRL[31] = FORCE_2FSK_SYNC_WORD bit.

Example: a 16-bit SYNC = 0x8888 is output by the transmitter (and decoded by the receiver) as 11010101 11010101 11010101 11010101 when the option is active and the selected constellation mapping is 0.

Furthermore, an option to swap the symbols is available for the 4-(G)FSK modulation using the FOUR_FSK_SYM_SWAP bit in the *PCKT_CTRL register (PCKT_CTRL)* as follows:

- FOUR_FSK_SYM_SWAP = 0:
  - $S_0 = <b_7b_6>$
  - $S_1 = <b_5b_4>$
  - $S_2 = <b_3b_2>$
  - $S_3 = <b_1b_0>$
- FOUR_FSK_SYM_SWAP = 1:
  - $S_0 = <b_6b7>$
  - $S_1 = <b_4b_5>$
  - $S_2 = <b_2b_3>$
  - $S_3 = <b_0b_1>$

Note: The symbol swapping applies only on the PAYLOAD of the frame (PREAMBLE, SYNC, LENGTH when present, CRC and POSTAMBLE are not swapped).

Gaussian filter

In 2-GFSK or 4-GFSK mode, the Gaussian filter BT product can be set by using the register BT_SEL to 1 or 0.5, using the BT_SEL bit in the dynamic *MOD0_CONFIG register (MOD0_CONFIG)*.

The Gaussian filtering is implemented by poly-phase filtering with eight taps per symbol time. In order to further smooth the filter shape and improve spectral shaping, the output of the filter can be linearly interpolated by setting the register MOD_INTERP_EN in the *PCKT_CTRL register (PCKT_CTRL)*.

The mathematical interpolation factor applied to each sample of the Gaussian filter output depends on the DATARATE_E[3:0] bit field in the *MOD0_CONFIG register (MOD0_CONFIG)*:

- DATARATE_E ≤ 5: x 64,
- 6 ≤ DATARATE_E ≤ 10: automatically scaled as
- 11 ≤ DATARATE_E: interpolation is disabled

Note: The actual interpolation factor achieved may be limited by the minimal frequency resolution of the frequency synthesizer.

ISI cancellation for 4-(G)FSK

Since the 4-(G)FSK modulation format strongly suffers from the effect of inter symbol interference, an ISI cancellation equalizer has been introduced in the demodulator. An equalizer can be enabled, by using AS_EQU_CTRL[1:0] bit field in the *AS_QI_CTRL register (AS_QI_CTRL)* register, with two modes: single pass equalization and dual pass equalization. The best performance is normally achieved using the dual pass equalizer.

Direct Sequence Spectrum Spreading (DSSS)

The Direct Sequence Spectrum Spreading aims at reducing overall signal interference. The principle is to make the transmitted signal bandwidth wider than the information bandwidth. The information bandwidth is restored in the receiver while the unintentional and intentional interference is reduced.

The MR_SubG IP embeds a DSSS mode with several spread factors and associated code sequences as described in the table below. This mode can be used only for the 2-(G)FSK transfers.

Note: The DSSS must not be activated with another coding feature (managed through PCKT_CTRL[22:21] = CODING_SEL[1:0]).

Table 112. DSSS codes

Code	Binary sequence	Spread Factor (SF)	Spread exponent	Correlation gain
0x03	0011	4	2	4
0x8D	1000 1101	8	3	8
0x6BC4	0110 1011 1100 0100	16	4	16
0x0A60E457B	1010 0110 0000 1110 0100 0101 0111 1011	32	5	32
0x540FBCEB0B8DA44D	0101 0100 0000 1111 1011 1100 1110 1011 0000 1011 1000 1101 1010 0100 0100 1101	64	6	56

For each data bit, the complete spreading code is used to multiple (XORed) the data, so for each data bit, the spread or expanded signal consists of n bits where n is the spreading factor (the n-bit signal is called chip).

In transmission, the spreading is done LSB first regarding the binary sequence value in the table above. In reception, the dispreading is done MSB first.

Example of transmission using SF=8 with data rate programmed to be 38.4ksps:

- ‘0’ is sent on the air as 1-0-1-1-0-0-0-1
- ‘1’ is sent on the air as 0-1-0-0-1-1-1-0
- each binary sequence coded bit is sent every 26.04 µs (38.4 kbit/s rate) so one bit (before coding) is transmitted in 8 x 26.04 µs which means the original message is transmitted with DR = programmed data rate and SF= spread factor

Note: The chip coding sequence is not standard: when the DSSS mode is enabled on a transmission on the MR_SubG IP, the user needs a MR_SubG IP to be able to decode it on receiver side.

The feature can be enabled through the DSSS_EN bit in the Global static DSSS_CTRL register.

On transmitter side, when DSSS_CTRL[7] = DSSS_EN = 1:

- the only programmable parameter is the Spreading Factor through DSSS_CTRL[6:4] = SPREADING_EXP[2:0] bit field

On receiver side, when DSSS_CTRL[7] = DSSS_EN = 1, several parameters must be programmed:

- the Spreading Factor through DSSS_CTRL[6:4] = SPREADING_EXP[2:0] bit field
- the Acquisition window through DSSS_CTRL[3:0] = ACQ_WINDOW[3:0] bit field
- the Acquisition threshold through DSSS_CTRL[15:10] = ACQ_THR[5:0] bit field
- the Acquisition hits through DSSS_CTRL[9:8] = ACQ_HITS[1:0] bit field

During the acquisition process the received signal preamble is correlated with the expected spreading sequence to determine the correct alignment between the received signal and the local sequence.

The two sequences are considered aligned when the correlation is strictly above a programmable threshold. Since the maximum correlation is equal to the spreading factor, the value of ACQ_THR[5:0] must be ≤ SF-1.

Setting ACQ_THR[5:0] at a lower value (for example, SF-3) may help the acquisition process in case of very low signal-to-noise ratio allowing for some chip errors in the correlation.

Two additional parameters are used to confirm the acquisition: ACQ_HITS[1:0] and ACQ_WINDOW[3:0]; in particular, in order to declare the acquisition valid, there must be ACQ_HITS[1:0]+1 valid correlation peaks in an observation window of (ACQ_WINDOW[3:0] + 1) x SF received chips

Amplitude modulation (ASK/OOK)

The user can select the wanted modulation through the MOD0_CONFIG[22:20] = MOD_TYP[2:0] bit field:

- MOD_TYP = 101:ASK/OOK

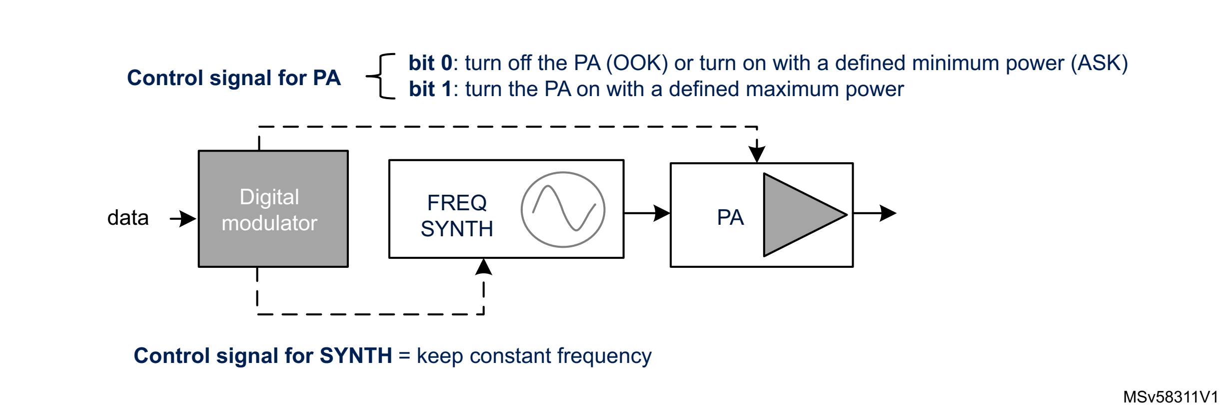
The ASK or OOK selection (at transmission level) depends on the value chosen for the PA_LEVEL0 bit field:

- if PA_LEVEL0[7:0] = 0, then OOK modulation
- if PA_LEVEL0[7:0] > 0, then ASK modulation

The 0/1 mapping can be reversed by setting the MOD0_CONFIG[25:24] = FOUR_GFSK_CONST_MAP bit field to any value other than zero.

For transmission

Figure 277. Transmission path for ASK/OOK modulation



The analog power amplifier embeds a filter with a cut-off frequency that can be programmed depending on the data rate through the PA_FC[1:0] bit field in the *ADDITIONAL_CTRL register (ADDITIONAL_CTRL)*.

Table 113. PA Bessel filter programming

PA_FC[1:0]	Cut-Off frequency (kHz)	Max Data Rate (kbit/s)
00	12.5	16
01	25	32
10	50	62.5
11	100	125

PA_MODE = 01

The PA controller can be programmed in FIR mode. This mode has been created especially for the ASK/OOK modulation in order to reduce bandwidth occupancy.

- set the PA_CONFIG[7] = ASK_OOK_EN
- program the PA_LEVEL0[7:0] and the PA_LEVEL7[7:0] bit fields to indicate respectively power value for symbol = 0 and for symbol = 1
- ensure the PA_CONFIG[6] = PA_INTERP_EN = 0
- (recommended) set the PA_CONFIG[14] = PA_RAMP_EN to activate the power ramp down feature (adds one 0 symbol at the end of the transmission to manage a power ramp down)

The FIR can be programmed in 3 different configurations through the *PA_REG register (PA_REG)*:

- filtering (default): filtering mode between PA_LEVEL0 and PA_LEVEL7
- ramping: ramping from “internal computed level1” to PA_LEVEL7 by repeating twice “internal computed level1” and PA_LEVEL7
- switching: direct switching between PA_LEVEL0 and PA_LEVEL7 (no smooth transition)

PA_MODE = 00 or 10

The PA controller can be programmed in Spirit1 Legacy mode (00) or in Linear mode (10). The only difference between those 2 options is the unit to program the PA power steps:

- PA_MODE = 00 (Spirit1 Legacy): the PA power values associated to each step must be entered in dBm unit using a value between 0 and 0x51 using the PA_LEVELx[7:0] bit fields (0x51 is the max power, then each -1 corresponds to -0.5 dB)
- PA_MODE = 10 (Linear mode): the PA power values associated to each step must be entered in linear unit using a value between 0 and 0xFF using the PA_LEVELx[7:0] bit fields

Then for those 2 modes:

- (Recommended) PA_CONFIG[14] = PA_RAMP_EN to have a power ramp-up at the beginning of the Tx and a PA ramp down at the end of the TX
- (Recommended) PA_CONFIG[6] = PA_INTERP_EN to improve the spectral emission mask. When set, the power values specified in the PA_LEVELx bit fields are linearly interpolated by the modulator before being applied to the PA. See the note below.
- Configure the intermediary step of power ramping through *PA_LEVEL_3_0 register (PA_LEVEL_3_0)* and *PA_LEVEL_7_4 register (PA_LEVEL_7_4)*. It is also possible to tune:
  - the number of steps can be customized through the PA_CONFIG[4:2] = PA_LEVEL_MAX_INDEX bit field
  - the step width through the PA_CONFIG[1:0] = PA_RAMP_STEP_WIDTH bit field.

Note: The interpolation factor of each ramp step is:

- *64 when MOD0_CONFIG[19:16] = DATARATE_E < 5,*
- *automatically scaled to 64 / $2^{(DATARATE\_E - 5)}$ when 5 ≤ MOD0_CONFIG[19:16] = DATARATE_E < 11*
- *automatically disabled when MOD0_CONFIG[19:16] = DATARATE_E ≥ 11*

For reception

The OOK/ASK demodulation is controlled by the OOK_PEAK_DECAY parameter in the *RSSI_FLT register (RSSI_FLT)* radio register. This parameter controls the speed at which the measured signal peak decays, the recommended value is 3.

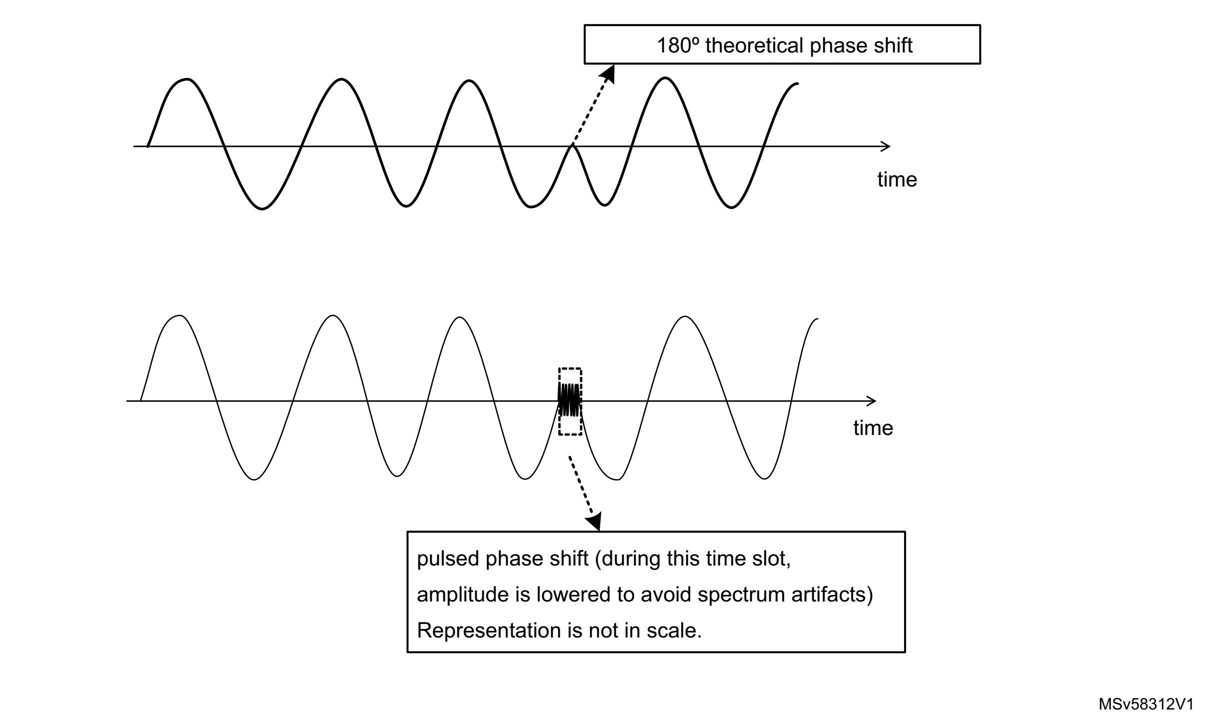
Direct Polar TX mode

The user can select the Direct Polar TX modulation through the MOD0_CONFIG[22:20] = MOD_TYP[2:0] bit field:

- MOD_TYP = 110: Polar TX

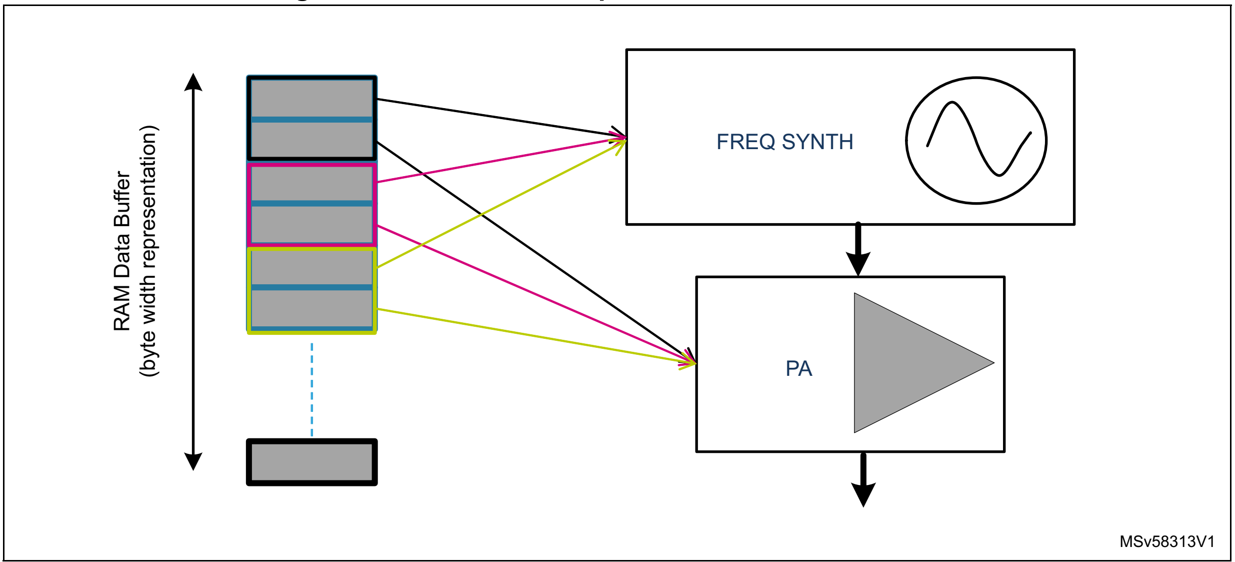
This function is suitable to implement differential binary phase shift keying modulation (DBPSK) as shown in the figure below:

Figure 278. DBPSK modulation emulation through Polar TX mode



For this reason, this mode allows driving the SYNTH and the PA at a very low level.

Figure 279. Transmission path for Polar TX modulation



The modulator samples byte couples from the RAM Data Buffer with a rate related to the programmed Data Rate (sampling rate = 8 x DATARATE).

The first byte of the RAM Data Buffer couple drives the frequency synthesizer to obtain an instantaneous output frequency deviation.

Equation 2 - Frequency deviation in polar mode

$$f_{dev} = f_{dev\_programmed} \times \frac{f_{dev\_fifo\_sample}}{128}$$

with:

- $f_{dev\_programmed}$ = frequency deviation programmed through FDEV_M[7:0] and FDEV_E[3:0] bit field of the *MOD1_CONFIG register (MOD1_CONFIG)* register.
- $f_{dev\_fifo\_sample}$ = first byte of the couple sampled in the RAM data buffer. The $f_{dev\_fifo\_sample}$ is interpreted as a 2-complement 8-bit number, thus it can be either a positive or a negative value.

Equation 3 - Instantaneous frequency in polar mode

$$f = f_{c\_programmed} + f_{dev}$$

The second byte of the RAM Data Buffer couple drives the PA giving an instantaneous output power. The output power is generated according to the PA_LEVELx values following the same codes as for other modulations.

The user must program the PA_MODE to “00” or “10” (depending on wish to control PA in log scale or linear scale).

Test modes

Some additional modes are embedded in the MR_SubG IP to be used for test and validation purposes.

Continuous Wave

The device can be programmed to generate a continuous wave carrier without any modulation. In this way, the carrier is continuously transmitted until a SABORT command is sent to the device.

The user can select the CW mode through the MOD0_CONFIG[22:20] = MOD_TYP[2:0] bit field:

- MOD_TYP = 111: CW

Warning: this MOD_TYP programming must be associated with a “free-running” TX mode , that is, a TX mode running until a SABORT command is issued like for instance direct through GPIO, PN mode or direct through buffer with PCKTLEN = 0.

PN mode

PN mode TX mode is mainly intended to perform spectrum measurement in the lab without the need of an external data source. Refer to the description of PN mode in *Section 29.3.5: Transmission modes* for details.

Frequencies and data rates

The digital part digital and the analog Radio IP together support several bands of frequencies and a large window of data rates.

Frequency bands

Table 114. Frequency bands

Frequency bands (MHz)	VCO frequency (MHz)
413 - 479	3304 - 3822
826 - 958	
398 - 464	3184 - 3712
159.2 - 185.6	

The frequency carrier (f_c) is programmable through the *SYNTH_FREQ register (SYNTH_FREQ)* and the *ADDITIONAL_CTRL register (ADDITIONAL_CTRL)* register (dynamic registers) according to the following formula:

Equation 4 - Frequency carrier formula

$$f_c = f_{base} + \left(\frac{f_{ref}}{2^{16}} \cdot \text{CH\_SPACING[7:0]}\right) \times \text{CH\_NUM[7:0]}$$

with:

$$f_{base} = \frac{f_{ref}}{2^{20}} \times (\text{SYNTH\_INT[7:0]} \times 2^{20} + \text{SYNTH\_FRAC[19:0]}) \times \frac{1}{B}$$

where f_{ref} = XO frequency (48 or 50 MHz, nominal = 48 MHz)

and B =

4 for the high band (corresponding to BS = 0 in the *SYNTH_FREQ register (SYNTH_FREQ)*)

8 for the low band (corresponding to BS = 1 in the *SYNTH_FREQ register (SYNTH_FREQ)*)

20 for the 169 MHz band device (associated to BS = 0 in the *SYNTH_FREQ register (SYNTH_FREQ)*)

CH_SPACING[7:0] and CH_NUM[7:0] are bit field in the *ADDITIONAL_CTRL register (ADDITIONAL_CTRL)*. Those 2 bit fields allow easy management of channel hopping around a targeted frequency carrier.

Data rates

The data rate specified in this section always applies to the on-the-air transmitted data (regardless coding selection).

The radio supports different data rates ranges depending on the chosen modulation as shown in the following table.

Table 115. Data rate range versus modulation

Modulation	Data rate (kbit/s)	
	Min	Max
2-(G)FSK	0.1	300
4-(G)FSK	0.2	600
ASK/OOK	0.1	125
TX Polar mode	0.1	100

(*): the data rate in this table is presented in kbits/s. So for 4-(G)FSK, it corresponds to a window from 0.1 to 300 ksymbols/s (unit of programming in the register).

The data rate is programmable through the dynamic *MOD0_CONFIG register (MOD0_CONFIG)* according to the following formula:

Equation 5 - Data rate programming formula (in ksps)

if DATA_RATE_E = 0

$$\text{data rate} = f_{sys} \times \frac{\text{DATA\_RATE\_M}}{2^{32}}$$

if DATA_RATE_E > 0

$$\text{data rate} = f_{sys} \times \frac{(2^{16} + \text{DATA\_RATE\_M}) \times 2^{\text{DATA\_RATE\_E}}}{2^{33}}$$

if DATA_RATE_E = 15 (jitter free mode)

$$\text{data rate} = \frac{f_{sys}}{8 \times \text{DATA\_RATE\_M}}$$

with f_{SYS}= digital fast clock provided by the SoC (16 MHz nominal)

RX channel filter bandwidth

The channel filter is the very last stage of the decimation filtering. The CHF is in charge to remove excess bandwidth to fine adjusting spectral occupation and optimize noise performance for the incoming demodulator block. The channel filter is to be intended as double-sided band.

The bandwidth of the receiver channel filter is programmable from 1.04 kHz to 1600 kHz. The configuration is programmed through the CHFLT_E[3:0] (represented by E in the table below) and CHFLT_M[3:0] (represented by M in the table below) bit fields of the dynamic *MOD1_CONFIG register (MOD1_CONFIG)* according to the following table.

Table 116. Channel filter bandwidth (in kHz) programming

M	E=0	E=1	E=2	E=3	E=4	E=5	E=6	E=7	E=8	E=9
M=0	1600	800	400	200	100	50	25	12.5	6.125	3.125
M=1	1510	755	377	188	94.4	47.2	23.6	11.8	5.91	2.94
M=2	1422	711	355	178	88.9	44.4	22.2	11.1	5.55	2.78
M=3	1332	666	333	166	83.3	41.6	20.8	10.4	5.2	2.6
M=4	1244	622	311	155	77.8	38.9	19.4	9.7	4.87	2.4
M=5	1154	577	288	144	72.2	36.1	18.1	9	4.5	2.2
M=6	1066	533	266	133	66.7	33.3	16.6	8.3	4.1	2.1
M=7	976	488	244	122	61.1	30.5	15.3	7.6	3.8	1.9
M=8	888	444	222	111	55.6	27.8	13.9	6.9	3.5	1.7
M=9	800	400	200	100	50	25	12.5	6.25	3.12	1.56
M=10	755.6	377.8	188.9	94.4	47.2	23.6	11.8	5.9	2.95	1.48
M=11	711.1	355.6	177.8	88.9	44.4	22.2	11.1	5.56	2.78	1.39
M=12	666.7	333.3	166.7	83.3	41.7	20.8	10.42	5.21	2.60	1.3
M=13	622.2	311.1	155.6	77.8	38.9	19.4	9.72	4.86	2.43	1.22
M=14	577.8	288.9	144.4	72.2	36.1	18.06	9.03	4.51	2.26	1.13
M=15	533.3	266.7	133.3	66.7	33.3	16.67	8.33	4.17	2.08	1.04
Output sample frequency (F_S) out of the complete chain (in kHz)										
-	2000	1000	500	250	125	62.5	31.25	15.625	7.8125	3.90625

The bandwidth values are intended as double-sided.

The actual filter bandwidth corresponds to a MR_SubG IP clock at 16 MHz. For any other digital clock frequency, f_{clk}, the channel filter bandwidth can be calculated by multiplying the values in the table above by the factor f_{clk}/16000000.

The channel filter bandwidth must be adapted to the targeted data rate (and FDEV for frequency modulations).

Guideline values for CHF (to provide a rough rule to help the user) are:

- in (G)FSK modulations: CHF = 2 x FDEV + DR (**in kbit/s**)
- in ASK/OOK modulation: CHF = 3 x DR **(in kbit/s)**

Intermediate frequency

The intermediate frequency (IF) can be programmed through the *IF_CTRL register (IF_CTRL)* register according to the following formula:

Equation 6 - Intermediate frequency formula

$$f_{IF} = \frac{f_{sys}}{2^{16}} \times IF\_OFFSET\_xx$$

with

_xx: _ANA or _DIG

f_{SYS}: digital fast clock provided by the SoC, supposed to be XO frequency / 3 (16 MHz nominal)

However, the user is supposed to limit the usage to 2 reference values depending on the channel filter bandwidth:

- IF = 300 kHz (recommended when CHF < 400 kHz)
- IF = 600 kHz (recommended when CHF > 400 kHz)

The register offers the possibility to program separately:

- the IF_OFFSET_ANA: used by the SYNTH interface block to generate the VCO frequency when in reception
- the IF_OFFSET_DIG: used by the demodulator

Note: The formula for IF_OFFSET_DIG is the same for any band.

However, this is planned for the AAF calibration operation and in the normal applicative usage of the device, both values are supposed to be aligned.

In addition, the *IF_CTRL register (IF_CTRL)* contains an IF_MODE bit used by the RFSUBG analog IP to select the AAF filter bandwidth:

- IF_MODE = 0: AAF bandwidth = 684 kHz -> to be selected when IF = 300 kHz
- IF_MODE = 1: AAF bandwidth = 1.55 MHz -> to be selected when IF = 600 kHz

29.3.4 Transmission and reception modes

The MR_SubG IP proposes several modes for transmission and reception. These modes are exclusive and are described in the following sections.

29.3.5 Transmission modes

The TX mode selection is done through the TX_MODE[1:0] bit field in the *PCKT_CTRL register (PCKT_CTRL)*.

The table below summarizes the available modes.

Table 117. TX mode summary

TX_MODE[1:0]	Mode name	Description	TX ending
00	Normal mode	Only payload is provided through RAM buffers Rest of the frame built from configuration registers (PREAMBLE, SYNC, CRC...)	Automatic
01	Direct through buffers	Full bit stream (including PREAMBLE, SYNC, CRC...) to be provided through RAM buffers	If PCKTLEN ≠ 0: Automatic (except if TX Polar modulation) If PCKTLEN = 0: SABORT command needed
10	Direct through GPIO	Full bit stream (including PREAMBLE, SYNC, CRC...) to be provided serially through the TX DATA GPIO	SABORT command needed
11	PN mode	Internal PN generator sends a polynomial bit stream on the antenna. Test purpose mode to perform spectrum measurement in the lab.	SABORT command needed

Normal mode

This mode is selected with TX_MODE[1:0] = “00”.

In Normal mode:

- the framer builds the PREAMBLE, the SYNC word, the LENGTH (if VARIABLE length mode) and the CRC using the information from the configuration registers and sends them on the air,
- the PAYLOAD is read from the RAM through the DBM block,
- the TX command stops automatically when all the elements of the frame have been transmitted.

Note: This mode is the only one supposed to be used with 802.15.4 packet format.

Even if PCKTLEN=0, the radio automatically prefetches 4 words in RAM (that won’t be used).

Direct through buffer mode

This mode is selected with TX_MODE[1:0] = “01”.

In Direct through buffer mode:

- the framer does not build any element of the frame sent on the antenna,
- all the bits sent on the air come from the RAM through the DBM block,
- the TX command:
  – stops automatically when the number of bytes sent correspond to the PCKTLEN bit field value if not null
  – must be stopped using a SABORT command when the PCKTLEN bit field is null

Note: If the chosen modulation is TX Polar, only a SABORT can stop the TX, whatever the PCKTLEN value.

In this mode, the SW is responsible to provide the full bit stream including a PREAMBLE, SYNC word, CRC, LENGTH (if to be added on the frame) in the RAM.

The direct modes intended to completely bypass the automatic packet handler, in order to give the user a maximum flexibility in the choice of frame formats. This allows building a homemade protocol that may not be achieved despite all the programmable options.

Direct through GPIO mode

This mode is selected with TX_MODE[1:0] = “10”.

In Direct through GPIO mode:

- the framer does not build any element of the frame sent on the antenna,
- all the bits sent on the air comes from an I/O of the device. The external world has to provide serially the bits to transmit on an I/O. In parallel an output TX clock signal is provided on another I/O of the device to indicate when to provide a new bit.
- data are sampled by the device on the rising edge of the provided TX clock; it is the responsibility of the external data source to provide a stable input on the rising edge.
- the TX command must be stopped using a SABORT command

In this mode, the SW is responsible to inject the full bit stream including a PREAMBLE, SYNC word, CRC, LENGTH (if needed in the frame) through the TX GPIO.

This mode allows a bypass of the automatic packet handler and offers a maximum flexibility to transmit a homemade protocol generated from an external processor / device.

PN mode

This mode is selected with TX_MODE[1:0] = “11”.

PN TX mode is mainly intended to perform spectrum measurement in the lab without the need of an external data source. When this mode is enabled, the framer block is bypassed and an internal PRBS9 or PRBS15 generator is used to feed the digital modulator.

In this mode, these data are continuously modulated until a SABORT command is sent to the device.

The selection of the PRBS polynomial is done through PN_SEL bit in the static *PCKT_CTRL register (PCKT_CTRL)*.

The polynomial generators are:

- for PN9: x^9 + x^5 + x^0 with the initial state equal to 0x1FF
- for PN15: x^15 + x^14 + 1 with initial state equal to 0x7FFF

29.3.6 Reception modes

The RX mode selection is done through the RX_MODE[2:0] bit field in the *PCKT_CTRL register (PCKT_CTRL)* register.

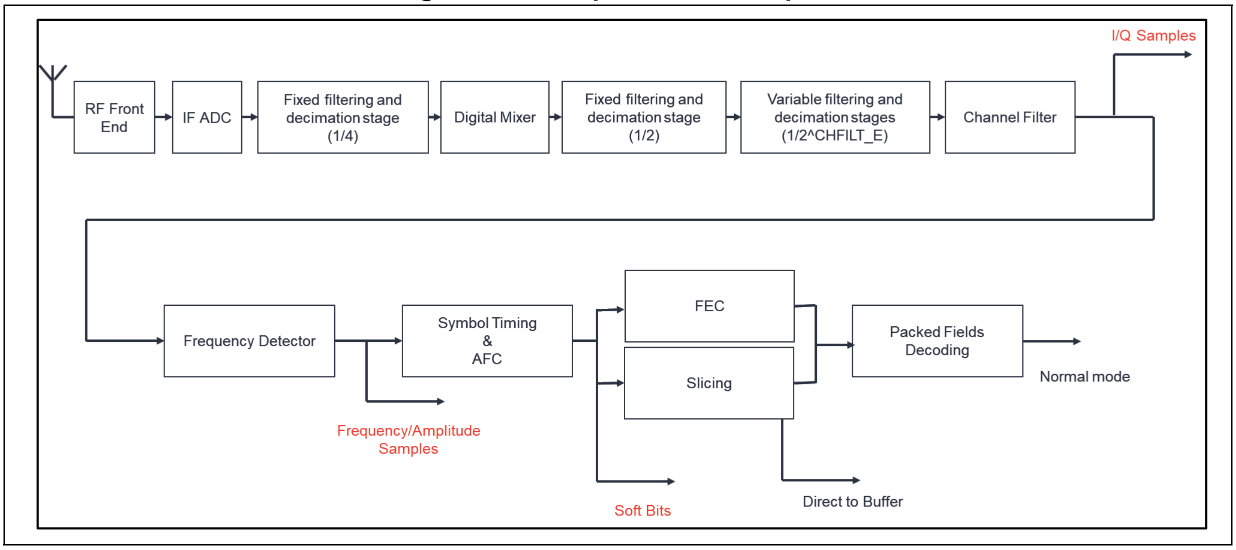
Table 118 summarizes the available RX modes.

Table 118. RX mode summary

RX_MODE[2:0]	Mode name	Description	RX ending
000	Normal mode	Only payload is stored into the RAM buffers. CRC and packet length are readable in dedicated status registers.	Automatic
001	Direct through buffers	Full bit stream (including noise when no signal on antenna, PREAMBLE, SYNC, CRC...) is stored into the RAM buffers.	SABORT command needed
010	Direct through GPIO	Full bit stream (including noise when no signal on antenna, PREAMBLE, SYNC, CRC...) is provided serially through the RX DATA GPIO	SABORT command needed
011	I/Q sampling	Raw I/Q sampling taken at the output of the Channel filter inside the demodulator are stored in RAM. PREAMBLE and SYNC word detections are active in parallel.	SABORT command needed
100	frequency/amplitude detector sampling	Raw data taken at the output of the frequency/amplitude detector inside the demodulator (detection of the instantaneous frequency/amplitude changes) are stored in RAM.	SABORT command needed
101	soft bits sampling	Raw data taken at the output of the clock recovery and frequency correction blocks are stored in RAM.	SABORT command needed

The following figure represents a simplified RX data path showing where the different reception mode pick-up the samples

Figure 280. Simplified RX data path



A few pieces of information can be extracted/summarized from the figure above concerning the different modes:

- I/Q samples are zero-IF “raw” data
- Frequency/amplitude detector samples provide the instantaneous frequency/amplitude measured deviation of the received signal
- Soft bits go through the clock recovery and AFC block: those samples provide the optimal sampling point as determined by the clock recovery block and, in case of (G)-FSK modulation, the measured frequency deviation is (ideally) centered around zero thanks to the AFC correcting the Carrier Frequency Offset.
- Direct through Buffer or Direct through GPIO modes samples do not cross the Viterbi/FEC decoding block: if received samples are FEC coded, then SW post-processing has to manage the decoding.
- Normal mode samples are fully managed by the RX digital chain and can be used directly by the SW.

Normal mode

This mode is selected with RX_MODE[2:0] = “000”.

In Normal mode:

- the demodulator starts a PREAMBLE detection and once detected, a SYNC word detection
- the deframer manages the length information and computes the CRC (if CRC mode is not 0) with comparison between the received CRC and the calculated CRC
- the PAYLOAD is written into the RAM through the DBM block
- the RX command stops automatically when all the elements of the frame have been received or on a RX timeout if activated.

Note: This mode is the only one supposed to be used with 802.15.4 packet format.

Direct through buffer mode

This mode is selected with RX_MODE[2:0] = “001”.

In Direct through buffer mode:

- the demodulator does not try to detect any PREAMBLE nor SYNC word and provides all received bits
- the deframer does not treat any length nor CRC information
- all the received bits are written into the RAM through the DBM block once RX state is effective
- the RX command must be stopped using a SABORT command.

In this mode, the SW is responsible to post-process the bit stream stored in RAM to detect if the frame is valid, if this device is the targeted recipient, etc.

The direct modes intended to completely bypass the automatic packet handler, in order to give the user a maximum flexibility in the choice of frame formats. This allows building a homemade protocol that may not be achieved despite all the programmable options.

Note: No Viterbi/FEC decoding is applied even if PCKT_CTRL[22:21] = CODING_SEL[1:0] = 01.

Direct through GPIO

This mode is selected with RX_MODE[2:0] = “010”.

In Direct through GPIO mode:

- the demodulator does not try to detect any PREAMBLE nor SYNC word and provides all received bits
- the deframer does not treat any length nor CRC information
- all the bits received from the air are sent to an I/O of the device once RX state is effective. In parallel an output RX clock (togging at data rate) is provided on a GPIO to allow the synchronization of an external sink
- data are updated by the device on the rising edge of the provided RX clock signal
- the RX command must be stopped using a SABORT command.

In this mode, the PREAMBLE, SYNC word, CRC, LENGTH (if present in the frame) are also provided through the RX GPIO.

This mode allows a bypass of the automatic packet handler and offers a maximum flexibility to receive a homemade protocol generated that is post-process by an external processor / device.

Note: No Viterbi/FEC decoding is applied even if PCKT_CTRL[22:21] = CODING_SEL[1:0] = 01.

I/Q sampling mode

This mode is selected with RX_MODE[2:0] = “011”.

This mode allows writing in RAM some raw I/Q sampling taken at the output of the Channel filter inside the demodulator.

In I/Q sampling mode:

- the demodulator provides all received bits once RX is effective
- in parallel, the demodulator starts a PREAMBLE detection and once detected, a SYNC word detection
- the deframer does not treat any length nor CRC information
- the I/Q samples are taken just after the channel filter of the demodulator as zero-IF signed data
- the I/Q samples are written in RAM as 16-bit I and 16-bit Q:
  – in cut 1.x, RAM word filled as byte0 = I[15:8], byte1 = I[7:0], byte2= Q[15:8], byte3 = Q[7:0]
  – in cut 2: RAM word filled as byte0 = I[7:0], byte1 = I[15:8], byte2= Q[7:0], byte3 = Q[15:8]
- the samples are produced at a rate that depends on the selected channel filter according to the following formula

$$F_{sample} = \frac{f_{sys}}{8 \times 2^{CHFLT\_E}}$$

- the SW is responsible to post-process those raw data
  – the value is linked to the received signal power through a scaling factor which depends on the current AGC gain setting and on an additional internal scale factor. Conversion to µV is not possible since this latter scaling factor is not accessible to the SW.
- the RX command must be stopped using a SABORT

As the data has a 32-bit granularity (= 16-bit of I + 16-bit of Q), the Data Buffer size must be a multiple of 4. In any case, the HW writes 32-bit so if the Data Buffer size is not modulo 4, the rest of the RAM address after the Data Buffer is overwritten by the Radio.

Note: The AGC activity may impact the amplitude result in this mode when enabled:

- the amplitude may do some 6 dB step down/up when the received RF signal cross an AGC threshold step,
- with default setting, the AGC locks (stop to adapt to the signal power) once the SYNC word is detected.

Frequency detector direct mode

This mode is selected with RX_MODE[2:0] = “100”.

This mode allows writing in RAM the instantaneous frequency deviation of the received signal when the receiver is configured in any FSK mode or the instantaneous amplitude of the received signal when the receiver is configured in OOK/ASK.

In Frequency detector direct mode:

- the demodulator provides all received bits once RX is effective
- in parallel, the demodulator starts a PREAMBLE detection and once detected, a SYNC word detection
- the deframer does not treat any length nor CRC information
- each frequency/amplitude sample is coded as 2’s complement byte
- the samples are produced at a rate that depends on the selected channel filter according to the following formula

$$F_{sample} = \frac{f_{sys}}{8 \times 2^{CHFLT\_E}}$$

- the SW is responsible to post-process those raw data
- the RX command must be stopped using a SABORT

For FSK modulations, the numerical value of the samples stored in RAM is linked to the actual frequency deviation as:

$$F_{dev\_RAM} = \frac{F_{dev\_Hz} \times 256}{F_{sample}}$$

Which means:

$$F_{dev\_Hz} = \frac{F_{dev\_Hz} \times F_{sample}}{256}$$

For ASK modulation, the situation is similar to the IQ samples: there is an internal scaling function that adapts the dynamic range at the receiver input to the dynamic range on 8 bits so cannot be converted directly to µV or dBm. The user can refer to the shape and to min/max values to extract the 0 or 1 bit estimation.

Note: *The AGC activity may impact the amplitude result in this mode when enabled:*

- the amplitude may do some 6 dB step down/up when the received RF signal cross an AGC threshold step,
- with default setting, the AGC locks (stop to adapt to the signal power) once the SYNC word is detected.

Soft symbols direct mode

This mode is selected with RX_MODE[2:0] = “101”.

This mode allows writing in RAM the “soft” bits at the output of the demodulator. The “soft” bits values represent the demodulated signal at the output of the clock recovery and frequency correction blocks before hard decision or FEC decoding.

In soft symbols direct mode:

- the demodulator provides all received bits once RX is effective
- in parallel, the demodulator starts a PREAMBLE detection and once detected, a SYNC word detection
- the deframer does not treat any length nor CRC information
- each soft bit is coded as a 2’s complement byte
- the samples are produced at symbol rate
- the SW is responsible to post-process those raw data
- the RX command must be stopped using a SABORT

For FSK modulations, the numerical value of the samples stored in RAM is linked to the actual frequency deviation and is similar to the frequency detector mode, that is:

$$F_{dev\_RAM} = \frac{F_{dev\_Hz} \times 256}{F_{sample}}$$

Which means:

$$F_{dev\_Hz} = \frac{F_{dev\_Hz} \times F_{sample}}{256}$$

The main difference with frequency detector RX mode is that:

- the frequency deviation is provided at the optimisal sampling point thanks to clock recovery algorithm
- the frequency deviation values are centered around zero thanks to the Carrier Frequency Offset correction done by the AFC algorithm (when AFC is enabled)

For ASK modulation, the situation is similar to the IQ samples: there is an internal scaling function that adapts the dynamic range at the receiver input to the dynamic range on 8 bits so cannot be converted directly to µV or dBm. The user can refer to the shape and to min/max values to extract the 0 or 1 bit estimation.

Note: The AGC activity may impact the amplitude result in this mode when enabled:

the amplitude may do some 6 dB step down/up when the received RF signal cross an AGC threshold step,

with default setting, the AGC locks (stop to adapt to the signal power) once the SYNC word is detected.

29.3.7 TX power control and configuration information

The Sub-GHz radio can be configured to optimize the efficiency of the power amplifier for three different the output power levels: +10 dBm, 14 dBm and +20 dBm for long distance emission. Each is frozen with a dedicated BOM configuration on the board:

- the chosen / implemented topology must be specified in the PA_CONFIG[9:8] = PA_DRV_MODE[1:0] bit field
  - PA_DRV_MODE[1:0] = “01” → TX→ up to 10 dBm
  - PA_DRV_MODE[1:0] = “10” → TX HP→ up to 14/16 dBm
  - PA_DRV_MODE[1:0] = “11” → TX + TX HP→ up to 20 dBm with VBatt ≥ 2.4V

Note: For PA_DRV_MODE[1:0]=11 and frequency modulation, it is recommended to set the PA_REG[3] = PA_DEGEN_ON bit to improve the power consumption and efficiency. This configuration is not adapted for amplitude modulation.

The figures below show the three topologies and the associated connections to make on the board.

Figure 281. TX and TX HP pin connections, TX PA drive mode, VBATT > 1.7 V

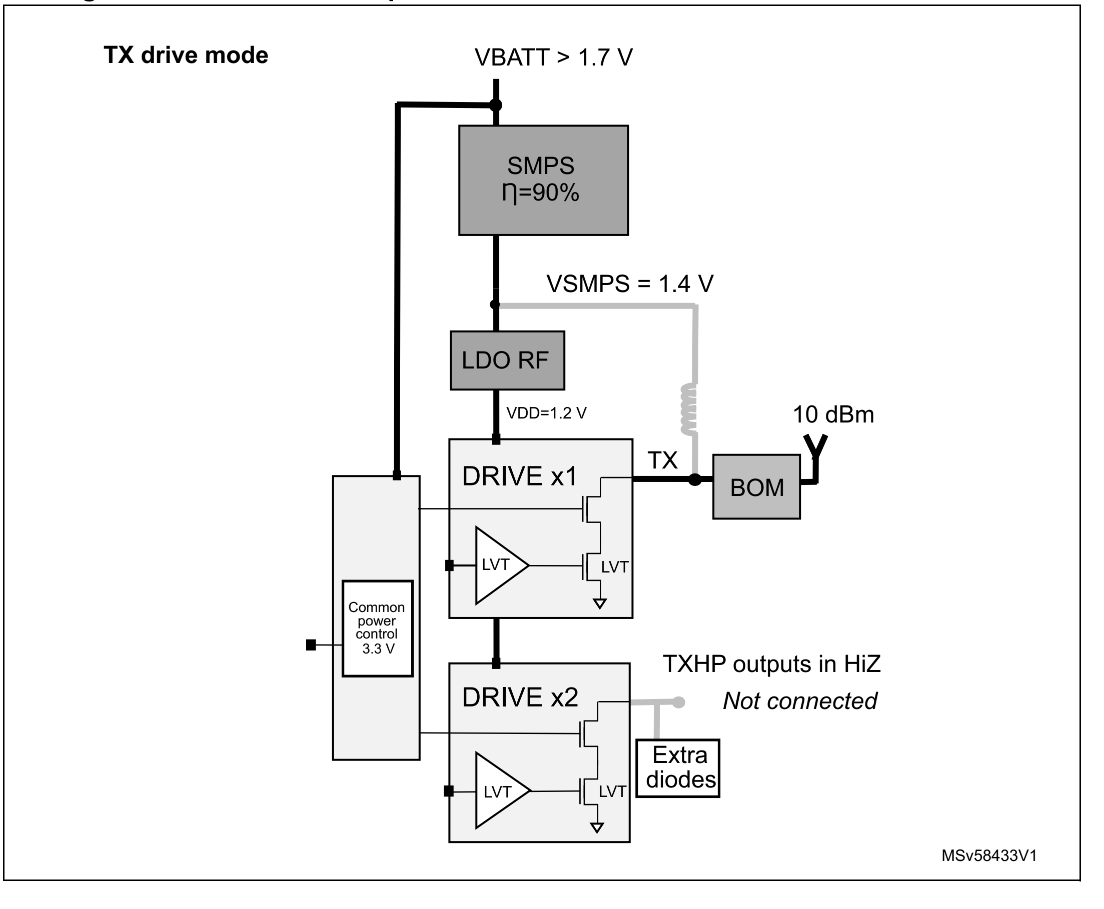


Figure 282. TX and TX HP pin connections, TX + TX HP PA drive mode, VBATT > 2.4 V

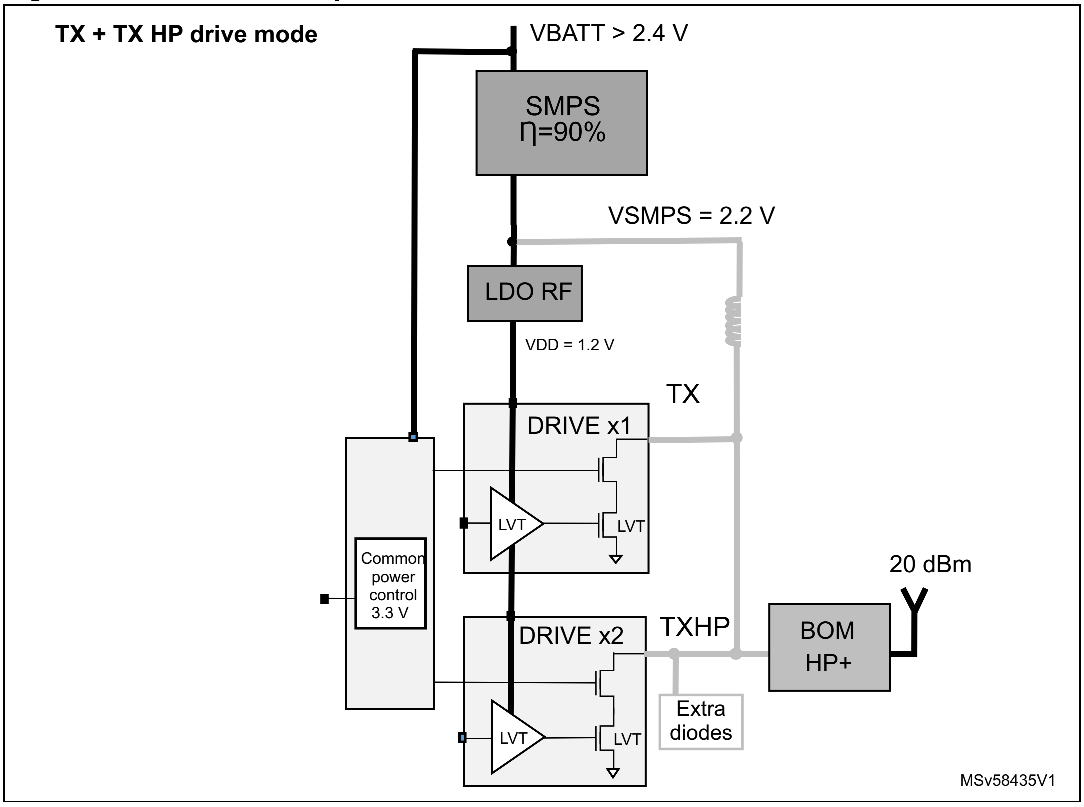
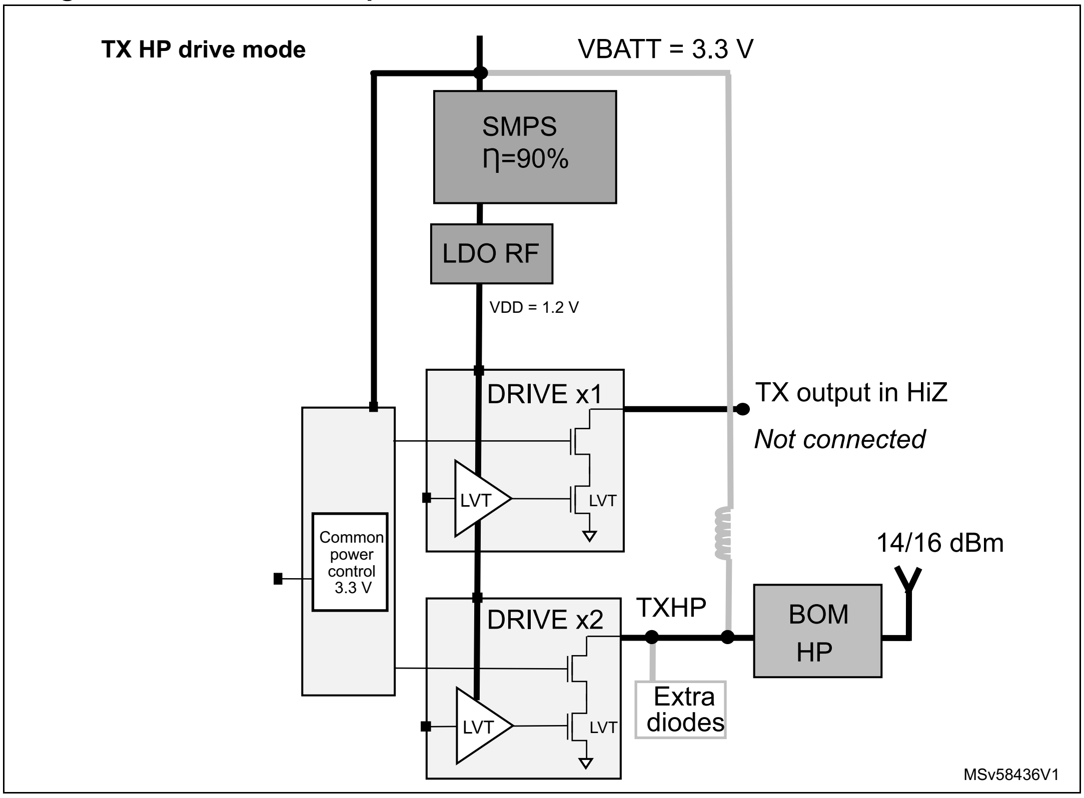


Figure 283. TX and TX HP pin connections, TX HP PA drive mode, VBATT = 3.3 V



29.3.8 Guidelines on registers configuration for standard RF

The MR_SubG IP offers a lot of flexibility to build any custom RF transfer. However, some HW implementation has been added to make some RF standard easy to build. This sub-chapter provides some guidelines on registers configuration to help building those RF standards with this radio IP.

802.15.4

This is the list of registers programming concerned/impacted by the 802.15.4 standard usage:

Packet format:

- Select 802.15.4 format in PCKT_CTRL[0] = PCKT_FORMAT bit

SYNC word:

- in Transmission:
  - if the FEC is enabled (see Coding/Interleave below): the HW sends the specified secondary SEC_SYNC word,
  - else, the HW sends the specified primary SYNC word.
- in Reception:
  - the receiver detects / searches a match on both SYNC and SEC_SYNC words

Note: In both TX and RX, PCKT_CONFIG[3] = SECONDARY_SYNC_SEL bit is not considered by the HW when packet format is 802.15.4.

Length of the packet (excludes the PHR length and includes the CRC length):

- in Transmission
  - the packet length is programmed in PCKTLEN_CONFIG[11:0] = PCKTLEN[11:0]
- in Reception:
  - the packet length is decoded in the PHR of the received frame, the HW does not consider the PCKTLEN_CONFIG[11:0] = PCKTLEN[11:0] bit field

Note: In both TX and RX, PCKT_CONFIG[10] = LEN_WIDTH and PCKT_CONFIG[11] = FIX_VAR_LEN bits are not considered by the HW when packet format is 802.15.4

The CRC is calculated on MHR + MAC Payload bit fields and appended to the end of the payload in transmission or calculated and compared to the CRC part of the frame in reception.

The programming for CRC generation (in TX) / decode on the fly (in RX) follows the rules:

- in Transmission
  - if CRC_MODE[1:0] = 0: no CRC inserted in the frame
  - else: CRC inserted in the frame depends on PCKT_CTRL[27] = FCS_TYP_4G bit

    FCS_TYP_4G = ‘0’, then 32-bit CRC mode5

    FCS_TYP_4G = ‘1’, then 13-bit CRC mode3
  - PCKT_CTRL[27] = FCS_TYP_4G

    defines the FCS bit value that is inserted in the PHR of the frame, regardless the CRC_MODE[1:0] bit field value
  - CRC_INIT static register must be programmed according to chosen CRC to fit the standard:

    if 2-byte CRC: CRC_INIT = 0x00000000

    if 4-byte CRC: CRC_INIT = 0xFFFFFFFF
- in Reception
  - if CRC_MODE[1:0] = 0: no CRC checked in the received frame
  - else: CRC check depends on the FCS bit value in the frame
  - FCS = ‘0’, then 32-bit CRC mode5 expected

    FCS = ‘1’, then 13-bit CRC mode3 expected
  - CRC_INIT static register must be programmed according to chosen CRC to fit the standard:

    if 2-byte CRC: CRC_INIT = 0x00000000
  - if 4-byte CRC: CRC_INIT = 0xFFFFFFFF
- Coding/Interleave:
  - In TX:

    Select FEC coding through PCKT_CTRL[22:21] = CODING_SEL[1:0]

    Select RSC or NRSC encoder through PCKT_CTRL[26] = FEC_TYPE_4G

    Enable or not interleave through PCKT_CTRL[25] = INT_EN_4G (this choice is considered by the HW only for frame with FEC active)
  - In RX:

    the receiver decides by HW if the FEC is active or not according to decoded synchro word (no FEC active if detected synchro word matches with the one defined in SYNC register / FEC active if the detected synchro word matches the one defined in the SEC_SYNC register)

    Select RSC or NRSC encoder through PCKT_CTRL[26] = FEC_TYPE_4G

    Enable or not interleave through PCKT_CTRL[25] = INT_EN_4G (this choice is considered by the HW only for frame with FEC active)

Note: PCKT_CTRL[22:21] = CODING_SEL[1:0] bit field is not considered by the receiver

Whitening:

- In TX: enable or not the whitening through PCKT_CTRL[11] = WHIT_EN
- In RX: the receiver decodes the whitening information on the received PHR (DW bit) ; it does not consider the PCKT_CTRL[11] information.
- **Warning**: the seed of the whitening PN9 algorithm is programmable in the PCKT_CTRL[20:12] = WHIT_INIT bit field.

  in 802.15.4 standard, this initialization seed must be programmed to 0x1FF.

  In any case (whatever the chosen seed), the SW is in charge to program the expected initialization value in the device in TX or in RX.
- Endianness:

  The SYNC/SEC_WORD is output MSB first (bit 31 first of the register)

  The PHR described in section *802.15.4 packet* is output on the antenna as b0 first (MS bit) and b15 (L0 bit) last

  MHR+MAC Payload: the bytes are read from the RAM and sent MSbit first (bit7 first on the antenna and bit0 last)

  CRC: the CRC is output MSbit first.

Sigfox transmission

This sub-chapter indicates the registers programming concerned/impacted to manage a Sigfox TX transfer.

Modulation:

- Select the TX Polar modulation in MOD0_CONFIG[22:20] = MOD_TYPE[2:0] bit field

PA configuration:

- Select the PA configuration to be Spirit1 legacy or linear mode through PA_CONFIG[11:10] = PA_MODE[1:0] bit field

Packet format:

- Select Basic format in PCKT_CTRL[0] = PCKT_FORMAT bit

CRC:

- Select mode0 (no CRC) in PCKT_CONFIG[2:0] =CRC_MODE[2:0] bit field

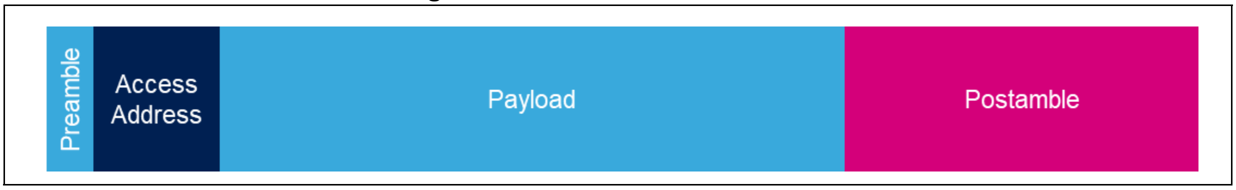
Data management:

- Build the Sigfox bits through RAM Data buffers, using the direct control of RF PLL and PA as described in section *Direct Polar TX mode*.

W-MBUS

This sub-chapter indicates the registers programming concerned/impacted to manage a W-MBUS RF transfer.

Figure 284. W-MBUS frame format



Packet format:

- Select Basic format in PCKT_CTRL[0] = PCKT_FORMAT bit

Access Address:

- Define the access address value in the SYNC register
- and its characteristics through PCKT_CONFIG[9] = SYNC_PRESENT and PCKT_CONFIG[8:4] = SYNC_LEN

CRC:

- The recommendation is to compute the CRC at SW level and not to use the HW CRC.
- Select mode0 in PCKT_CONFIG[2:0] = CRC_MODE[2:0]

Note: Main reason is that, if 3oo6 coding is selected, the CRC is calculated on encoded data and not on clear data

POSTAMBLE:

- Define a POSTAMBLE through
  - PCKT_CONFIG[31:30] = POSTAMBLE_SEQ bit field
  - and PCKT_CONFIG[29:24] = POSTAMBLE_LENGTH bit field

Coding:

- If needed, the 3oo6 coding can be enabled in PCKT_CTRL[22:21] = CODING_SEL[1:0]

The rest of the W-MBUS standard constraints must be built in SW.

29.4 Global registers

The Global registers block manages all the registers needed for the MR_SubG control except a few sets of registers targeting specific features like analog RF IP control and debug and test configuration (refer to *Section 29.10: Radio register descriptions* for details).

All the registers are accessible through an APB interface.

The Global registers block is also in charge of generating the MR_SubG IP interrupt as it contains the interrupt status register.

29.4.1 Register organization

The Global registers are grouped in several categories:

- Static Configuration registers: read / write registers used to configure the radio transfer (modulation, preamble format…)
- Dynamic Configuration registers: read / write registers used to configure the radio transfer (packet length, command to execute…)
- Status registers: read-only registers providing status information linked to the radio blocks.

Note: Interrupt status registers are "read / write 1 to clear" instead of read-only.

- Misc registers: registers to manage features not directly link to the RF transfer (for example, slow clock period measurement).
- Retained registers: registers keeping their content under low power mode sequences (so not to be restored at wakeup).

Table 119. Global registers sub-groups base address

Global register sub-group	Offset versus the Global register base address
Static registers	0x000
Dynamic registers	0x100
Status registers	0x200
Misc registers	0x300
Retained registers	0x380

Note: The global registers base address has an offset of 0x400 versus the Radio registers that represents the base address of the MR_SubG IP inside the SoC.

The Static and Dynamic configuration registers have the specificity to have two masters that can write them:

- the CPU using the MR_SubG IP APB interface,
- the sequencer through a direct internal access when starting a sequence

Note: The sequencer has the priority to write in the static or dynamic registers:

The Sequencer fills the Static and the Dynamic registers by burst and keep the access until the concerned category is fully written. So, if the CPU tries to do a write access to a static register while the Sequence is in a burst write to the Static registers, the CPU accesses is *delayed until the Sequencer access is over through a low HREADY (means CPU is stuck and cannot process any other action until the current request is served).*

There is a separate lock for Static and Dynamic group so the CPU can do a write access in parallel of the Sequencer writing in the Static group.

Having the Sequencer writing in those registers implies the Automatic mode has been selected and the SW is not supposed to modify those two groups of registers in addition of the Sequencer.

29.5 STATIC_REG_BLOCK register descriptions

The STATIC_REG_BLOCKBaseAddress keyword used for all registers base address information corresponds to Global registers base address decided by the SoC when integrating the IP.

*Note: **STATIC_REG_BLOCKBaseAddress** is 0x4900_0400 in the STM32WL30xx/31xx/33xx.*

The 0x400 offset is due to the fact the Radio Registers mapping corresponds to the first kByte of the APB mapping and Global registers to the second Kbyte.

Table 120. STATIC_REG_BLOCK register list

Offset	Register Name	Register Title	Reset
0x0000	PCKT_CONFIG	PCKT_CONFIG register	0x0001 03F1
0x0004	SYNC	SYNC register	0x2323 2323
0x0008	SEC_SYNC	SEC_SYNC register	0x0000 0000
0x000C	CRC_INIT	CRC_INIT register	0x0000 0000
0x0010	PCKT_CTRL	PCKT_CTRL register	0x0000 0000
0x0014	DATABUFFER0_PTR	DATABUFFER0_PTR register	0x0000 0000
0x0018	DATABUFFER1_PTR	DATABUFFER1_PTR register	0x0000 0000
0x001C	DATABUFFER_SIZE	DATABUFFER_SIZE register	0x0000 0000
0x0020	PA_LEVEL_3_0	PA_LEVEL_3_0 register	0x230B 0100
0x0024	PA_LEVEL_7_4	PA_LEVEL_7_4 register	0x5147 3B2F
0x0028	PA_CONFIG	PA_CONFIG register	0x0000 015C
0x002C	IF_CTRL	IF_CTRL register	0x04CD 04CD
0x0030	AS_QI_CTRL	AS_QI_CTRL register	0x5800 8028
0x0034	IQC_CONFIG	IQC_CONFIG register	0xC000 0000
0x0038	DSSS_CTRL	DSSS_CTRL register	0x0000 0000

29.5.1 PCKT_CONFIG register (PCKT_CONFIG)

Address offset: 0x0000

Reset value: 0x0001 03F1

PCKT_CONFIG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POSTAMBLE_SEQ[1:0]		POSTAMBLE_LENGTH[5:0]						PREAMBLE_SEQ[1:0]		PREAMBLE_LENGTH[9:4]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREAMBLE_LENGTH[3:0]				FIX_VAR_LEN	LEN_WIDTH	SYNC_PRESENT	SYNC_LEN[4:0]					SECONDARY_SYNC_SEL	CRC_MODE[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **POSTAMBLE_SEQ[1:0]**: Packet postamble control: postamble bit sequence selection

Possible patterns:

00: 010101…

01: 101010…

10: depends on last payload/CRC bit

=> If 0: 101010…

=> If 1: 010101…

Bits 29:24 **POSTAMBLE_LENGTH[5:0]**: Length of the POSTAMBLE in pair of bits (0 to 126 bits)

Bits 23:22 **PREAMBLE_SEQ[1:0]**: Select the PREAMBLE pattern to be applied

Possible patterns in 2-(G) FSK and ASK/OOK (leftmost bit is transmitted first):

00: 0101…

01: 1010…

10: 0011

11: 1100

Possible patterns in 4-(G) FSK (leftmost bit is transmitted first):

00: 0111…

01: 0010…

10: 1101

11: 1000

Bits 21:12 **PREAMBLE_LENGTH[9:0]**: Length of the PREAMBLE in pairs of bits (0 to 2046)

Bit 11 **FIX_VAR_LEN**: Select the length mode

0: FIXED length mode (no LENGTH field added in the frame in TX and no decode in RX)

1: VARIABLE length mode (LENGTH field put in the frame in TX and decoded in RX)

Note: this bit is considered/relevant only for Basic packets (PCKT_FORMAT=0)

Bit 10 **LEN_WIDTH**: Indicates if the LENGTH field is defined on 1 byte or 2 bytes
0: LENGTH bit field is defined on 1 byte
1: LENGTH bit field is defined on 2 bytes
Note: this bit field is considered/relevant only if FIX_VAR_LEN=1 (VARIABLE length mode).
Note: this bit field is considered/relevant only if FIX_VAR_LEN=1 (VARIABLE length mode) and Basic packets (PCKT_FORMAT=0).

Bit 9 **SYNC_PRESENT**: Indicate if a SYNC word is present on the frame or not (null length)
0: no SYNC word on the frame (equivalent to SYNC word length=0)
1: a SYNC word is present on the frame (length in SYNC_LEN bit field)

Bits 8:4 **SYNC_LEN[4:0]**: Length of the SYNC (and secondary) SYNC word in 1-bit granularity
00000: 1-bit wide
00001: 2-bit wide
-
11111: 32-bit wide (default)

Bit 3 **SECONDARY_SYNC_SEL**: In TX mode, this bit selects which synchro word is sent on the frame between SYNC and SEC_SYNC.
In RX mode: it enables the detection of SEC_SYNC in parallel of SYNC word.
Note: the SYNC is always detected.

Bits 2:0 **CRC_MODE[2:0]**: CRC type (0, 8, 16, 16 802.15.4 compatible, 24, 32 bits)
000: no CRC
001: 8-bit CRC (x^8+x^2+x+1)
010: 16-bit CRC (x^16+x^15+x^2+1)
011: 16-bit CRC (x^16+x^12+x^5+1) 15.4g compatible
100: 24-bit CRC (x^24+x^23+x^18+x^17+x^14+x^11+x^10+x^7+x^6+x^5+x^4+x^3+x+1)
101: 32-bit CRC (x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x+1)
11x: Reserved for future used
Note: the init value of the CRC polynomial is programmable in CRC_INIT register

29.5.2 SYNC register (SYNC)

Address offset: 0x0004

Reset value: 0x2323 2323

SYNC register

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

Bits 31:0 **SYNC[31:0]**: Synchro word.

Note: in transmission, the SYNC word is read from MSB. So if SYNC_LEN is less than 32-bit, used value is MSB part, for example, SYNC_LEN = 17 means SYNC[31:15] is sent from bit31 to bit15 on the air.

29.5.3 SEC_SYNC register (SEC_SYNC)

Address offset: 0x0008

Reset value: 0x0000 0000

SEC_SYNC register

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

Bits 31:0 **SEC_SYNC[31:0]**: Secondary Synchro word.

Note: in transmission, the SEC_SYNC word is read from MSB. So if SYNC_LEN is less than 32-bit, used value is MSB part, for example, SYNC_LEN = 17 means SEC_SYNC[31:15] is sent from bit31 to bit15 on the air.

29.5.4 CRC_INIT register (CRC_INIT)

Address offset: 0x000C

Reset value: 0x0000 0000

CRC_INIT register

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

Bits 31:0 **CRC_INIT_VAL[31:0]**: CRC intialization value

29.5.5 PCKT_CTRL register (PCKT_CTRL)

Address offset: 0x0010

Reset value: 0x0000 0000

PCKT_CTRL register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FORCE_2FSK_SYNC_MODE	Res.	PN_SEL	MOD_INTERP_EN	FCS_TYPE_4G	FEC_TYPE_4G	INT_EN_4G	MANCHESTER_TYPE	Res.	CODING_SEL[1:0]		WHIT_INIT[8:4]				
rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WHIT_INIT[3:0]				WHIT_EN	WHIT_BF_FEC	Res.	TX_MODE[1:0]		RX_MODE[2:0]			FOUR_FSK_SYM_SWAP	BYTE_SWAP	Res.	PCKT_FORMAT
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **FORCE_2FSK_SYNC_MODE**: Force SYNC word to be formatted as a 2-(G) FSK bit steam instead of 4-(G)FSK

0: the SYNC word is treated in 4-(G) FSK mapping as the rest of the frame

1: the SYNC word of the frame is mapped as follows:

- '1' is mapped to +Fdev; '0' is mapped to -Fdev for constellation mapping = 0 and 1
- '1' is mapped to -Fdev; '0' is mapped to +Fdev for constellation mapping = 2 and 3

Bit 30 Reserved, must be kept at reset value.

Bit 29 **PN_SEL**: Select the Pseudo Random Binary Sequence (PRBS) polynomial to apply when the selected transmission mode is PN mode (TX_MODE = "11")

0: PRBS9 (x^9 + x^5 +1) (default)

1: PRBS15 (x^15+ x^14 + 1)

Bit 28 **MOD_INTERP_EN**: Enable frequency interpolator (for 2-GFSK and 4-GFSK)

This feature linearly interpolates the output of the Gaussian filter.

Bit 27 **FCS_TYPE_4G**: FCS type value in header field for 802.15.4g

0: 32-bit CRC (equivalent to mode5 in PCKT_CONFIG[2:0])

1: 16-bit CRC (equivalent to mode3 in PCKT_CONFIG[2:0])

Bit 26 **FEC_TYPE_4G**: FEC type for 802.15.4g

0: NRNSC

1: RSC

Bit 25 **INT_EN_4G**: This field is used as Interleaving enable for 802.15.4g.

Note: considered only for 802.15.4 packets.

Bit 24 **MANCHESTER_TYPE**: Select the Manchester encoding polarity
0: ‘0’is encoded with ’01’and ‘1’is encoded with ‘10’
1: ‘0’is encoded with ’10’and ‘1’is encoded with ‘01’

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **CODING_SEL[1:0]**: Coding / decoding selection
00: none
01: FEC in TX / Viterbi in RX
10: 3oo6 coding
11: Manchester coding (not compliant with 802.15.4 packet format nor 4-(G) FSK modulation type)
Note: Viterbi (FEC decoding) in RX selection is relevant only for RX_MODE = normal mode

Bits 20:12 **WHIT_INIT[8:0]**: Whitening initialization value.
Note: use 0x1FF to be compliant with the S2-LP behavior.

Bit 11 **WHIT_EN**: Whitening enable
Note: if the packet format is 802.15.4, this bit is considered only in transmission.

Bit 10 **WHIT_BF_FEC**: Whitening before FEC feature
0: S2-LP order: FEC then whitening
1: whitening then FEC
Note: this bit must be kept to 0 for 802.15.4 packets

Bit 9 Reserved, must be kept at reset value.

Bits 8:7 **TX_MODE[1:0]**: TX mode
00: normal mode,
01: direct mode (through Data Buffer),
10: direct mode through GPIO,
11: PN mode (test purpose)
Note: direct modes are supposed to be used only on Basic packet format (not on 802.15.4 packets)

Bits 6:4 **RX_MODE[2:0]**: RX mode
000: normal mode,
001: direct mode (through Data Buffer),
010: direct mode through GPIO
011: direct I/Q sampling mode
100: frequency detector direct mode
101: soft symbols direct mode
11x: reserved for future used
Note: direct modes are supposed to be used only on Basic packet format (not on 802.15.4 packets)

Bit 3 **FOUR_FSK_SYM_SWAP**: Invert bit to symbol mapping for 4-(G) FSK
Note: the swap applies only on the payload (excluding PREAMBLE, SYNC, LENGTH and CRC part of the frame)
Note: the swap is applied only on the payload (exclude PREAMBLE, SYNC, LENGTH and CRC part of the frame)

Bit 2 **BYTE_SWAP**: Invert MSB-LSB transmission order (bitendianess)
0: bytes are transmitted MSBit first
1: bytes are transmitted LSBit first
Note: this feature can be used only if the packet format is Basic Packet (not for 802.15.4 packet)
Note: the swap is applied only on the payload (exclude PREAMBLE, SYNC, LENGTH and CRC part of the frame)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **PCKT_FORMAT**: Packet format
0: Basic Packet
1: 802.15.4 packet

29.5.6 DATABUFFER0_PTR register (DATABUFFER0_PTR)

Address offset: 0x0014

Reset value: 0x0000 0000

DATABUFFER0_PTR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATABUFFER0_PTR[29:14]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATABUFFER0_PTR[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **DATABUFFER0_PTR[29:0]**: Start address to be used by the Data Buffer0
Note: the register value is considered in real-time by the DBM. The SW has the responsibility not to modify the pointer address while the DBM is using it.
Note: As the address programmed in this register must be 32-bit aligned, the 2 LSBit of the register are not writable and is always be considered as 0.

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

29.5.7 DATABUFFER1_PTR register (DATABUFFER1_PTR)

Address offset: 0x0018

Reset value: 0x0000 0000

DATABUFFER1_PTR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATABUFFER1_PTR[29:14]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATABUFFER1_PTR[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **DATABUFFER1_PTR[29:0]**: Start address to be used by the Data Buffer1

Note: the register value is considered in real-time by the DBM. The SW has the responsibility not to modify the pointer address while the DBM is using it.

Note: As the address programmed in this register must be 32-bit aligned, the 2 LSBit of the register are not writable and is always be considered as 0.

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

29.5.8 DATABUFFER_SIZE register (DATABUFFER_SIZE)

Address offset: 0x001C

Reset value: 0x0000 0000

DATABUFFER_SIZE register

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

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

Bits 15:0 **DATABUFFER_SIZE[15:0]**: Size of the Data Buffers (Data Buffer0 and Data Buffer1) expressed in byte unit.

Note: this paramater is automatically checked by the DBM whatever the selected TX/RX mode. So when 0, a DBM_FIFO_ERROR_F flag is raised even for modes not using the RAM. In this cases, this flag is meaningless.

Warning: the register value is considered in real-time by the DBM. The SW has the responsibility not to modify the Data Buffer size while the DBM is using it. Especially in RX, decreasing the size on-the-fly may have catastrophic effects in RAM: if the DBM already passed the new size limit, the DBM chains writings incrementing the address without switching the buffers anymore. For a TX in direct modes, in same conditions, garbage is sent on the antenna.

29.5.9 PA_LEVEL_3_0 register (PA_LEVEL_3_0)

Address offset: 0x0020

Reset value: 0x230B 0100

PA_LEVEL_3_0 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA_LEVEL3[7:0]								PA_LEVEL2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA_LEVEL1[7:0]								PA_LEVEL0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **PA_LEVEL3[7:0]**: Output power level for fourth step

(Default value corresponds to Max - 23 dB)

Note: this bit field is not used when PA_MODE="01"

Bits 23:16 **PA_LEVEL2[7:0]**: Output power level for third step

(Default value corresponds to Max - 35 dB)

Note: this bit field is not used when PA_MODE="01"

Bits 15:8 **PA_LEVEL1[7:0]**: Output power level for second step

(Default value corresponds to Max - 40 dB)

Note: this bit field is not used when PA_MODE="01"

Bits 7:0 **PA_LEVEL0[7:0]**: Output power level for first step
(Default value corresponds to no output power)
For all PA_LEVELx bit fields:
- when PA_MODE = 0x: PA_LEVEL[6:0] only used and value is in dBm
- when PA_MODE = 10: PA_LEVEL[7:0] fully used and value is linear

29.5.10 PA_LEVEL_7_4 register (PA_LEVEL_7_4)

Address offset: 0x0024

Reset value: 0x5147 3B2F

PA_LEVEL_7_4 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA_LEVEL7[7:0]								PA_LEVEL6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA_LEVEL5[7:0]								PA_LEVEL4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **PA_LEVEL7[7:0]**: Output power level for eighth step (Max up to 16 or 20 dBm depending on the BOM)

Note: in units of dBm, 0x51 corresponds to the maximum possible PA power. Any decrement (-1) then represents -0.5 dB

Bits 23:16 **PA_LEVEL6[7:0]**: Output power level for seventh step.

(Default value corresponds to Max - 5 dB)

Note: this bit field is not used when PA_MODE="01"

Bits 15:8 **PA_LEVEL5[7:0]**: Output power level for sixth step

(Default value corresponds to Max - 11 dB)

Note: this bit field is not used when PA_MODE="01"

Bits 7:0 **PA_LEVEL4[7:0]**: Output power level for fifth step

(Default value corresponds to Max - 17 dB)

- When PA_MODE = 0x: PA_LEVELx[6:0] only used and value is in dB
- When PA_MODE = 10: PA_LEVELx[7:0] fully used and value is linear

Warning: dBm coding is inverted versus S2-LP: bigger is the PA_LEVEL_x value, bigger is the power)

Note: this bit field is not used when PA_MODE="01"

29.5.11 PA_CONFIG register (PA_CONFIG)

Address offset: 0x0028

Reset value: 0x0000 015C

PA_CONFIG register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PA_RAMP_ENABLE	LIN_NLOG	Res.	PA_MODE[1:0]		PA_DRV_MODE[1:0]		ASK_OOK_EN	PA_INTERP_EN	Res.	PA_LEVEL_MAX_INDEX[2:0]			PA_RAMP_STEP_WIDTH[1:0]	
	rw	rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **PA_RAMP_ENABLE**: Enable the power ramping
- when PA_MODE[1:0] = "x0": enable the (up and down) power ramping
- when PA_MODE[1:0] = "01": add a delay of 1 symbol before to stop the transmission to consider the FIR latency (to let time to last bit to be issued on the antenna)

Bit 13 **LIN_NLOG**: Enable/disable the linear-to- log conversion of the PA code output from Safe-ASK calibrator
0: PA_CODEMAX[7:0] content is expressed in logarithmic scale
1: PA_CODEMAX[7:0] content is expressed in linear scale

Bit 12 Reserved, must be kept at reset value.

Bits 11:10 **PA_MODE[1:0]**: Configure the Power Amplifier (PA) mode
00: SPIRIT1 'legacy' mode (default)
01: FIR active (to be used in ASK/OOK modulations only)
10: direct linear mode
11: reserved for future used

Bits 9:8 **PA_DRV_MODE[1:0]**: Select the PA topology
Depends on BOM / board connections choice
00: Reserved
01: TX pin
10: TX HP1 + TX HP2
11: TX + TX HP1 + TX HP2

Bit 7 **ASK_OOK_EN**: Enable the generation of the internal TXDATA signal provided to the FIR.
Needed when PA_MODE[1:0] = 01 (otherwise FIR is excluded, output fixed at level '1')

Bit 6 **PA_INTERP_EN**: Enable power level interpolator.
When enabled, the PA_POWER registers are linearly interpolated by the modulator before being applied to the PA.
Note: to be kept at '0' when PA_MODE[1:0] = "01"

Bit 5 Reserved, must be kept at reset value.

Bits 4:2 **PA_LEVEL_MAX_INDEX[2:0]**: Final level for power ramping (that is, number of steps of ramping)

Note: this bit field is not considered when PA_MODE[1:0] = "01"

Bits 1:0 **PA_RAMP_STEP_WIDTH[1:0]**: Step width (unit: 1/8 of bit period).

00: 1 step duration is 1/8 of bit period
01: 1 step duration is 2/8 of bit period
10: 1 step duration is 3/8 of bot period
11: 1 step duration is 4/8 of bit period

The full PA ramping (up or down) time is (in symbol unit):

TRAMP = (PA_LEVEL_MAX_INDEX + 1) x PA_RAMP_STEP_WIDTH + 1) / 8

Note: this bit field is not considered when PA_MODE[1:0] = "01"

29.5.12 IF_CTRL register (IF_CTRL)

Address offset: 0x002C

Reset value: 0x04CD 04CD

IF_CTRL register

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

Bit 31 **IF_MODE**: Select the cutoff frequency of the AAF for the analog RFSUBG IP

0 (default): FC=684 kHz (for IF=300 kHz)
1: (extended) FC=1.55 MHz (for IF=600 kHz)

Bits 30:29 Reserved, must be kept at reset value.

Bits 28:16 **IF_OFFSET_ANA[12:0]**: Intermediate frequency setting for the synthesizer configuration (default: 300 kHz).

Granularity of programming is steps of 244Hz with XO = 48 MHz.

Note: it is recommended to keep IF_OFFSET_ANA = IF_OFFSET_DIG

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **IF_OFFSET_DIG[12:0]**: Intermediate frequency setting for the digital shift-to-baseband circuits (default: 300 kHz)

Granularity of programming is steps of 244 Hz with XO = 48 MHz.

Note: it is recommended to use only 300 kHz and 600 kHz values. The choice between both is driven by the programming of the channel filter bandwidth.

29.5.13 AS_QI_CTRL register (AS_QI_CTRL)

Address offset: 0x0030

Reset value: 0x5800 8028

AS_QI_CTRL register

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

Bit 31 **AS_CS_BLANKING**: Blank received data if signal is below the CS threshold

0: the radio may detect PREAMBLE and SYNC when receiving a frame with RSSI under programmed threshold (CS_F flag low)

1: the PREAMBLE and the SYNC word detections are gated if RSSI is under threshld (CS_F flag low)

Bits 30:28 **AS_MEAS_TIME[2:0]**: Select the RSSI measurement duration during Antenna switching procedure

Bits 27:26 **AS_EQU_CTRL[1:0]**: ISI cancellation equalizer

00: equalization disabled

01: single pass equalization

10: dual pass equalization

11: reserved for future use

Bits 25:19 Reserved, must be kept at reset value.

Bits 18:16 **SQI_THR[2:0]**: SQI threshold defining the precision requested to detect the SYNC word.

0: perfect match is required between detected SYNC word and expected one (SYNC[x:0] with x = SYNC_LEN-1)

1: 1 bit error is accepted

- …

- 7: 7 bits of error is accepted

Bit 15 **SQI_EN**: SQI enable

0: SQI check is disabled

1: "SYNC valid" flag is raised when the SQI threshold is passed

Bits 14:13 **CS_MODE[1:0]**: Carrier Sense mode selection
00: static carrier sensing
01: dynamic carrier sensing with 6 dB dynamic threshold
10: dynamic carrier sensing with 12 dB dynamic threshold
11: dynamic carrier sensing with 18 dB dynamic threshold

Bits 12:9 **PQI_THR[3:0]**: PQI threshold (if 0 then).
- When 0, PQI check is disabled.
In this case, the PREAMBLE valid flag is imemdiately raised at the beginning of a reception (even if no signal on the antenna)
- else, optionnal configuration to constrain the SYNC word detection to the detection of a valid PREAMBLE
In this case, PREAMBLE_VALID_F is raised as soon as PQI_INFO reaches 4 x PQI_THR
Note: when PQI_THR is non null, the threshold value must be much lower than effective PREAMBLE length sent by the transmitter

Bits 8:0 **RSSI_THR[8:0]**: Signal detect threshold in 1 dB resolution.
Default: 40d which is around -140 dBm

29.5.14 IQC_CONFIG register (IQC_CONFIG)

Address offset: 0x0034

Reset value: 0xC000 0000

IQC_CONFIG register

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

Bit 31 **IQC_ENABLE**: Enable IQC

Bit 30 **REUSE_CORRECTION**: Reuse last correction value

Bit 29 **LOAD_IQC_INIT**: Action bit.
Loads the IQC_CORRECT_IN[23:0] bit field in the recirculation register when this bit is written to 1.
This bit is cleared by HW (always read back to 0).

Bits 28:24 Reserved, must be kept at reset value.

Bits 23:0 **IQC_CORRECT_IN[23:0]**: Correction value Input for the IQ compensation engine (to be used as starting point or when the engine is disabled).
This value is loaded in the recirculation by means of the LOAD_IQC_INIT bit.

29.5.15 DSSS_CTRL register (DSSS_CTRL)

Address offset: 0x0038

Reset value: 0x0000 0000

DSSS_CTRL register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACQ_THR[5:0]						ACQ_HITS[1:0]		DSSS_EN	SPREADING_EXP[2:0]			ACQ_WINDOW[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:10 **ACQ_THR[5:0]**: DSSS acquisition threshold

Bits 9:8 **ACQ_HITS[1:0]**: DSSS acquisition hits

Bit 7 **DSSS_EN**: DSSS mode enable

Bits 6:4 **SPREADING_EXP[2:0]**: DSSS spreading exponent

Bits 3:0 **ACQ_WINDOW[3:0]**: DSSS acquisition window

29.5.16 STATIC_REG_BLOCK register map

Table 121. STATIC_REG_BLOCK register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PCKT_CONFIG	POSTAMBLE_SEQ[1:0]		POSTAMBLE_LENGTH[5:0]						PREAMBLE_SEQ[1:0]		PREAMBLE_LENGTH[9:0]										FIX_VAR_LEN	LEN_WIDTH	SYNC_PRESENT	SYNC_LEN[4:0]					SECONDARY_SYNC_SEL	CRC_MODE[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	1
0x004	SYNC	SYNC[31:0]																															
	Reset value	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1
0x008	SEC_SYNC	SEC_SYNC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	CRC_INIT	CRC_INIT_VAL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	PCKT_CTRL	FORCE_2FSK_SYNC_MODE	Res.	PN_SEL	MOD_INTERP_EN	FCS_TYPE_4G	FEC_TYPE_4G	INT_EN_4G	MANCHESTER_TYPE	Res.	CODING_SEL[1:0]		WHIT_INIT[8:0]									WHIT_EN	WHIT_BF_FEC	Res.	TX_MODE[1:0]		RX_MODE[2:0]			FOUR_FSK_SYM_SWAP	BYTE_SWAP	Res.	PCKT_FORMAT
	Reset value	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0
0x014	DATABUFFER0_PTR	DATABUFFER0_PTR[29:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x018	DATABUFFER1_PTR	DATABUFFER1_PTR[29:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x01C	DATABUFFER_SIZE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATABUFFER_SIZE[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	PA_LEVEL_3_0	PA_LEVEL3[7:0]								PA_LEVEL2[7:0]								PA_LEVEL1[7:0]								PA_LEVEL0[7:0]							
	Reset value	0	0	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x024	PA_LEVEL_7_4	PA_LEVEL7[7:0]								PA_LEVEL6[7:0]								PA_LEVEL5[7:0]								PA_LEVEL4[7:0]							
	Reset value	0	1	0	1	0	0	0	1	0	1	0	0	0	1	1	1	0	0	1	1	1	0	1	1	0	0	1	0	1	1	1	1

Table 121. STATIC_REG_BLOCK register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	PA_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PA_RAMP_ENABLE	LIN_NLOG	Res.	PA_MODE[1:0]		PA_DRV_MODE[1:0]		ASK_OOK_EN	PA_INTERP_EN	Res.	PA_LEVEL_MAX_INDEX[2:0]			PA_RAMP_STEP_WIDTH[1:0]	
	Reset value																		0	0		0	0	0	1	0	1		1	1	1	0	0
0x02C	IF_CTRL	IF_MODE	Res.	Res.	IF_OFFSET_ANA[12:0]													Res.	Res.	Res.	IF_OFFSET_DIG[12:0]												
	Reset value	0			0	0	1	0	0	1	1	0	0	1	1	0	1				0	0	1	0	0	1	1	0	0	1	1	0	1
0x030	AS_QI_CTRL	AS_CS_BLANKING	AS_MEAS_TIME[2:0]			AS_EQU_CTRL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQI_THR[2:0]			SQI_EN	CS_MODE[1:0]		PQI_THR[3:0]				RSSI_THR[8:0]								
	Reset value	0	1	0	1	1	0								0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
0x034	IQC_CONFIG	IQC_ENABLE	REUSE_CORRECTION	LOAD_IQC_INIT	Res.	Res.	Res.	Res.	Res.	IQC_CORRECT_IN[23:0]																							
	Reset value	1	1	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DSSS_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACQ_THR[5:0]						ACQ_HITS[1:0]		DSSS_EN	SPREADING_EXP[2:0]			ACQ_WINDOW[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

29.6 DYNAMIC_REG_BLOCK register descriptions

The **DYNAMIC_REG_BLOCKBaseAddress** keyword used for all registers base address information corresponds to Global registers base address+ 0x100 (with Global registers base address decided by the SoC when integrating the IP).

*Note: **DYNAMIC_REG_BLOCKBaseAddress** is 0x4900_0500 in the STM32WL30xx/31xx/33xx.*

Table 122. DYNAMIC_REG_BLOCK register list

Offset	Register Name	Register Title	Reset
0x0000	PCKTLEN_CONFIG	PCKTLEN_CONFIG register	0x0000 0014
0x0004	MOD0_CONFIG	MOD0_CONFIG register	0x0008 3A93
0x0008	MOD1_CONFIG	MOD1_CONFIG register	0x0040 0435
0x000C	SYNTH_FREQ	SYNTH_FREQ register	0x0485 1615
0x0010	VCO_CAL_CONFIG	VCO_CAL_CONFIG register	0x0040 0088
0x0014	RX_TIMER	RX_TIMER register	0x0000 0000
0x0018	DATABUFFER_THR	DATABUFFER_THR register	0x0000 0000
0x001C	RFSEQ_IRQ_ENABLE	RFSEQ_IRQ_ENABLE register	0x0000 0000
0x0020	ADDITIONAL_CTRL	ADDITIONAL_CTRL register	0x0003 8800
0x0024	FAST_RX_TIMER	FAST_RX_TIMER register	0x0000 0000
0x0028	COMMAND	COMMAND register	0x0000 0000

29.6.1 PCKTLEN_CONFIG register (PCKTLEN_CONFIG)

Address offset: 0x0000

Reset value: 0x0000 0014

PCKTLEN_CONFIG register

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

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

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

Bits 15:0 **PCKTLEN[15:0]**: This bit field has different meanings/usages:

When Normal Buffer mode and FIXED length configuration (FIX_VAR_LEN=0): length of packet in bytes for both RX and TX transfers.

When Normal Buffer mode and VARIABLE length configuration (FIX_VAR_LEN=1):

- in TX: length of packet in bytes
- in RX: MaxReceive size in bytes (so the RX_TOP block stops the reception if decoded length is above PCKTLEN value)

When Direct mode through Buffer:

- in TX: number of bytes before TX_TOP block stops the transmission (if selected modulation is not Polar TX)

Note: in this case, 0 means infinite TX until a SABORT command is raised.

- in RX: this bit field is not used

For all other use-cases, this bit field is not used.

Note: for 802.15.4 packets (PCKT_FORMAT=1), only the 11 LSBs are considered and only in transmission.

29.6.2 MOD0_CONFIG register (MOD0_CONFIG)

Address offset: 0x0004

Reset value: 0x0008 3A93

MOD0_CONFIG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA_CLKON_LOCKONTX	Res.	Res.	Res.	Res.	BT_SEL	CONST_MAP[1:0]		Res.	MOD_TYPE[2:0]			DATARATE_E[3:0]			
rw					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

Bit 31 **PA_CLKON_LOCKONTX**: Enable the clock on analog PA in LOCKONTX state
- 0: analog PA is clocked only in EN_PA/TX/PA_DOWN_ANA states of the Radio FSM
- 1: analog PA is also clock during LOCKONTX state

Note: this bit can be used to absorb a potential XO drift due to RF regulator reaction on consumption peak generated by analog PA clocking

Bits 30:27 Reserved, must be kept at reset value.

Bit 26 **BT_SEL**: Select BT value for GFSK
- 0: BT = 1
- 1: BT = 0.5

Bits 25:24 **CONST_MAP[1:0]**: Also known as FOUR_GFSK_CONST_MAP.

2- and 4-GFSK constellation mapping

Note: in ASK/OOK modulation, any different value from "00" reverse the 0/1 bit generation

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **MOD_TYPE[2:0]**: Select the modulation type
- 000: 2-FSK
- 001: 4-FSK
- 010: 2-GFSK
- 011: 4-GFSK
- 100: reserved
- 101: ASK/OOK
- 110: Polar TX
- 111: CW

Bits 19:16 **DATARATE_E[3:0]**: The exponent of the specified data rate (default: 38.4 ksps)

Bits 15:0 **DATARATE_M[15:0]**: The mantissa of the specified data rate (default: 38.4 ksps)

29.6.3 MOD1_CONFIG register (MOD1_CONFIG)

Address offset: 0x0008

Reset value: 0x0040 0435

MOD1_CONFIG register

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

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

Bits 23:20 **CHFLT_E[3:0]**: Exponent of the channel filter BW (default: 100 kHz)

Bits 19:16 **CHFLT_M[3:0]**: Mantissa of the channel filter BW (default: 100 kHz)

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

Bits 11:8 **FDEV_E[3:0]**: Exponent of the frequency deviation (default: 28.29 kHz)

Bits 7:0 **FDEV_M[7:0]**: Mantissa of the frequency deviation (default: 28.29 kHz)

29.6.4 SYNTH_FREQ register (SYNTH_FREQ)

Address offset: 0x000C

Reset value: 0x0485 1615

SYNTH_FREQ register

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **BS**: Synthesizer band selector, that is, out-of-loop total divide factor, 4 or 8
- 0: out-of-loop divider factor is 4 (for 826 MHz - 958 MHz band)
- 1 (default): out-of-loop divider factor is 8 (for 413MHz - 479MHz)

For devices with the 159 -185 MHz frequency band option **only -** Synthesizer band selector, that is, out-of-loop total divide factor, 4 or 8
- 0: out-of-loop divider factor is 20 (for 169 MHz band)
- 1 (default): out-of-loop divider factor is 8 (for 413 MHz – 479 MHz band)

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **SYNTH_INT[7:0]**: PLL integer divide factor (default: 868 MHz, XTAL: 48 MHz)

For devices with the 159 -185 MHz frequency band option **only -** PLL integer divide factor (default: 169 MHz, XTAL: 48 MHz)

Bits 19:0 **SYNTH_FRAC[19:0]**: Fractional part of the PLL fractional divide factor (default: 868 MHz, XTAL: 48 MHz)

For devices with the 159 -185 MHz frequency band option **only -** Fractional part of the PLL fractional divide factor (default: 169 MHz, XTAL: 48 MHz)

29.6.5 VCO_CAL_CONFIG register (VCO_CAL_CONFIG)

Address offset: 0x0010

Reset value: 0x0040 0088

VCO_CAL_CONFIG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VCO_CALIB_REQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALFREQ_EXT_SEL	VCO_CALFREQ_EXT[6:0]						
rw								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 31 **VCO_CALIB_REQ**: Define if the Radio FSM must launch a VCO calibration request after VCO start-up

0:the Radio FSM goes directly from SYNTH_SETUP state to LOCKRXTX state (no new calibration requested, the calibration block provides the last calibration results).

1: the Radio FSM goes through VCO_CALIB state after SYNTH_SETUP to request a new VCO calibration (amplitude and/or frequency).

Note: the VCO_CALFREQ_EXT and VCO_CALAMP_EXT bits can be combined to this bit to recalibrate only amplitude or frequency.

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **VCO_CALFREQ_EXT_SEL**: Select the mode to provide an external VCO frequency calibration value through VCO_CALFREQ_EXT bit field

0: the VCO frequency calibration value generated by the VCO calibration block is applied to the VCO

1: the VCO_CALFREQ_EXT[6:0] bit field of this register is applied to the VCO

Bits 22:16 **VCO_CALFREQ_EXT[6:0]**: VCO Cbank frequency calibration word.

This value is applied to the VCO if VCO_CALFREQ_EXT_SEL=1.

Bit 15 **VCO_CALAMP_EXT_SEL**: Select the mode to provide an external VCO amplitude calibration value through VCO_CALAMP_EXT bit field

0: the VCO amplitude calibration value generated by the VCO calibration block is applied to the VCO

1: the VCO_CALAMP_EXT[13:0] bit field of this register is applied to the VCO

Bit 14 Reserved, must be kept at reset value.

Bits 13:0 **VCO_CALAMP_EXT[13:0]**: VCO magnitude calibration word in thermometric code

This value is applied to the VCO if VCO_CALAMP_EXT_SEL=1.

29.6.6 RX_TIMER register (RX_TIMER)

Address offset: 0x0014

Reset value: 0x0000 0000

RX_TIMER register

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

Bit 31 **RX_OR_NAND_SELECT**: Select logical OR or logcial AND to apply on CS/PQI/SQI timeout mask

0: AND is applied

1: OR is applied

Bit 30 **RX_SQI_TIMEOUT_MASK**:

0: SYNC valid flag does not contribute to timeout disabling

1: SYNC valid flag contributes to timeout disabling

Bit 29 **RX_PQI_TIMEOUT_MASK**:

0: PREAMBLE valid flag does not contribute to timeout disabling

1: PREAMBLE valid flag contributes to timeout disabling

Bit 28 **RX_CS_TIMEOUT_MASK**:

0: CS flag does not contribute to timeout disabling

1: CS flag contributes to timeout disabling

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:0 **RX_TIMEOUT[22:0]**: RX timer timeout (relative duration in interpolated absolute time unit)

29.6.7 DATABUFFER_THR register (DATABUFFER_THR)

Address offset: 0x0018

Reset value: 0x0000 0000

DATABUFFER_THR register

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

Bits 31:16 **TX_ALMOST_EMPTY_THR[15:0]**: Almost Empty threshold for TX Data Buffers.
TX_ALMOST_EMPTY_F flag is raised when
CURRENT_DATABUFFER_COUNT = TX_ALMOST_EMPTY_THR.
Granularity of the threshold is in byte.

Bits 15:0 **RX_ALMOST_FULL_THR[15:0]**: Almost Full threshold for RX Data Buffers
RX_ALMOST_FULL_F flag is raised when
CURRENT_DATABUFFER_COUNT = RX_ALMOST_FULL_THR
Granularity of the threshold is in byte.

29.6.8 RFSEQ_IRQ_ENABLE register (RFSEQ_IRQ_ENABLE)

Address offset: 0x001C

Reset value: 0x0000 0000

RFSEQ_IRQ_ENABLE register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AGC_CALIB_DONE_E	SAFEASK_CALIB_DONE_E	Res.	RRM_CMD_END_E	RRM_CMD_START_E	SEQ_E	Res.	HW_ANA_FAILURE_E	Res.	AHB_ACCESS_ERROR_E	TX_ALMOST_EMPTY_1_E	TX_ALMOST_EMPTY_0_E	RX_ALMOST_FULL_1_E	RX_ALMOST_FULL_0_E	DATABUFFER1_USED_E	DATABUFFER0_USED_E
rw	rw		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	SYNC_VALID_E	PREAMBLE_VALID_E	CS_E	Res.	Res.	COMMAND_REJECTED_E	SABORT_DONE_E	RXTIMER_STOP_CDT_E	Res.	Res.	FAST_RX_TERM_E	RX_CRC_ERROR_E	RX_TIMEOUT_E	RX_OK_E	TX_DONE_E
	rw	rw	rw			rw	rw	rw			rw	rw	rw	rw	rw

Bit 31 **AGC_CALIB_DONE_E**: Enable interrupt on AGC_CALIB_DONE_F flag

Bit 30 **SAFEASK_CALIB_DONE_E**: Enable interrupt on SAFEASK_CALIB_DONE_F flag

Bit 29 Reserved, must be kept at reset value.

Bit 28 **RRM_CMD_END_E**: Enable interrupt on RRM_CMD_END_F flag

Bit 27 **RRM_CMD_START_E**: Enable interrupt on RRM_CMD_END_F flag

Bit 26 **SEQ_E**: Enable interrupt on SEQ_F flag

Bit 25 Reserved, must be kept at reset value.

Bit 24 **HW_ANA_FAILURE_E**: Enable interrupt on HW_ANA_FAILURE_F flag

Bit 23 Reserved, must be kept at reset value.

Bit 22 **AHB_ACCESS_ERROR_E**: Enable interrupt on AHB_ACCESS_ERROR_F flag

Bit 21 **TX_ALMOST_EMPTY_1_E**: Enable interrupt on TX_ALMOST_EMPTY_1_F flag

Bit 20 **TX_ALMOST_EMPTY_0_E**: Enable interrupt on TX_ALMOST_EMPTY_0_F flag

Bit 19 **RX_ALMOST_FULL_1_E**: Enable interrupt on RX_ALMOST_FULL_1_F flag

Bit 18 **RX_ALMOST_FULL_0_E**: Enable interrupt on RX_ALMOST_FULL_0_F flag

Bit 17 **DATABUFFER1_USED_E**: Enable interrupt on DATABUFFER1_USED_F flag

Bit 16 **DATABUFFER0_USED_E**: Enable interrupt on DATABUFFER0_USED_F flag

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYNC_VALID_E**: Enable interrupt on SYNC_VALID_F flag

Bit 13 **PREAMBLE_VALID_E**: Enable interrupt on PREAMBLE_VALID_F flag

Bit 12 **CS_E**: Enable interrupt on CS_F flag

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **COMMAND_REJECTED_E**: Enable interrupt on COMMAND_REJECTED flag

Bit 8 **SABORT_DONE_E**: Enable interrupt on SABORT command treated and done flag

Bit 7 **RXTIMER_STOP_CDT_E**: Enable interrupt on RXTIMER_STOP_CDT_F flag

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **FAST_RX_TERM_E**: Enable interrupt on FAST_RX_TERM_F flag

Bit 3 **RX_CRC_ERROR_E**: Enable interrupt on RX_CRC_ERROR_F flag

Bit 2 **RX_TIMEOUT_E**: Enable interrupt on RX_TIMEOUT_F flag

Bit 1 **RX_OK_E**: Enable interrupt on RX_OK_F flag

Bit 0 **TX_DONE_E**: Enable interrupt on TX_DONE_F flag

29.6.9 ADDITIONAL_CTRL register (ADDITIONAL_CTRL)

Address offset: 0x0020

Reset value: 0x0003 8800

ADDITIONAL_CTRL register

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

Bit 31 **AS_ENABLE**: Enable the antenna switching feature.

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:20 **TIME_CAPTURESEL[2:0]**: Select the trigger event to capture the interpolated absolute time in the TIME_CAPTURE[31:0] register

000: feature is disabled

001: the absolute time is latched on end of transmission information

010: the absolute time is latched on end of reception information whatever the CRC result (RX OK or CRC error)

011: the absolute time is latched on the SYNC word detection event

1xx: reserved for future used

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **PA_FC[1:0]**: Power control bandwidth selection according data rate

00: BW: 12.5 kHz

01: BW: 25.0 kHz

10: BW: 50.0 kHz

11: BW: 100 kHz (default)

Bits 15:8 **CH_SPACING[7:0]**: Channel spacing. From ~732 Hz to ~187 kHz in 732 Hz steps - default: 100 kHz.

Bits 7:0 **CH_NUM[7:0]**: Channel number. This value is multiplied by the channel spacing and added to the synthesizer base frequency to generate the actual RF carrier frequency

29.6.10 FAST_RX_TIMER register (FAST_RX_TIMER)

Address offset: 0x0024

Reset value: 0x0000 0000

FAST_RX_TIMER register

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

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

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

Bit 8 **FAST_CS_TERM_EN**: Enable the Fast RX Termination feature
- 0: Fast Termination is disabled
- 1: Fast Termination is enabled

Bits 7:0 **FAST_RX_TIMEOUT[7:0]**: Fast RX termination timer value (corresponding to the delay to measure the RSSI and to let the HW check CS flag information)
Timer unit is: (2**CHFLT_E x 8) / fsys_freq with fsys = fXO / 3 (typically 16 MHz)

29.6.11 COMMAND register (COMMAND)

Address offset: 0x0028

Reset value: 0x0000 0000

COMMAND register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	BACK2LOCKON	BACK2ACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw									

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

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **BACK2LOCKON**: Request to the Radio FSM to stay in LOCKON state when exiting a RX or a TX

0: no request to go to LOCKON state, the Radio FSM has to check the BACK2ACTIVE2 bit.

1: Request to go to LOCKON state (BACK2ACTIVE2 bit value is not considered).

Warning: this bit must be set only when a specific use-case requires it. The SW is responsible then to request compatible commands from this default state as the HW does not raise any COMMAND_REJECTED flag (for example, if a TX command is requested from a LOCKONRX state)

Note: if the started command combined with BACK2LOCKON=1 is aborted before to reach LOCKONxX state, the final state of the Radio FSM is ACTIVE2 or IDLE (depending on BACK2ACTIVE2 bit)

Bit 25 **BACK2ACTIVE**: Select the default/return state for the Radio FSM to be ACTIVE2

0: no request to go to / stay in ACTIVE2 state. If BACK2LOCK is also low, then the Radio FSM mechanically goes back to IDLE once ACTIVE2 is reached.

1: Request to go in ACTIVE2 state (and stay until the bit is cleared or a new TX/TX/LOCK activity is started).

Note: this bit can be used to bring the Radio FSM from IDLE to ACTIVE2 state without issuing any specific command.

Bits 24:4 Reserved, must be kept at reset value.

Bits 3:0 **COMMAND_ID[3:0]**: Opcode coresponding to a command:

Possible OpCodes are:

- NOP = no operation (Opcode = 0x0)
- TX (OpCode = 0x1)
- RX (OpCode = 0x2)
- LOCKRX (OpCode = 0x3)
- LOCKTX (OpCode = 0x4)
- SABORT (OpCode = 0x5)
- RELOAD_RX_TIMER (OpCode = 0x6)
- CALIB_SAFEASK (OpCode = 0xA)
- CALIB_AGC (OpCode = 0xB)

29.6.12 DYNAMIC_REG_BLOCK register map

Table 123. DYNAMIC_REG_BLOCK register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PCKTLEN_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCKTLEN[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
0x004	MOD0_CONFIG	PA_CLKON_LOCKONTX	Res.	Res.	Res.	Res.	BT_SEL	FOUR_GFSK_CONST_MAP[1:0]		Res.	MOD_TYPE[2:0]			DATARATE_E[3:0]				DATARATE_M[15:0]															
	Reset value	0					0	0	0		0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1
0x008	MOD1_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHFLT_E[3:0]				CHFLT_M[3:0]				Res.	Res.	Res.	Res.	FDEV_E[3:0]				FDEV_M[7:0]							
	Reset value									0	1	0	0	0	0	0	0					0	1	0	0	0	0	1	1	0	1	0	1
0x00C	SYNTH_FREQ	Res.	BS	Res.	Res.	SYNTH_INT[7:0]								SYNTH_FRAC[19:0]																			
	Reset value		0			0	1	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	1	1	0	0	0	0	1	0	1	0	1
0x010	VCO_CAL_CONFIG	VCO_CALIB_REQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALFREQ_EXT_SEL	VCO_CALFREQ_EXT[6:0]							VCO_CALAMP_EXT_SEL	Res.	VCO_CALAMP_EXT[13:0]													
	Reset value	0								0	1	0	0	0	0	0	0	0		0	0	0	0	0	0	1	0	0	0	1	0	0	0
0x014	RX_TIMER	RX_OR_NAND_SELECT	RX_SQI_TIMEOUT_MASK	RX_PQI_TIMEOUT_MASK	RX_CS_TIMEOUT_MASK	Res.	Res.	Res.	Res.	Res.	RX_TIMEOUT[22:0]																						
	Reset value	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	DATABUFFER_THR	TX_ALMOST_EMPTY_THR[15:0]																RX_ALMOST_FULL_THR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 123. DYNAMIC_REG_BLOCK register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	RFSEQ_IRQ_ENABLE	AGC_CALIB_DONE_E	SAFEASK_CALIB_DONE_E	Res.	RRM_CMD_END_E	RRM_CMD_START_E	SEQ_E	Res.	HW_ANA_FAILURE_E	Res.	AHB_ACCESS_ERROR_E	TX_ALMOST_EMPTY_1_E	TX_ALMOST_EMPTY_0_E	RX_ALMOST_FULL_1_E	RX_ALMOST_FULL_0_E	DATABUFFER1_USED_E	DATABUFFER0_USED_E	Res.	SYNC_VALID_E	PREAMBLE_VALID_E	CS_E	Res.	Res.	COMMAND_REJECTED_E	SABORT_DONE_E	RXTIMER_STOP_CDT_E	Res.	Res.	FAST_RX_TERM_E	RX_CRC_ERROR_E	RX_TIMEOUT_E	RX_OK_E	TX_DONE_E
	Reset value	0	0		0	0	0		0		0	0	0	0	0	0	0		0	0	0			0	0	0			0	0	0	0	0
0x020	ADDITIONAL_CTRL	AS_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIME_CAPTURESEL[2:0]			Res.	Res.	PA_FC[1:0]		CH_SPACING[7:0]								CH_NUM[7:0]							
	Reset value	0									0	0	0			1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0x024	FAST_RX_TIMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FAST_CS_TERM_EN	FAST_RX_TIMEOUT[7:0]							
	Reset value																								0	0	0	0	0	0	0	0	0
0x028	COMMAND	Res.	Res.	Res.	Res.	Res.	BACK2LOCKON	BACK2ACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMMAND_ID[3:0]			
	Reset value						0	0																						0	0	0	0

29.7 READ-ONLY_REG_BLOCK register descriptions

The **STATUS_REG_BLOCKBaseAddress** keyword used for all registers base address information corresponds to Global registers base address+ 0x200 (with Global registers base address decided by the SoC when integrating the IP).

*Note: **STATUS_REG_BLOCKBaseAddress** is 0x4900_0600 in the STM32WL30xx/31xx/33xx.*

Table 124. READ-ONLY_REG_BLOCK register list

Offset	Register Name	Register Title	Reset
0x0000	RFSEQ_IRQ_STATUS	RFSEQ_IRQ_STATUS register	0x0000 0000
0x0004	RFSEQ_STATUS_DETAIL	RFSEQ_STATUS_DETAIL register	0x0000 0000
0x0008	RADIO_FSM_INFO	RADIO_FSM_INFO register	0x0000 0000
0x000C	RX_INDICATOR	RX_INDICATOR register	0x0000 0000
0x0010	RX_INFO_REG	RX_INFO_REG register	0x0000 0000
0x0014	RX_CRC_REG	RX_CRC_REG register	0x0000 0000
0x0018	QI_INFO	QI_INFO register	0x0000 0000
0x001C	DATABUFFER_INFO	DATABUFFER_INFO register	0x0000 0000
0x0020	TIME_CAPTURE	TIME_CAPTURE register	0x0000 0000
0x0024	IQC_CORRECTION_OUT	IQC_CORRECTION_OUT register	0x0000 0000
0x0028	PA_SAFEASK_OUT	PA_SAFEASK_OUT register	0x0000 0000
0x002C	VCO_CALIB_OUT	VCO_CALIB_OUT register	0x0000 FF40
0x0030	SEQ_INFO	SEQ_INFO register	0x0000 0000
0x0034	SEQ_EVENT_STATUS	SEQ_EVENT_STATUS register	0x0000 0000

29.7.1 RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)

Address offset: 0x0000

Reset value: 0x0000 0000

RFSEQ_IRQ_STATUS register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AGC_CALIB_DONE_F	SAFEASK_CALIB_DONE_F	Res.	RRM_CMD_END_F	RRM_CMD_START_F	SEQ_F	Res.	HW_ANA_FAILURE_F	Res.	AHB_ACCESS_ERROR_F	TX_ALMOST_EMPTY_1_F	TX_ALMOST_EMPTY_0_F	RX_ALMOST_FULL_1_F	RX_ALMOST_FULL_0_F	DATABUFFER1_USED_F	DATABUFFER0_USED_F
rc_w1	rc_w1		rc_w1	rc_w1	rc_w1		rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SYNC_VALID_F	PREAMBLE_VALID_F	CS_F	Res.	Res.	COMMAND_REJECTED_F	SABORT_DONE_F	RXTIMER_STOP_CDT_F	Res.	Res.	FAST_RX_TERM_F	RX_CRC_FRROR_F	RX_TIMEOUT_F	RX_OK_F	TX_DONE_F
	rc_w1	rc_w1	rc_w1			rc_w1	rc_w1	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 31 **AGC_CALIB_DONE_F**: Valid RSSI value available in the RSSI_RUNNING bit field flag.

This flag is raised after an AGC_AAF_CALIB command to inform the command is over and the RSSI measurement result is available in RX_INDICATOR status register.

Bit 30 **SAFEASK_CALIB_DONE_F**: End of Safe-ASK PA calibration flag.

This flag is raised after a CALIB_SAFEASK command to inform the command is over and the PA_CODEMAX result is available in PA_SAFEASK_OUT status register.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **RRM_CMD_END_F**: RRM-UDRA command list execution ended flag.

When set, indicates a Radio register restoring phase has been completed by the RRM UDRA

Bit 27 **RRM_CMD_START_F**: RRM-UDRA command list execution started flag.

When set, indicates a Radio register restoring phase has been started by the RRM UDRA

Bit 26 **SEQ_F**: Sequencer completion flag.

When set, indicates the Sequencer has ended a SeqAction (thanks to a next action match or an ActionTimeout event). The end reason can be found in the annex register called RFSEQ_STATUS_DETAIL.

Note: clearing this bit disables the interrupt signal towards the CPU but has no impact on RFSEQ_STATUS_DETAILS bits. The SW must also clear them in parallel.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **HW_ANA_FAILURE_F**: Analog HW failure flag (PLL lock / unlock error, calibration error)

Complementary information available in the annex register called RFSEQ_STATUS_DETAIL.

Note: clearing this bit disables the interrupt signal towards the CPU but has no impact on RFSEQ_STATUS_DETAILS bits. The SW must also clear them in parallel.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **AHB_ACCESS_ERROR_F**: An AHB transfer issue occurred for one of the AHB masters (RRM, Data Buffer Manager, Sequencer). Details in an annex register = RFSEQ_STATUS_DETAIL.

Note: clearing this bit disables the interrupt signal towards the CPU but has no impact on RFSEQ_STATUS_DETAILS bits. The SW must also clear them in parallel.

Note: in the STM32WL30xx/31xx/33xx, this flag can be set only by the DBM_FIFO_ERROR_F

Bit 21 **TX_ALMOST_EMPTY_1_F**: Data Buffer1 used (read during a TX) up to programmed thresold flag

Bit 20 **TX_ALMOST_EMPTY_0_F**: Data Buffer0 used (read during a TX) up to programmed thresold flag

Bit 19 **RX_ALMOST_FULL_1_F**: Data Buffer1 used (written during a RX) up to programmed thresold flag

Bit 18 **RX_ALMOST_FULL_0_F**: Data Buffer0 used (written during a RX) up to programmed thresold flag

Bit 17 **DATABUFFER1_USED_F**: Data Buffer 1 fully read in TX or fully written in RX flag

Bit 16 **DATABUFFER0_USED_F**: Data Buffer 0 fully read in TX or fully written in RX flag

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYNC_VALID_F**: Valid SYNC word detection flag.

Note: this flag is never raised in Direct through Buffer/GPIO RX modes

Bit 13 **PREAMBLE_VALID_F**: Valid PREAMBLE detection flag.

Note: this flag is never raised in Direct through Buffer/GPIO RX modes

Bit 12 **CS_F**: Carrier Sense (RSSI over threshold) flag

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **COMMAND_REJECTED_F**: Command rejection flag.

Indicates the last requested command has been rejected because not compatible with an already on-going command (or just after a SABORT where the Radio FSM is not yet back to default targeted state = LOCKON, ACTIVE2 or IDLE)

Bit 8 **SABORT_DONE_F**: SABORT command treated and done flag

Bit 7 **RXTIMER_STOP_CDT_F**: Enable interrupt on RXTIMER_STOP_CDT_F flag

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **FAST_RX_TERM_F**: Fast RX Termination flag

Bit 3 **RX_CRC_FRROR_F**: Reception with CRC error flag

Bit 2 **RX_TIMEOUT_F**: Reception timeout flag

Bit 1 **RX_OK_F**: Reception ended and OK flag

Bit 0 **TX_DONE_F**: Transmission done flag

29.7.2 RFSEQ_STATUS_DETAIL register (RFSEQ_STATUS_DETAIL)

Address offset: 0x0004

Reset value: 0x0000 0000

RFSEQ_STATUS_DETAIL register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SEQ_COMPLETE_F	SEQ_ACTIONTIMEOUT_F	Res.	Res.	PLL_CALAMP_ERROR_F	PLL_CALFREQ_ERROR_F	PLL_UNLOCK_F	PLL_LOCK_FAIL_F	Res.	Res.	DBM_FIFO_ERROR_F	Res.	Res.	Res.	Res.	Res.
rc_w1	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1			rc_w1					

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

Bit 15 **SEQ_COMPLETE_F**: The Sequencer has ended the last defined SeqAction properly(NextAction math or null pointer)

Bit 14 **SEQ_ACTIONTIMEOUT_F**: The Sequencer has ended because the current SeqAction reached its ActionTimeout.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **PLL_CALAMP_ERROR_F**: VCO amplitude calibration error flag

Bit 10 **PLL_CALFREQ_ERROR_F**: VCO frequency calibration error flag

Bit 9 **PLL_UNLOCK_F**: PLL unlock event flag

Bit 8 **PLL_LOCK_FAIL_F**: PLL lock fail status flag

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

Bit 5 **DBM_FIFO_ERROR_F**: Data Buffer Manager internal FIFO overflow/underflow flag.

The Data Buffer Manager did not manage to write (for RX) or read (for TX) in RAM on time probably due to lack of bandwitdh on the SoC interconnect

or the DATABUFFER_SIZE bit field was null when an AHB transfer is requested (for RX or TX)

Bits 4:0 Reserved, must be kept at reset value.

29.7.3 RADIO_FSM_INFO register (RADIO_FSM_INFO)

Address offset: 0x0008

Reset value: 0x0000 0000

RADIO_FSM_INFO register

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

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

Bits 4:0 **RADIO_FSM_STATE[4:0]**: State of the Radio FSM
0x00: IDLE
0x01: ENA_RF_REG
0x02: WAIT_ACTIVE2
0x03: ACTIVE2
0x04: ENA_CURR
0x05: SYNTH_SETUP
0x06: CALIB_VCO
0x07: LOCKRXTX
0x08: LOCKONTX
0x09: EN_PA
0x0A: TX
0x0B: PA_DWN_ANA
0x0C: END_TX
0x0D: LOCKONRX
0x0E: EN_RX
0x0F: EN_LNA
0x10: RX
0x11: END_RX
0x12: SYNTH_PWDN

29.7.4 RX_INDICATOR register (RX_INDICATOR)

Address offset: 0x000C

Reset value: 0x0000 0000

RX_INDICATOR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ANT_SELECT	Res.	Res.	Res.	AGC_WORD[3:0]				Res.	Res.	Res.	RSSI_LEVEL_RUN[8:4]				
r				r	r	r	r				r	r	r	r	r

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

Bit 31 **ANT_SELECT**: Currently selected antenna

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

Bits 27:24 **AGC_WORD[3:0]**: AGC word of the received packet.
This bit field indicates the number of steps of attenuation applied on the received packet.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:12 **RSSI_LEVEL_RUN[8:0]**: Continuous level of the output of the measured RSSI value

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

Bits 8:0 **RSSI_LEVEL_ON_SYNC[8:0]**: RSSI level captured at the end of the SYNC word detection of the received packet.

29.7.5 RX_INFO_REG register (RX_INFO_REG)

Address offset: 0x0010

Reset value: 0x0000 0000

RX_INFO_REG register

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

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

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

Bits 15:0 **RX_PCKTLEN_OUT[15:0]**: Indicates received packet length in bytes:
- PCKTLEN_CONFIG[15:0] value for reception in FIXED length mode
- decoded length (inside the frame) for reception with VARIABLE length mode

29.7.6 RX_CRC_REG register (RX_CRC_REG)

Address offset: 0x0014

Reset value: 0x0000 0000

RX_CRC_REG register

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

Bits 31:0 **RX_CRC_OUT[31:0]**: CRC field of the received packet (read-only info)

29.7.7 QI_INFO register (QI_INFO)

Address offset: 0x0018

Reset value: 0x0000 0000

QI_INFO register

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

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

Bits 23:16 **AFC_CORRECTION[7:0]**: AFC value frozen at sync reception.
Unit: fsys/2^14 = around 976.56Hz with typical 16 MHz

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SQI_SEC**: Indicate if measured SQI refers to SYNC word or secondary SYNC word
0: SQI value versus the SYNC word
1: SQI value versus the secondary SYNC word

Bits 13:8 **SQI_INFO[5:0]**: SYNC Quality Indicator (SQI) value of the received packet. Represents the peak value between 0 and 32.

Bits 7:0 **PQI_INFO[7:0]**: Preamble Quality Indicator (PQI) value of the received packet.

29.7.8 DATABUFFER_INFO register (DATABUFFER_INFO)

Address offset: 0x001C

Reset value: 0x0000 0000

DATABUFFER_INFO register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRENT_DATABUFFER	NB_DATABUFFER_USED[14:0]														
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 31 **CURRENT_DATABUFFER**: Indicates which Data Buffer is currently used by the HW

0 = Data Buffer0

1 = Data Buffer1

Bits 30:16 **NB_DATABUFFER_USED[14:0]**: Provides the number of data buffers which have been fully used

- written during RX
- read during TX

Bits 15:0 **CURRENT_DATABUFFER_COUNT[15:0]**: Indicates the number of bytes used in the last used DATA BUFFER.

This information has to be added to the (NB_DATABUFFER_USED x DATABUFFER_SIZE) to know the total bytes number read/written in RAM:

- in RX, it corresponds to number of received bytes copied in RAM
- in TX, it represents the number of bytes read from RAM (not necessarily taken by the Transmit block and transmitted on the antenna).

29.7.9 TIME_CAPTURE register (TIME_CAPTURE)

Address offset: 0x0020

Reset value: 0x0000 0000

TIME_CAPTURE register

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

Bits 31:0 **TIME_CAPTURE[31:0]**: Interpolated absolute time value captured on specific programmable event through TIME_CAPTURESEL[2:0] bit field.

29.7.10 IQC_CORRECTION_OUT register (IQC_CORRECTION_OUT)

Address offset: 0x0024

Reset value: 0x0000 0000

IQC_CORRECTION_OUT register

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

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

Bits 23:0 **IQC_CORRECT_OUT[23:0]**: Final correction value output from IQC (compensation engine).
This value is updated at the end of a RX sequence or on a LOAD_IQC_INIT command.
Note: this value is not retained so the SW has to copy it in a retained or non-volatile memory to be able to reuse it after a low power mode session.

29.7.11 PA_SAFEASK_OUT register (PA_SAFEASK_OUT)

Address offset: 0x0028

Reset value: 0x0000 0000

PA_SAFEASK_OUT register

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

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

Bits 7:0 **PA_CODEMAX[7:0]**: Safe ASK level (provided after a CALIB_SAFEASK command), indicates the maximum PA Power to program before the PA enters the ohmic zone.

The result format depends on PA_CONFIG[13] = LIN_NLOG bit value.

If LIN_NLOG=0: result is provided in dB

- bit[7] = REACHMAX

- when 0: the calibration detected the PA enters the ohmic zone before it reaches the maximum requested power; the result is less than the PA_LEVEL7[7:0] value

- when 1: the maximum power targeted (PA_LEVEL7[7:0] bit field) has been reached without the PA entering the ohmic zone

- bit[6:0] = Maximum PA power before the PA enters the ohmic zone (expressed logarithmically)

if LIN_NLOG=1: result is provided in linear format

- bit[7:0] = maximum PA power before the PA enters the ohmic zone (expressed linearly)

Note: when expressed in linear scale, only the 8 MSbs out of 9-bits are provided in this register

29.7.12 VCO_CALIB_OUT register (VCO_CALIB_OUT)

Address offset: 0x002C

Reset value: 0x0000 FF40

VCO_CALIB_OUT register

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

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:8 **VCO_CALAMP_OUT[13:0]**: VCO amplitude calibration value currently output by the VCO calibration block (and applied on the VCO when ON)

Note: if VCO_CAL_CONFIG[15]=VCO_CALAMP_EXT=1, the value read here is equal to the value written in VCO_CAL_CONFIG[13:0]=VCO_CALAMP_EXT[13:0]

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **VCO_CALFREQ_OUT[6:0]**: VCO frequency calibration value currently output by the VCO calibration block (and applied on the VCO when ON)

Note: if VCO_CAL_CONFIG[23]=VCO_CALFREQ_EXT=1, the value read here is equal to the value written in VCO_CAL_CONFIG[22:16]=VCO_CALFREQ_EXT[6:0]

29.7.13 SEQ_INFO register (SEQ_INFO)

Address offset: 0x0030

Reset value: 0x0000 0000

SEQ_INFO register

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

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

Bits 4:0 **SEQ_FSM_STATE[4:0]**: Current state of the Sequencer

0x00: WAIT_TRIGGER_W_SLEEP_EN
0x01: WAIT_TRIGGER_WO_SLEEP_EN
0x02: EXTERNAL_TRIGGER
0x03: GLOBALCONFIG_W0_AND_W1
0x04: GLOBALCONFIG_STATIC_FIELDS_W2
0x05: GLOBALCONFIG_STATIC_FIELDS
0x06: ACTIONCONFIG_ACTION1_MASK
0x07: ACTIONCONFIG_ACTION2_MASK
0x08: ACTIONCONFIG_DYNAMIC_FIELDS_W9
0x09: ACTIONCONFIG_DYNAMIC_FIELDS
0x0A: ACTIONCONFIG_W8
0x0B: ACTIVE_WAIT
0x0C: NEXT_ACTION1_TRIGGER
0x0D: NEXT_ACTION1_CONTROL_W3
0x0E: NEXT_ACTION1_CONTROL_WRITE_W0_AND_W1
0x0F: NEXT_ACTION1_CONTROL_W1
0x10: NEXT_ACTION2_TRIGGER
0x11: NEXT_ACTION2_CONTROL_W7
0x12: NEXT_ACTION2_CONTROL_WRITE_W0_AND_W1
0x13: NEXT_ACTION2_CONTROL_W5
0x14: NEXT_ACTION_TRIGGER_PROGRAM
0x15: ABORTING
0x16: EXTERNAL_TRIGGER_ABORTING

29.7.14 SEQ_EVENT_STATUS register (SEQ_EVENT_STATUS)

Address offset: 0x0034

Reset value: 0x0000 0000

SEQ_EVENT_STATUS register

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

Bits 31:0 **SEQ_EVENT_STATUS[31:0]**: Current value of the seq_event_status used by the Sequencer for next action mask comparison. It contains both recorded events (TX_DONE, RX_OK) and real time events (Data buffer flags, RX indicator)

Reminder: the mapping of this bus is aligned on RFSEQ_IRQ_STATUS register.

29.7.15 READ-ONLY_REG_BLOCK register map and reset values

Table 125. READ-ONLY_REG_BLOCK register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RFSEQ_IRQ_STATUS	AGC_AAF_CALIB_DONE_F	SAFEASK_CALIB_DONE_F	Res.	RRM_CMD_END_F	RRM_CMD_START_F	SEQ_F	Res.	HW_ANA_FAILURE_F	Res.	AHB_ACCESS_ERROR_F	TX_ALMOST_EMPTY_1_F	TX_ALMOST_EMPTY_0_F	RX_ALMOST_FULL_1_F	RX_ALMOST_FULL_0_F	DATABUFFER1_USED_F	DATABUFFER0_USED_F	Res.	SYNC_VALID_F	PREAMBLE_VALID_F	CS_F	Res.	Res.	COMMAND_REJECTED_F	SABORT_DONE_F	RXTIMER_STOP_CDT_F	Res.	Res.	FAST_RX_TERM_F	RX_CRC_ERROR_F	RX_TIMEOUT_F	RX_OK_F	TX_DONE_F
	Reset value	0	0		0	0	0		0		0	0	0	0	0	0	0		0	0	0			0	0	0			0	0	0	0	0
0x004	RFSEQ_STATUS_DETAIL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQ_COMPLETE_F	SEQ_ACTIONTIMEOUT_F	Res.	Res.	PLL_CALAMP_ERROR_F	PLL_CALFREQ_ERROR_F	PLL_UNLOCK_F	PLL_LOCK_FAIL_F	Res.	Res.	DBM_FIFO_ERROR_F	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0			0	0	0	0			0					
0x008	RADIO_FSM_INFO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADIO_FSM_STATE[4:0]				
	Reset value																												0	0	0	0	0
0x00C	RX_INDICATOR	ANT_SELECT	Res.	Res.	Res.	AGC_WORD[3:0]				Res.	Res.	Res.	RSSI_LEVEL_RUN[8:0]									Res.	Res.	Res.	RSSI_LEVEL_ON_SYNC[8:0]								
	Reset value	0				0	0	0	0				0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0
0x010	RX_INFO_REG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_PCKTLEN_OUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	RX_CRC_REG	RX_CRC_OUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 125. READ-ONLY_REG_BLOCK register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	QI_INFO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFC_CORRECTION[7:0]								Res.	SQI_SEC	SQI_INFO[5:0]						PQI_INFO[7:0]							
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	DATABUFFER_INFO	CURRENT_DATABUFFER	NB_DATABUFFER_USED[14:0]															CURRENT_DATABUFFER_COUNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TIME_CAPTURE	TIME_CAPTURE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	IQC_CORRECTION_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IQC_CORRECT_OUT[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	PA_SAFEASK_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PA_CODEMAX[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C	VCO_CALIB_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALAMP_OUT[13:0]														Res.	VCO_CALFREQ_OUT[6:0]						
	Reset value											0	0	0	0	0	0	1	1	1	1	1	1	1	1		1	0	0	0	0	0	0
0x030	SEQ_INFO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQ_FSM_STATE[4:0]				
	Reset value																												0	0	0	0	0
0x034	SEQ_EVENT_STATUS	SEQ_EVENT_STATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

29.8 MISC_REG_BLOCK register descriptions

The **MISC_REG_BLOCKBaseAddress** keyword used for all registers base address information corresponds to Global registers base address+ 0x300 (with Global registers base address decided by the SoC when integrating the IP).

*Note: **MISC_REG_BLOCKBaseAddress** is 0x4900_0700 in the STM32WL30xx/31xx/33xx.*

Table 126. MISC_REG_BLOCK register list

Offset	Register Name	Register Title	Reset
0x0000	RFIP_VERSION	RFIP_VERSION register	0x0000 1200
0x0004	RRM_UDRA_CTRL	RRM_UDRA_CTRL register	0x0000 0000
0x0008	SEQUENCER_CTRL	SEQUENCER_CTRL register	0x0000 0000
0x000C	ABSOLUTE_TIME	ABSOLUTE_TIME register	0x0000 0000
0x0010	SCM_COUNTER_VAL	SCM_COUNTER_VAL register	0x0000 0000
0x0014	SCM_MIN_MAX	SCM_MIN_MAX register	0x0000 7FFF
0x0018	WAKEUP_IRQ_STATUS	WAKEUP_IRQ_STATUS register	0x0000 0000

29.8.1 RFIP_VERSION register (RFIP_VERSION)

Address offset: 0x0000

Reset value: 0x0000 1200

RFIP_VERSION register

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

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

Bits 15:12 **PRODUCT[3:0]**: Used for major upgrades (new protocols support / new features)

Bits 11:8 **VERSION[3:0]**: Version of the MR_SubG (to be used for cut upgrades)

Bits 7:4 **REVISION[3:0]**: Revision of the MR_SubG (to be used for metal fixes)

Bits 3:0 Reserved, must be kept at reset value.

29.8.2 RRM_UDRA_CTRL register (RRM_UDRA_CTRL)

Address offset: 0x0004

Reset value: 0x0000 0000

RRM_UDRA_CTRL register

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

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

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

Bit 0 **RRM_CMD_REQ**: Action bit.

Write 1 to request a RRM-UDRA command. The RRM checks the RRM_CMDLIST_PTR[31] = CMDLIST_VALID bit and, if set, starts fetching the command in RAM.

Note: this bit is immediately cleared by HW.

29.8.3 SEQUENCER_CTRL register (SEQUENCER_CTRL)

Address offset: 0x0008

Reset value: 0x0000 0000

SEQUENCER_CTRL register

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

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

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

Bit 1 **DISABLE_SEQ**: Enable/disable the Sequencer

0: enable the Sequencer (default). The Sequencer starts in the case of a trigger event

1: disable the Sequencer (including a potential on-going sequence) and mask any further trigger events

This bit is cleared by the HW (Sequencer) once the potential on-going sequence is effectively aborted

Note: from cut 2, there is no more need to set GEN_SEQ_TRIGGER bit with DISABLE_SEQ (cut 1 bug fixed)

Bit 0 **GEN_SEQ_TRIGGER**: Action bit.

Write 1 to generate a trigger event on Sequencer.

Note: this bit is immediately cleared by HW.

29.8.4 ABSOLUTE_TIME register (ABSOLUTE_TIME)

Address offset: 0x000C

Reset value: 0x0000 0000

ABSOLUTE_TIME register

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

Bits 31:0 **ABSOLUTE_TIME[31:0]**: Indicate the interpolated absolute.

The time value is expressed in 16 x slow clock periods unit (typically 512 kHz).

It is called "interpolated" as it is built with the 28 MSbit based on the slock clock and the 4 LSbit interpolated thanks to the fast system clock.

Warning: this register is

- read at 0 when the SoC system clock tree is not (/no more) based on accurate clock

- restored to a meaningful value 3 slow clock cycles after the Soc system clock tree is based (again) on an accurate clock

Reminder: in this document

- the absolute time represents only the 28 MSbit information (without interpolated part)
- the interpolated absolute time represents the 32-bit information (with the interpolated part)

29.8.5 SCM_COUNTER_VAL register (SCM_COUNTER_VAL)

Address offset: 0x0010

Reset value: 0x0000 0000

SCM_COUNTER_VAL register

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **SCM_COUNTER_CURRVAL[14:0]**: Slow Clock Measurement: number of 16 MHz clock cycles contained in 32 slow clock periods.

This bit field is updated every 32 slow clock periods.

29.8.6 SCM_MIN_MAX register (SCM_MIN_MAX)

Address offset: 0x0014

Reset value: 0x0000 7FFF

SCM_MIN_MAX register

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

Bit 31 **CLEAR_MIN_MAX**: Write '1'to clear the SCM_COUNTER_MINVAL and SCM_COUNTER_MAXVAL bit fields.

Note: this bit is immediately cleared by the HW.

Bits 30:16 **SCM_COUNTER_MAXVAL[14:0]**: Slow Clock Measurement: maximum SCM_COUNTER value seen since the counter is ON and since last clear request.

This bit field is updated every 32 slow clock periods.

Note: 0x0000 in this bit field means no measurement done yet since last reset or last CLEAR_MIN_MAX action

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **SCM_COUNTER_MINVAL[14:0]**: Slow Clock Measurement: minimum SCM_COUNTER value seen since the counter is ON and since last clear request.

This bit field is updated every 32 slow clock periods.

Note: 0x7FFF in this bit field means no measurement done yet since last reset or last CLEAR_MIN_MAX action

29.8.7 WAKEUP_IRQ_STATUS register (WAKEUP_IRQ_STATUS)

Address offset: 0x0018

Reset value: 0x0000 0000

WAKEUP_IRQ_STATUS register

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

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

Bit 1 **RFIP_WAKEUP_F**: Set when the interpolated absolute time matches the RFIP_WAKEUPTIME while WAKEUP_CTRL.RFIP_WAKEUP_EN=1.

Note: the associated interrupt is always enabled at MR_SubG IP level. The SW must manage the Interrupt enable/disable directly through the NVIC.

Bit 0 **CPU_WAKEUP_F**: Set when the interpolated absolute time matches the CPU_WAKEUPTIME while WAKEUP_CTRL.CPU_WAKEUP_EN=1.

Note: the associated interrupt is always enabled at MR_SubG IP level. The SW must manage the Interrupt enable/disable directly through the NVIC.

29.8.8 MISC_REG_BLOCK register map

Table 127. MISC_REG_BLOCK register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RFIP_VERSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRODUCT[3:0]				VERSION[3:0]				REVISION[3:0]				Res.	Res.	Res.	Res.
	Reset value																	0	0	0	1	0	0	1	0	0	0	0	0				
0x004	RRM_UDRA_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRM_CMD_REQ
	Reset value																																0
0x008	SEQUENCER_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DISABLE_SEQ	GEN_SEQ_TRIGGER
	Reset value																															0	0
0x00C	ABSOLUTE_TIME	ABSOLUTE_TIME[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	SCM_COUNTER_VAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCM_COUNTER_CURRVAL[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	SCM_MIN_MAX	CLEAR_MIN_MAX	SCM_COUNTER_MAXVAL[14:0]															Res.	SCM_COUNTER_MINVAL[14:0]														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x018	WAKEUP_IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFIP_WAKEUP_F	CPU_WAKEUP_F
	Reset value																															0	0

29.9 RETAINED_REG_BLOCK register descriptions

Table 128. RETAINED_REG_BLOCK register list

Offset	Register Name	Register Title	Reset
0x0000	RFIP_WAKEUPTIME	RFIP_WAKEUPTIME register	0x0000 0000
0x0004	CPU_WAKEUPTIME	CPU_WAKEUPTIME register	0x0000 0000
0x0008	WAKEUP_CTRL	WAKEUP_CTRL register	0x0000 0000
0x000C	RRM_CMDLIST_PTR	RRM_CMDLIST_PTR register	0x0000 0000
0x0010	SEQ_GLOBALTABLE_PTR	SEQ_GLOBALTABLE_PTR register	0x0000 0000

The **RETAINED_REG_BLOCKBaseAddress** keyword used for all registers base address information corresponds to Global registers base address+ 0x380 (with Global registers base address decided by the SoC when integrating the IP).

*Note: **RETAINED_REG_BLOCKBaseAddress** is 0x4900_0780 in the STM32WL30xx/31xx/33xx.*

29.9.1 RFIP_WAKEUPTIME register (RFIP_WAKEUPTIME)

Address offset: 0x0000

Reset value: 0x0000 0000

RFIP_WAKEUPTIME register

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

Bits 31:0 **RFIP_WAKEUPTIME[31:0]**: (Absolute) Target time to wakeup the RFIP.
The value to program is in interpolated absolute time.
Note: the HW anticipates the wakeup request to the PWRC by subtracting the SOC_WAKEUP_OFFSET value.
This bit field is managed by HW through the Sequencer according to ActionTable info.

29.9.2 CPU_WAKEUPTIME register (CPU_WAKEUPTIME)

Address offset: 0x0004

Reset value: 0x0000 0000

CPU_WAKEUPTIME register

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

Bits 31:1 **CPU_WAKEUPTIME[30:0]**: Absolute target time to wakeup the CPU.
The value to program is in interpolated absolute time.
Note: the HW anticipates the wakeup request to the PWRC by subtracting the SOC_WAKEUP_OFFSET value.

Bit 0 Reserved, must be kept at reset value.

29.9.3 WAKEUP_CTRL register (WAKEUP_CTRL)

Address offset: 0x0008

Reset value: 0x0000 0000

WAKEUP_CTRL register

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

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

Bit 31 **RFIP_WAKEUP_EN**: Indicates if the wakeup timer has to wakeup the SoC.

(Match on RFIP_WAKEUPTIME[31:4] bit field only) + trigger an event on the Sequencer and set the RFIP_WAKEUP_F in the WAKEUP_IRQ_STATUS Misc register when match on RFIP_WAKEUPTIME[31:0] occurs.

0: no wakeup event to be generated when RFIP_WAKEUPTIME = ABSOLUTE_TIME

1: wakeup event to be generated when RFIP_WAKEUPTIME = ABSOLUTE_TIME

Note: This bit is managed by HW (through the Sequencer according to ActionTable info) and is auto cleared when a RFIP wakeup event occurs (one shot timer).

Warning: There is a latency of 2 slow clock periods between setting this bit and effectiveness of the bit. So the associated wakeup time must be more than current ABSOLUTE_TIME[31:0]+0x20.

Bit 30 **CPU_WAKEUP_EN**: Indicates if the wakeup timer has to wakeup the SoC.

(Match on CPU_WAKEUPTIME[31:4] bit field only) + set the CPU_WAKEUP_F in the WAKEUP_IRQ_STATUS Misc register when match on CPU_WAKEUPTIME[31:0] occurs.

0: no wakeup event to be generated when CPU_WAKEUPTIME = ABSOLUTE_TIME

1: wakeup event to be generated when CPU_WAKEUPTIME = ABSOLUTE_TIME

Warning: There is a latency of 2 slow clock periods between setting this bit and effectiveness of the bit. So the associated wakeup time must be more than current ABSOLUTE_TIME[31:0]+0x20.

Bits 29:8 Reserved, must be kept at reset value.

Bits 7:0 **SOC_WAKEUP_OFFSET[7:0]**: Delay to be considered by the Wakeup block to anticipate the wakeup request to the PWRC of the SoC versus the target to wakeup the RFIP (or the CPU).

Value to program is time(to_enter_sleep_mode)+ time(to_exit_sleep_mode)

Unit is in slow clock period

Note: the value programmed in this bit field must be at least the duration from a wakeup event up to accurate system clock ready information + 3 slow clock periods (as the TMU needs 3 slow clock periods after the system clock accurate ready information to restore a valid 32-bit interpolated absolute time).

29.9.4 RRM_CMDLIST_PTR register (RRM_CMDLIST_PTR)

Address offset: 0x000C

Reset value: 0x0000 0000

RRM_CMDLIST_PTR register

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

Bit 31 **CMDLIST_PTR_VALID**: Indicate if a command list has to be executed or not

0: no need to go in RAM to get commands to execute

1: a command list is present in RAM. The RRM-UDRA can use the CMDLIST_OFFSET[15:0] to fetch the command list in RAM.

Note: for reminder, RRM-UDRA checks this bit on any MR_SubG reset exit or a write 1 in the RRM_COMMAND_REQ action bit.

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

Bits 15:0 **CMDLIST_PTR_OFFSET[15:0]**: Contain the offset versus the SoC RAM base address where to find the RRM-UDRA command list entry point.

29.9.5 SEQ_GLOBALTABLE_PTR register (SEQ_GLOBALTABLE_PTR)

Address offset: 0x0010

Reset value: 0x0000 0000

SEQ_GLOBALTABLE_PTR register

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

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

Bits 15:0 **SEQ_GLOBALTABLE_PTR[15:0]**: Contain the offset versus the SoC RAM base address of the GlobalConfiguration RAM table entry point. This information is used by the Sequencer.

29.9.6 RETAINED_REG_BLOCK register map

Table 129. RETAINED_REG_BLOCK register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RFIP_WAKEUPTIME	RFIP_WAKEUPTIME[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	CPU_WAKEUPTIME	CPU_WAKEUPTIME[30:0]																															Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x008	WAKEUP_CTRL	RFIP_WAKEUP_EN	CPU_WAKEUP_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOC_WAKEUP_OFFSET[7:0]							
	Reset value	0	0																							0	0	0	0	0	0	0	0
0x00C	RRM_CMDLIST_PTR	CMDLIST_PTR_VALID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDLIST_PTR_OFFSET[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	SEQ_GLOBALTABLE_PTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQ_GLOBALTABLE_PTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

29.10 Radio register descriptions

The Radio registers are 8-bit registers mainly used to control the RFSUBG analog IP and the Radio FSM. They also allow taking control for validation/test purposes.

They can be accessed through two different mappings:

- as 32-bit APB registers (address incremented by 4 between each register) through RRM Direct Access interface,
- as 8-bit register (address incremented by 1 between each register) through RRM UDRA command list in RAM.

29.10.1 Radio register overview

The Radio registers control the parameters for the analog and few modulator/demodulator features.

The **RADIO_REG_BLOCKBaseAddress** keyword used for all registers base address information corresponds to Radio registers base address decided by the SoC when integrating the IP.

*Note: **RADIO_REG_BLOCKBaseAddress** is 0x4900_0000 in the STM32WL30xx/31xx/33xx.*

Table 130. RADIO_REG_BLOCK register list

Address Offset		Register Name	Register Title
RRM UDRA	APB		
0x00	0x0000	RF_FSM0_TIMEOUT	RF_FSM0_TIMEOUT register
0x01	0x0004	RF_FSM1_TIMEOUT	RF_FSM1_TIMEOUT register
0x02	0x0008	RF_FSM2_TIMEOUT	RF_FSM2_TIMEOUT register
0x03	0x000C	RF_FSM3_TIMEOUT	RF_FSM3_TIMEOUT register
0x04	0x0010	RF_FSM4_TIMEOUT	RF_FSM4_TIMEOUT register
0x05	0x0014	RF_FSM5_TIMEOUT	RF_FSM5_TIMEOUT register
0x06	0x0018	RF_FSM6_TIMEOUT	RF_FSM6_TIMEOUT register
0x07	0x001C	RF_FSM7_TIMEOUT	RF_FSM7_TIMEOUT register
0x08	0x0020	AFC0_CONFIG	AFC0_CONFIG register
0x09	0x0024	AFC1_CONFIG	AFC1_CONFIG register
0x0A	0x0028	AFC2_CONFIG	AFC2_CONFIG register
0x0B	0x002C	AFC3_CONFIG	AFC3_CONFIG register
0x0C	0x0030	CLKREC_CTRL0	CLKREC_CTRL0 register
0x0D	0x0034	CLKREC_CTRL1	CLKREC_CTRL1 register
0x0E	0x0038	DCREM_CTRL0	DCREM_CTRL0 register
0x0F	0x003C	DCREM_CTRL1	DCREM_CTRL1 register
0x10	0x0040	IQC_CTRL0	IQC_CTRL0 register
0x11	0x0044	IQC_CTRL1	IQC_CTRL1 register
0x12	0x0048	IQC_CTRL2	IQC_CTRL2 register
0x13	0x004C	IQC_CTRL3	IQC_CTRL3 register
0x14	0x0050	AGC_ANA_ENG	AGC_ANA_ENG register
0x15	0x0054	AGC0_CTRL	AGC0_CTRL register
0x16	0x0058	AGC1_CTRL	AGC1_CTRL register
0x17	0x005C	AGC2_CTRL	AGC2_CTRL register
0x18	0x0060	AGC3_CTRL	AGC3_CTRL register
0x19	0x0064	AGC4_CTRL	AGC4_CTRL register

Table 130. RADIO_REG_BLOCK register list

Address Offset		Register Name	Register Title
RRM UDRA	APB		
0x1A	0x0068	AGC_ATTEN0	AGC_ATTEN0 register
0x1B	0x006C	AGC_ATTEN1	AGC_ATTEN1 register
0x1C	0x0070	AGC_ATTEN2	AGC_ATTEN2 register
0x1D	0x0074	AGC_ATTEN3	AGC_ATTEN3 register
0x1E	0x0078	AGC_ATTEN4	AGC_ATTEN4 register
0x1F	0x007C	AGC_ATTEN5	AGC_ATTEN5 register
0x20	0x0080	AGC_ATTEN6	AGC_ATTEN6 register
0x21	0x0084	AGC_ATTEN7	AGC_ATTEN7 register
0x22	0x0088	AGC_ATTEN8	AGC_ATTEN8 register
0x23	0x008C	AGC_ATTEN9	AGC_ATTEN9 register
0x24	0x0090	AGC1_ATTEN_TRIM	AGC1_ATTEN_TRIM register
0x25	0x0094	AGC2_ATTEN_TRIM	AGC2_ATTEN_TRIM register
0x26	0x0098	AGC3_ATTEN_TRIM	AGC3_ATTEN_TRIM register
0x27	0x009C	AGC4_ATTEN_TRIM	AGC4_ATTEN_TRIM register
0x28	0x00A0	AGC_PGA_HWTRIM_OUT	AGC_PGA_HWTRIM_OUT register
0x2A	0x00A8	PA_REG	PA_REG register
0x2B	0x00AC	PA_HWTRIM_OUT	PA_HWTRIM_OUT register
0x2F	0x00BC	RSSI_FLT	RSSI_FLT register
0x32	0x00C8	SYNTH2_ANA_ENG	SYNTH2_ANA_ENG register
0x33 to 0x39	0x00CC to 0x00E4	Reserved	Reserved
0x3A	0x00E8	RXADC_HWDELAYTRIM_OUT	RXADC_HWDELAYTRIM_OUT register
0x3B	0x00EC	RXADC_SW_DELAYTRIM	RXADC_SW_DELAYTRIM register
0x3C	0x00F0	Reserved	Reserved
0x3D	0x00F4	RX_AAF_HWTRIM_OUT	RX_AAF_HWTRIM_OUT register
0x3E	0x00F8	RX_AAF_SWTRIM	RX_AAF_SWTRIM register
0x3F	0x00FC	Reserved	Reserved
0x40	0x0100	SINGEN_ANA_ENG	SINGEN_ANA_ENG register
0x41	0x0104	Reserved	Reserved
0x42	0x0108	RF_INFO_OUT	RF_INFO_OUT register
0x49	0x0124	RF_FSM8_TIMEOUT	RF_FSM8_TIMEOUT register
0x4A	0x0128	RF_FSM9_TIMEOUT	RF_FSM9_TIMEOUT register

Table 130. RADIO_REG_BLOCK register list

Address Offset		Register Name	Register Title
RRM UDRA	APB		
0x4B	0x012C	RF_FSM10_TIMEOUT	RF_FSM10_TIMEOUT register
0x4C to 0x50	0x0130 to 0x0140	Reserved	Reserved
0x51	0x0144	SUBG_DIG_CTRL0	SUBG_DIG_CTRL0 register

29.10.2 RF_FSM0_TIMEOUT register (RF_FSM0_TIMEOUT)

Address offset: 0x0000

Reset value: 0x0000 0000

RF_FSM0_TIMEOUT register

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

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

Bits 7:0 **ENA_RFREG_TIMER[7:0]**: Timeout for the RF regulator startup (duration in ENA_RF_REG state)
Step = 16/Fsys=1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=0 µs as LDO_RF is supposed to be already started for high-speed crystal)

29.10.3 RF_FSM1_TIMEOUT register (RF_FSM1_TIMEOUT)

Address offset: 0x0004

Reset value: 0x0000 0006

RF_FSM1_TIMEOUT register

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

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

Bits 7:0 **SYNTH_SETUP_TIMER[7:0]**: Timeout management for the RF regulator to stabilize after RF PLL power on
Step = 16/Fsys=1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=6 µs)

29.10.4 RF_FSM2_TIMEOUT register (RF_FSM2_TIMEOUT)

Address offset: 0x0008

Reset value: 0x0000 0050

RF_FSM2_TIMEOUT register

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

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

Bits 7:0 **VCO_CALIB_LOCK_TIMER[7:0]**: Timeout for the RF PLL calibration + RF PLL lock (duration in CALIB_VCO+LOCKRXTX state)
Step = 16/Fsys=1 µs
Min timeout = 27 µs (this value must not be less than a cailbration session duration)
Max timeout=255 µs
(Default value=80 µs)

29.10.5 RF_FSM3_TIMEOUT register (RF_FSM3_TIMEOUT)

Address offset: 0x000C

Reset value: 0x0000 0028

RF_FSM3_TIMEOUT register

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

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

Bits 7:0 **VCO_LOCK_TIMER[7:0]**: Timeout for the RF PLL lock event when no calibration is requested (duration in LOCKRXTX state)
Step = 16/Fsys=1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=40 µs)

29.10.6 RF_FSM4_TIMEOUT register (RF_FSM4_TIMEOUT)

Address offset: 0x0010

Reset value: 0x0000 000F

RF_FSM4_TIMEOUT register

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

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

Bits 7:0 **EN_RX_TIMER[7:0]**: Timeout for the analog RX chain setup (duration in EN_RX state)
Step = 16/Fsys=1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=15 µs)

29.10.7 RF_FSM5_TIMEOUT register (RF_FSM5_TIMEOUT)

Address offset: 0x0014

Reset value: 0x0000 0019

RF_FSM5_TIMEOUT register

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

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

Bits 7:0 **EN_PA_TIMER[7:0]**: Timeout for the analog PA (DAC) setup (duration in EN_PA state)
Step = 16/Fsys=1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=25 µs)

29.10.8 RF_FSM6_TIMEOUT register (RF_FSM6_TIMEOUT)

Address offset: 0x0018

Reset value: 0x0000 0019

RF_FSM6_TIMEOUT register

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

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

Bits 7:0 **PA_DWN_ANA_TIMER[7:0]**: Timeout for the analog PA (DAC) ramp down (duration in PA_DWN_ANA state)
Step = 16/Fsys=1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=25 µs)

29.10.9 RF_FSM7_TIMEOUT register (RF_FSM7_TIMEOUT)

Address offset: 0x001C

Reset value: 0x0000 0005

RF_FSM7_TIMEOUT register

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

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

Bits 7:0 **EN_LNA_TIMER[7:0]**: Timeout for the analog RX chain signals settlement once PGA precharge is shut down (duration in EN_LNA state)
Step = 16/Fsys=1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=5 µs)

29.10.10 AFC0_CONFIG register (AFC0_CONFIG)

Address offset: 0x0020

Reset value: 0x0000 0025

AFC0_CONFIG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFC_FAST_GAIN_LOG2[3:0]				AFC_SLOW_GAIN_LOG2[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:4	**AFC_FAST_GAIN_LOG2[3:0]**: AFC loop gain in fast mode. (2's log.)
Bits 3:0	**AFC_SLOW_GAIN_LOG2[3:0]**: AFC loop gain in slow mode. (2's log.)

29.10.11 AFC1_CONFIG register (AFC1_CONFIG)

Address offset: 0x0024

Reset value: 0x0000 0018

AFC1_CONFIG register

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

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

Bits 7:0 **AFC_FAST_PERIOD[7:0]**: Length of the AFC fast period (in number of samples unit)

The recommended setting for this bit is such that the fast period equals the preamble length. Since the algorithm operates typically on 2 samples per symbol, the programmed value should be twice the number of preamble symbols.

29.10.12 AFC2_CONFIG register (AFC2_CONFIG)

Address offset: 0x0028

Reset value: 0x0000 00C8

AFC2_CONFIG register

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

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

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

Bit 7 **AFC_FREEZE_ON_SYNC**: Freeze AFC correction upon SYNC word detection

Bit 6 **AFC_EN**: Enable AFC.

Note: this bit can be used for frequency modulation such as 2-(G) FSK and 4-(G)FSK. In ASK/OOK, the AFC algorithm is systematically used by the HW to adjust the amplitude offset before the slicer.

Bit 5 **AFC_MODE**: Select AFC mode

0: slider correction mode (AFC Loop closed on slicer)

1: 2nd IF corection mode (AFC Loop closed on 2nd IF conversion stage)

Bits 4:0 **AFC_PD_LEAKAGE[4:0]**: AFC Peak Detection leakage.

This bit field sets the decay speed of the min/max frequency peak detector

0: no leakage

- ...

- 31: high leakage

Note: the recommended value for this bit field is 4.

29.10.13 AFC3_CONFIG register (AFC3_CONFIG)

Address offset: 0x002C

Reset value: 0x0000 00E8

AFC3_CONFIG register

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

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

Bits 7:6 **AFC_REINIT_OPTION[1:0]**: Select the AFC reinitialization option:
00: No reinitialization allowed
01: Reinitialization based on RSSI ramping
10: Reinitialization based on abnormal Fdev
11: Reinitialization on both RSSI or abnormal Fdev

Bits 5:2 **AFC_TH_SIGN_PERM[3:0]**: Threshold of chech sign permanence mechanism.

Bit 1 **AFC_SIGN_PERM_CHECK**: Enable the check of sign permanence of AFC corrected signal.

Bit 0 **AFC_INIT_MODE**: Control the initialization phase of the AFC and clock recovery algorithms
0: The initial frequency error is estimated based on the average of the frequency deviation of the first 4 received symbols after that the RSSI has passed the threshold level. After the initial estimation, the frequency correction is adjusted in closed loop tracking the frequency error on the received PREAMBLE, SYNC and PAYLOAD symbols.
1: The initial frequency error is estimated simultaneously with the detection of the SYNC word, based on the average frequency deviation of the last 4 PREAMBLE symbols before the SYNC word itself.
When this mode is selected, the clock recovery algorithm is initialized based on the zero crossing points of such PREAMBLE symbols. After the initial estimation, the frequency correction is adjusted in closed loop tracking the frequency error on the packet PAYLOAD symbols.
Note: This option provides better performance in case of packet formats with very short preamble

29.10.14 CLKREC_CTRL0 register (CLKREC_CTRL0)

Address offset: 0x0030

Reset value: 0x0000 00B8

CLKREC_CTRL0 register

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

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

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

Bit 7 **PSTFLT_LEN**: Control the length of the demodulator post-filter
0: 8 symbols,
1: 16 symbols
Note: Setting this value to 1 may improve demodulation performance but requires a slower recovery

Bits 6:4 **CLKREC_P_GAIN_FAST[2:0]**: Clock recovery fast loop gain (log2)
- in DLL mode: control the convergence speed of the loop
- in PLL mode: gain coefficient used before the SYNC word detection

Bits 3:0 **CLKREC_I_GAIN_FAST[3:0]**: Integral fast gain for the clock recovery loop (PLL mode only)
This coefficient is used before the SYNC word detection.

29.10.15 CLKREC_CTRL1 register (CLKREC_CTRL1)

Address offset: 0x0034

Reset value: 0x0000 005C

CLKREC_CTRL1 register

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

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

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

Bit 7 **CLKREC_ALGO_SEL**: Symbol timing recovery algorithm selection
0: simple first order algorithm is used (shortly referred to as DLL)
1: second order algorithm is used (shortly referred to as PLL)

Bits 6:4 **CLKREC_P_GAIN_SLOW[2:0]**: Clock recovery slow loop gain (log2)
- in DLL mode: control the convergence speed of the loop
- in PLL mode: gain coefficient used after the SYNC word detection

Bits 3:0 **CLKREC_I_GAIN_SLOW[3:0]**: Integral slow gain for the clock recovery loop (PLL mode only)
This coefficient is used after the SYNC word detection.

29.10.16 DCREM_CTRL0 register (DCREM_CTRL0)

Address offset: 0x0038

Reset value: 0x0000 00E8

DCREM_CTRL0 register

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

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

Bit 7 **TRACK_GAIN**: Filter gain in track mode for the DC removal block.
This bit field sets the final bandwidth of the filter which is used after the track period.
The recommended value for this parameter is 14.

Bits 6:5 Reserved, must be kept at reset value.

Bits 4:0 **START_GAIN[4:0]**: Filter gain in start mode for the DC removal block.
This bit field sets the final bandwidth of the filter which is used after the start period.
The recommended value for this parameter is 8.

29.10.17 DCREM_CTRL1 register (DCREM_CTRL1)

Address offset: 0x003C

Reset value: 0x0000 0005

DCREM_CTRL1 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.18 IQC_CTRL0 register (IQC_CTRL0)

Address offset: 0x0040

Reset value: 0x0000 00E3

IQC_CTRL0 register

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

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

Bits 7:4 **SLOW_GAIN[3:0]**: Gain of the correction loop in slow mode.
A lower value corresponds to a larger gain.
Note: setting a large gain makes the loop faster but increases the noise.

Bits 3:0 **FAST_GAIN[3:0]**: Gain of the correction loop in fast mode.
A lower value corresponds to a larger gain.
Note: setting a large gain makes the loop faster but increases the noise.

29.10.19 IQC_CTRL1 register (IQC_CTRL1)

Address offset: 0x0044

Reset value: 0x0000 0008

IQC_CTRL1 register

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

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

Bits 7:0 **QPD_ATTACK[7:0]**: Attack coefficient for QPD:
This bit field controls the QPD detector inside the IQC algorithm.
The recommended value is 8.

29.10.20 IQC_CTRL2 register (IQC_CTRL2)

Address offset: 0x0048

Reset value: 0x0000 0008

IQC_CTRL2 register

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

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

Bits 7:0 **QPD_DECAY[7:0]**: Decay coefficient for QPD:
This bit field controls the QPD detector inside the IQC algorithm.
The recommended value is 8.

29.10.21 IQC_CTRL3 register (IQC_CTRL3)

Address offset: 0x004C

Reset value: 0x0000 0007

IQC_CTRL3 register

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

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

Bits 3:0 **FAST_TIME[3:0]**: Duration of the fast mode.

29.10.22 AGC_ANA_ENG register (AGC_ANA_ENG)

Address offset: 0x0050

Reset value: 0x0000 0000

AGC_ANA_ENG register

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

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

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

Bits 7:5 **RFD_RX_PGA_AGCGAIN[2:0]**: Attenuation at PGA level by step of 6 dB with binary code
000: no attenuation
001: -6 dB
010: -12 dB
011: -18 dB
100: -24 dB
101: -30 dB
11x: -30 dB

Bits 4:1 **RFD_RX_ATTEN_AGCGAIN[3:0]**: Attenuation at LNA level by step of 6 dB with thermometric code
0000: no attenuation
0001: -6 dB
0011: -12 dB
0111: -18 dB
1111: -24 dB

Bit 0 **FOREC_AGC_GAINS**: Select the test mode for AGC analog part
0: the AGC signals are managed by the AGC controller in RX top
1: the signals listed above are forced by the associated register bit value

29.10.23 AGC0_CTRL register (AGC0_CTRL)

Address offset: 0x0054

Reset value: 0x0000 0099

AGC0_CTRL register

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

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

Bit 7 **AGC_EN**: Enable the AGC
0: AGC is disabled
1: AGC is enabled (default)

Bit 6 **AGC_START_ONHOLD**: Start the AGC with a hold phase.

Bits 5:0 **AGC_HOLD_TIME[5:0]**: AGC hold time.
Time unit is Fsys (16 MHz) clock cycle.

29.10.24 AGC1_CTRL register (AGC1_CTRL)

Address offset: 0x0058

Reset value: 0x0000 0062

AGC1_CTRL register

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

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

Bits 7:4 **AGC_MAX_THR[3:0]**: Maximum signal threshold.
Attenuation is increased immediately if the signal goes above this level.

Bits 3:0 **AGC_MIN_THR[3:0]**: Minimum signal threshold.
Attenuation is reduced if the signal goes below this level for the while measurement time.

29.10.25 AGC2_CTRL register (AGC2_CTRL)

Address offset: 0x005C

Reset value: 0x0000 00AF

AGC2_CTRL register

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

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

Bit 7 **AGC_HIGH_ATTEN_MODE**: Enable the high attenuation mode.
In high attenuation mode 2 attenuation steps is added if the ADC is fully saturated.

Bit 6 **AGC_FREEZE_ON_STEADY**: Enable the autofreeze feature

Bit 5 **AGC_FREEZE_ON_SYNC**: Enable the freeze on SYNC detection feature

Bit 4 **AGC_START_MAX_ATTEN**: Start the AGC with maximum attenuation

Bits 3:0 **AGC_MEAS_TIME[3:0]**: Measure time.
Time unit is: 16 MHz (Fsys) clock period x 2^(AGC_MEAS_TIME+2).

29.10.26 AGC3_CTRL register (AGC3_CTRL)

Address offset: 0x0060

Reset value: 0x0000 0090

AGC3_CTRL register

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

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

Bits 7:4 **AGC_MAX_ATTEN[3:0]**: Maximum AGC attenuation.

Limits the maximum attenuation step to this value.

Note: max value is 9 (as maximum 9 steps of attenuation are available in RX chain)

Bits 3:0 **AGC_MIN_ATTEN[3:0]**: Minimum AGC attenuation.

Limits the minimum attenuation step to this value.

29.10.27 AGC4_CTRL register (AGC4_CTRL)

Address offset: 0x0064

Reset value: 0x0000 0002

AGC4_CTRL register

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

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

Bits 3:0 **AGC_FREEZE_THR[3:0]**: Signal threshold for the autofreeze feature.

The AGC may be frozen if the signal level remains between the max threshold and the freeze threshold for the whole measure time.

29.10.28 AGC_ATTEN0 register (AGC_ATTEN0)

Address offset: 0x0068

Reset value: 0x0000 0000

AGC_ATTEN0 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.29 AGC_ATTEN1 register (AGC_ATTEN1)

Address offset: 0x006C

Reset value: 0x0000 0010

AGC_ATTEN1 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.30 AGC_ATTEN2 register (AGC_ATTEN2)

Address offset: 0x0070

Reset value: 0x0000 0020

AGC_ATTEN2 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.31 AGC_ATTEN3 register (AGC_ATTEN3)

Address offset: 0x0074

Reset value: 0x0000 0030

AGC_ATTEN3 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.32 AGC_ATTEN4 register (AGC_ATTEN4)

Address offset: 0x0078

Reset value: 0x0000 0040

AGC_ATTEN4 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.33 AGC_ATTEN5 register (AGC_ATTEN5)

Address offset: 0x007C

Reset value: 0x0000 0041

AGC_ATTEN5 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.34 AGC_ATTEN6 register (AGC_ATTEN6)

Address offset: 0x0080

Reset value: 0x0000 0043

AGC_ATTEN6 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.35 AGC_ATTEN7 register (AGC_ATTEN7)

Address offset: 0x0084

Reset value: 0x0000 0047

AGC_ATTEN7 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.36 AGC_ATTEN8 register (AGC_ATTEN8)

Address offset: 0x0088

Reset value: 0x0000 004F

AGC_ATTEN8 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.37 AGC_ATTEN9 register (AGC_ATTEN9)

Address offset: 0x008C

Reset value: 0x0000 005F

AGC_ATTEN9 register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.38 AGC1_ATTEN_TRIM register (AGC1_ATTEN_TRIM)

Address offset: 0x0090

Reset value: 0x0000 00AF

AGC1_ATTEN_TRIM register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.39 AGC2_ATTEN_TRIM register (AGC2_ATTEN_TRIM)

Address offset: 0x0094

Reset value: 0x0000 00AF

AGC2_ATTEN_TRIM register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.40 AGC3_ATTEN_TRIM register (AGC3_ATTEN_TRIM)

Address offset: 0x0098

Reset value: 0x0000 00AF

AGC3_ATTEN_TRIM register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.41 AGC4_ATTEN_TRIM register (AGC4_ATTEN_TRIM)

Address offset: 0x009C

Reset value: 0x0000 00AF

AGC4_ATTEN_TRIM register

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

Bits 31:0 Reserved, must be kept at reset value.

29.10.42 AGC_PGA_HWTRIM_OUT register (AGC_PGA_HWTRIM_OUT)

Address offset: 0x00A0

Reset value: 0x0000 0008

AGC_PGA_HWTRIM_OUT register

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

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

Bits 3:0 **AGC_HW_PGA_TRIM[3:0]**: AGC PGA calibration information loaded by HW from the SoC flash.

Default value if flash is empty is 0x8.

29.10.43 PA_REG register (PA_REG)

Address offset: 0x00A8

Reset value: 0x0000 0000

PA_REG register

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

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

Bit 3 **PA_DEGEN_ON**: Enable a 'degeneration' mode, which introduces a pre-distortion to linearize the power control curve.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **CFG_FILT[1:0]**: FIR configuration

00: filtering
01: ramping
10: switching
11: filtering

29.10.44 PA_HWTRIM_OUT register (PA_HWTRIM_OUT)

Address offset: 0x00AC

Reset value: 0x0000 0088

PA_HWTRIM_OUT register

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

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

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

Bits 7:4 **PA_HW_DEGEN_TRIM[3:0]**: MSB part meaning:
00xx: Code threshold: 418
01xx: Code threshold: 439
10xx: Code threshold: 465 (default when no trimming in flash)
11xx: Code threshold: 485
LSB part meaning
xx00 => Clamp voltage: 0.40 V (default when no trimming in flash)
xx01 => Clamp voltage: 0.45 V
xx10 => Clamp voltage: 0.50 V
xx11 => Clamp voltage: 0.55 V

Bits 3:0 Reserved, must be kept at reset value.

29.10.45 RSSI_FLT register (RSSI_FLT)

Address offset: 0x00BC

Reset value: 0x0000 00E0

RSSI_FLT register

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

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

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

Bits 7:4 **RSSI_FLT[3:0]**: Gain of the RSSI filter

Bit 3 **FREEZE_SYNC_ON_SYNC_OOK_PEAK_DECAY**: freeze on sync OK peak decay.

Bits 2:0 **OOK_PEAK_DECAY[3:0]**: Peak decay control for OOK:

Bits 2:0

000b: ook peak decay disabled
001b: min fast decay
011b: slow decay
111b: max slow decay
Others: reserved.

29.10.46 SYNTH2_ANA_ENG register (SYNTH2_ANA_ENG)

Address offset: 0x00C8

Reset value: 0x0000 004C

SYNTH2_ANA_ENG register

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

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

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

Bit 3 **RFD_PLL_LD_WIN_ACC**: Select the PLL lock detector window selection
1 (default): The PLL lock detector uses a smaller (divided by 2) window
0: The PLL lock detector uses the largest window (cut 1 configuration)

Bits 2:0 **RFD_PLL_VCO_ALC_AMP[2:0]**: Select the level of max VCO amplitude in amplitude level control loop.
Default value is "100".

29.10.47 RXADC_HWDELAYTRIM_OUT register (RXADC_HWDELAYTRIM_OUT)

Address offset: 0x00E8

Reset value: 0x0000 001B

RXADC_HWDELAYTRIM_OUT register

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

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

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:3 **RXADC_HW_DELAYTRIM_Q[2:0]**: Control bits of the RX ADC loop delay for Q channel (from SoC flash memory).

Default value mentioned here is when empty flash.

Bits 2:0 **RXADC_HW_DELAYTRIM_I[2:0]**: Control bits of the RX ADC loop delay for I channel (from SoC flash memory).

Default value mentioned here is when empty flash.

29.10.48 RX_AAF_HWTRIM_OUT register (RX_AAF_HWTRIM_OUT)

Address offset: 0x00F4

Reset value: 0x0000 0006

RX_AAF_HWTRIM_OUT register

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

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

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

Bits 3:0 **AAF_HW_FCTRIM[3:0]**: AAF calibration information loaded by HW.

Default value mentioned here is when empty flash.

29.10.49 SINGEN_ANA_ENG register (SINGEN_ANA_ENG)

Address offset: 0x0100

Reset value: 0x0000 0000

SINGEN_ANA_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_SINGEN_LBE	RFD_SINGEN_DIV2_PUP	RFD_SINGEN_ENA
													rw	rw	rw

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

Bit 2 **RFD_SINGEN_LBE**: This bit value is directly connected to the RFSUBG analog IP pin.

This bit must be set to:

1: when the SINGEN is used in the 400-500 MHz band

0: else

Bit 1 **RFD_SINGEN_DIV2_PUP**: This bit value is directly connected to the RFSUBG analog IP pin.
This bit must be set to:

1: when the SINGEN is used in the 400-500 MHz band

0: else

For devices with the 159 -185 MHz frequency band option **only -** This bit value is directly connected to the RFSUBG analog IP pin.

This bit must be set to:

1: when the SINGEN is used in the 169 MHz band

0: else

Bit 0 **RFD_SINGEN_ENA**: Enable SINGEN signal for the RFSUBGanalog IP.

This bit value is directly connected to the RFSUBG analog IP pin.

29.10.50 RF_INFO_OUT register (RF_INFO_OUT)

Address offset: 0x0108

Reset value: 0x0000 0040 (0x0000 0043 for 169 MHz band part number)

RF_INFO_OUT register

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

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

Bits 7:4 **RFSUBG_ID[3:0]**: Indicates the version of the analog RFSUBG IP embedded in the device
0000: cut 1.0
0001: cut 1.1
0100: cut 2.0

Bits 3:0 **FQCY_BAND_ID[3:0]**: Indicates the version of the RFSUBG IP embedded in the device
0000: standard band
0011: 169 MHz band

29.10.51 RF_FSM8_TIMEOUT register (RF_FSM8_TIMEOUT)

Address offset: 0x0124

Reset value: 0x0000 000A

RF_FSM8_TIMEOUT register

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

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

Bits 7:0 **SYNTH_PDWN_TIMER[7:0]**: Timeout management for the RF regulator to stabilize after PLL shut down

Step = 16/CLK16M = 1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=10 µs)

29.10.52 RF_FSM9_TIMEOUT register (RF_FSM9_TIMEOUT)

Address offset: 0x0128

Reset value: 0x0000 0006

RF_FSM9_TIMEOUT register

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

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

Bits 7:0 **END_RX_TIMER[7:0]**: Timeout management for the RF regulator to stabilize after analog RX chain shut down

Step = 16/CLK16M=1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=6 µs)

29.10.53 RF_FSM10_TIMEOUT register (RF_FSM10_TIMEOUT)

Address offset: 0x012C

Reset value: 0x0000 0006

RF_FSM10_TIMEOUT register

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

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

Bits 7:0 **END_TX_TIMER[7:0]**: Timeout management for the RF regulator to stabilize after clock stops on the analog PA block
Step = 16/CLK16M=1 µs
Min timeout = 1 µs
Max timeout=255 µs
(Default value=6 µs)

29.10.54 SUBG_DIG_CTRL0 register (SUBG_DIG_CTRL0)

Address offset: 0x0144

Reset value: 0x0000 0000

SUBG_DIG_CTRL0 register

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

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

Bit 0 **FORCE_GPIO_OUTPUT**: Option for the direct GPIO signal output
0 (default): The GPIO signals (RX Clock / RX Data for reception and TX Clock for transmission) are propagated to the SoC only when direct through GPIO mode is selected.
1: The GPIO signals (RX Clock / RX Data for reception and TX Clock for transmission) are propagated to the SoC regardless the RX or TX mode.

29.10.55 RX_CHAIN_ENG register (RX_CHAIN_ENG)

Address offset: 0x0148

Reset value: 0x0000 0003

RX_CHAIN_ENG register

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

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

Bit 1 **PGA_PRECH_ENA**: Select the activation or not of the PGA precharge during the EN_RX state of the Radio FSM

0: PGA precharge is disabled

1 (default): PGA precharge is active/enabled

Bit 0 **LNA_ISOL_ENA**: Select the isolation or not of the LNA during the EN_RX state of the Radio FSM

0: LNA is enabled

1 (default): LNA is isolated (floating input)

29.10.56 DEMOD_DIG_ENG register (DEMOD_DIG_ENG)

Address offset: 0x014C

Reset value: 0x0000 0003

DEMOD_DIG_ENG register

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

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

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

Bit 0 **RX_BLANKING_LENGTH**: Number of data samples at RX start for which the signal at the output of the channel filter is kept forced to zero

0: no blanking at RX start, Channel filter output considered immeditely

1: 16 (1x16) channel filter samples are forced to zero

- 2: 32 (2x16) channel filter samples are forced to zero

- ...

- 7: 112 (7x16) channel filter samples are forced to zero

Unit of the bit field is 16 channel filter samples.

Note: during this RX blanking window, the demodulator is not able to treat any information present on the antenna.

29.10.57 RADIO_REG_REG_BLOCK register map

Table 131. RADIO_REG_REG_BLOCK register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RF_FSM0_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENA_RFREG_TIMER[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x004	RF_FSM1_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNTH_SETUP_TIMER[7:0]							
	Reset value																									0	0	0	0	0	1	1	0
0x008	RF_FSM2_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALIB_LOCK_TIMER[7:0]							
	Reset value																									0	1	0	1	0	0	0	0
0x00C	RF_FSM3_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_LOCK_TIMER[7:0]							
	Reset value																									0	0	1	0	1	0	0	0
0x010	RF_FSM4_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_RX_TIMER[7:0]							
	Reset value																									0	0	0	0	1	1	1	1
0x014	RF_FSM5_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_PA_TIMER[7:0]							
	Reset value																									0	0	0	1	1	0	0	1

Table 131. RADIO_REG_REG_BLOCK register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	RF_FSM6_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PA_DWN_ANA_TIMER[7:0]							
	Reset value																									0	0	0	1	1	0	0	1
0x01C	RF_FSM7_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_LNA_TIMER[7:0]							
	Reset value																									0	0	0	0	0	1	0	1
0x020	AFC0_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x024	AFC1_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFC_FAST_PERIOD[7:0]							
	Reset value																									0	0	0	1	1	0	0	0
0x028	AFC2_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFC_FREEZE_ON_SYNC	AFC_EN	AFC_MODE	AFC_PD_LEAKAGE[4:0]				
	Reset value																									1	1	0	0	1	0	0	0
0x02C	AFC3_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFC_REINIT_OPTION[1:0]		AFC_TH_SIGN_PERM[3:0]				AFC_SIGN_PERM_CHECK	AFC_INIT_MODE
	Reset value																									1	1	1	0	1	0	0	0

Table 131. RADIO_REG_REG_BLOCK register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	CLKREC_CTRL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSTFLT_LEN	CLKREC_P_GAIN_FAST[2:0]			CLKREC_I_GAIN_FAST[3:0]			
	Reset value																									1	0	1	1	1	0	0	0
0x034	CLKREC_CTRL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLKREC_ALGO_SEL	CLKREC_P_GAIN_SLOW[2:0]			CLKREC_I_GAIN_SLOW[3:0]			
	Reset value																									0	1	0	1	1	1	0	0
0x038	DCREM_CTRL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACK_GAIN	Res.	Res.	START_GAIN[4:0]				
	Reset value																									1			0	1	0	0	0
0x03C	DCREM_CTRL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x040	IQC_CTRL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_GAIN[3:0]				FAST_GAIN[3:0]			
	Reset value																									1	1	1	0	0	0	1	1
0x044	IQC_CTRL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QPD_ATTACK[7:0]							
	Reset value																									0	0	0	0	1	0	0	0
0x048	IQC_CTRL2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QPD_DECAY[7:0]							
	Reset value																									0	0	0	0	1	0	0	0

Table 131. RADIO_REG_REG_BLOCK register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04C	IQC_CTRL3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FAST_TIME[3:0]			
	Reset value																													0	1	1	1
0x050	AGC_ANA_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_RX_PGA_AGCGAIN[2:0]			RFD_RX_ATTEN_AGCGAIN[3:0]				FORCE_AGC_GAINS
	Reset value																									0	0	0	0	0	0	0	0
0x054	AGC0_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_EN	AGC_START_ONHOLD	AGC_HOLD_TIME[5:0]					
	Reset value																									1	0	0	1	1	0	0	1
0x058	AGC1_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_MAX_THR[3:0]				AGC_MIN_THR[3:0]			
	Reset value																									0	1	1	0	0	0	1	0
0x05C	AGC2_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_HIGH_ATTEN_MODE	AGC_FREEZE_ON_STEADY	AGC_FREEZE_ON_SYNC	AGC_START_MAX_ATTEN	AGC_MEAS_TIME[3:0]			
	Reset value																									1	0	1	0	1	1	1	1

Table 131. RADIO_REG_REG_BLOCK register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x060	AGC3_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_MAX_ATTEN[3:0]				AGC_MIN_ATTEN[3:0]			
	Reset value																									1	0	0	1	0	0	0	0
0x064	AGC4_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_FREEZE_THR[3:0]			
	Reset value																													0	0	1	0
0x068	AGC_ATTEN0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x06C	AGC_ATTEN1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x070	AGC_ATTEN2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x074	AGC_ATTEN3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x078	AGC_ATTEN4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x07C	AGC_ATTEN5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x080	AGC_ATTEN6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x084	AGC_ATTEN7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x088	AGC_ATTEN8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x08C	AGC_ATTEN9	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																

Table 131. RADIO_REG_REG_BLOCK register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x090	AGC1_ATTEN_TRIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x094	AGC2_ATTEN_TRIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x098	AGC3_ATTEN_TRIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x09C	AGC4_ATTEN_TRIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0A0	AGC_PGA_HWTRIM_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_HW_PGA_TRIM[3:0]			
	Reset value																													1	0	0	0
0x0A8	PA_REG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PA_DEGEN_ON	Res.	CFG_FILT[1:0]	
	Reset value																													0		0	0
0x0AC	PA_HWTRIM_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PA_HW_DEGEN_TRIM[3:0]				Res.			
	Reset value																													1	0	0	0				

Table 131. RADIO_REG_REG_BLOCK register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0BC	RSSI_FLT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FREEZE_SYNC_ON_SYNC_OOK_PEAK_DECAY	RSSI_FLT[3:0]				OOK_PEAK_DECAY[3:0]			
	Reset value																								0	1	1	1	0	0	0	0	0
0x0C8	SYNTH2_ANA_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL_CP_ISEL_TX[2:0]			RFD_PLL_LD_WIN_ACC	RFD_PLL_VCO_ALC_AMP[2:0]		
	Reset value																										1	0	0	1	1	0	0
0x0CC	SYNTH0_ANA_TST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SD_RESET_N	RFD_PLL_LOLB_SEL	RFD_PLL_LOTX_PUP	RFD_PLL_LORX_PUP	RFD_PLL_DIV2_PUP	RFD_PLL_BUF_PIP_E	RFD_PLL_CP_PUP	SYNTH_ANA_TST_SE
	Reset value																									0	0	0	0	0	0	0	0
0x0D0	SYNTH1_ANA_TST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_PLL_VCO_VREF_PUP	RFD_PLL_VCO_CORE_PUP	RFD_PLL_BUF_OUTLOOP_E	RFD_PLL_ISOBUF_PUP	RFD_PLL_PIPELINE_PUP	RFD_PLL_PFD_PUP	RFD_PLL_PFD_LD_EN
	Reset value																										0	0	0	0	0	0	0

Table 131. RADIO_REG_REG_BLOCK register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D4	SYNTHCAL_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL_RFCOUNTER_OFFSET[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0D8	SYNTHCAL0_ANA_TST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_PLL_LF_CALVCO_ON	RFD_PLL_RFCOUNTER_XRST	RFD_PLL_RFCOUNTER_WINDOW	RFD_PLL_RFCOUNTER_PUP	RFD_PLL_PFD_PD_CALON	RFD_PLL_VCO_ALC_PUP	SYNTHCAL_ANA_TST_SEL
	Reset value																										0	0	0	0	0	0	0
0x0DC	SD_CTRL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SD_INV_CLK	SD_ORDER[1:0]		SD_INT_DIV_EN	FORCE_SD_HOLD
	Reset value																												0	0	0	0	0
0x0E0	RXADC_ANA_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_RXADC_CAL_ENA_Q	RFD_RXADC_CAL_ENA_I
	Reset value																															0	0

Table 131. RADIO_REG_REG_BLOCK register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0E4	RXADC_ANA_TST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_RXADC_CLK_ENA	RFD_RXADC_EN_Q	RFD_RXADC_EN_I	RXADC_ANA_TST_SEL
	Reset value																													0	0	0	0
0x0E8	RXADC_HWDELAYTRIM_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXADC_HW_DELAYTRIM_Q[2:0]			RXADC_HW_DELAYTRIM_I[2:0]		
	Reset value																											0	1	1	0	1	1
0x0F4	RX_AAF_HWTRIM_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AAF_HW_FCTRIM[3:0]			
	Reset value																													0	1	1	0
0x100	SINGEN_ANA_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_SINGEN_LBE	RFD_SINGEN_DIV2_PUP	RFD_SINGEN_ENA
	Reset value																														0	0	0
0x108	RF_INFO_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFSUBG_ID[3:0]				Res.			
	Reset value																									0	1	0	0				

Table 131. RADIO_REG_REG_BLOCK register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x124	RF_FSM8_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNTH_PDWN_TIMER[7:0]							
	Reset value																									0	0	0	0	1	0	1	0
0x128	RF_FSM9_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END_RX_TIMER[7:0]							
	Reset value																									0	0	0	0	0	1	1	0
0x12C	RF_FSM10_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END_TX_TIMER[7:0]							
	Reset value																									0	0	0	0	0	1	1	0
0x144	SUBG_DIG_CTRL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCE_GPIO_OUTPUT
	Reset value																																0

29.10.58 Notes for 169 MHz configuration

Setting the desired IF (300 kHz) in RX, is delegated to the software instead of the HW RFIP:

- When transmitting, the dynamic registers configuring the base frequency SYNTH_FREQ.SYNTH_INT and SYNTH_FREQ.SYNTH_FRAC, are programmed to the desired frequency (that is, 169 MHz)
- When the user wants to switch to an RX operation, this dynamic register must be configured to the desired frequency minus the desired IF (that is,168.7 MHz)

In addition, the register allowing control of the IF by the RFIP must be adapted to deactivate the hardware switch of the IF between RX and TX, by setting IF_CTRL.IF_OFFSET_ANA to 0 and keeping IF_CTRL.IF_OFFSET_DIG at the desired IF (0x04CD = 300 kHz). This configuration is static and can be used for both RX and TX operations.

Note: This does not impact CHAN_NUM, CHAN_SPACING operations because shifting the base frequency is shifting in the same way all the channels carrier frequencies.

This does not limit the sequencer operations as this register is part of the dynamic registers.

An additional configuration, only required for TX operations, involves the dynamic registers MOD1_CONFIG.FDEV_E and MOD1_CONFIG.FDEV_M, whose values must be multiplied by 5.

The register contents must be reverted to their original values when switching to RX mode.

29.11 Receive block

The Recieve block gets the I/Q data provided by the ADC analog RFSUBG IP and provides the final data to the Data Buffer (for RAM) or to the GPIO (for direct through GPIO mode).

The Recieve block also takes in charge some RX indicator measurements like RSSI.

The Recieve block also offers some reception improvement solution like:

- an AGC controller,
- an I/Q compensation (IQC) block,
- a DC removal block,
- an Automatic Frequency Compensation (AFC) block,
- an antenna switching feature.

29.11.1 Automatic gain control (AGC)

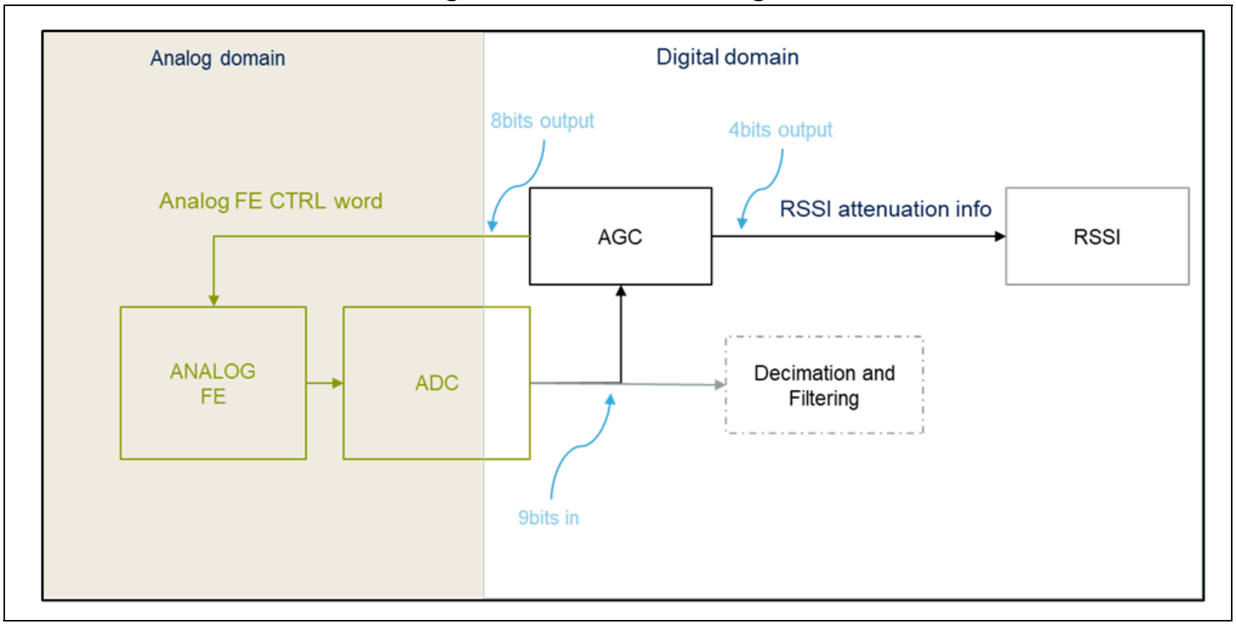
The AGC feature is based on:

- the possibility to attenuate the gain introduced by the RX chain of the RFSUBG analog IP at 2 different locations inside the RX chain,
- the presence of an AGC controller in the RX_Top block used to monitor and apply/release attenuation when needed.

AGC presentation

The algorithm of the AGC controller is designed to keep the signal amplitude at the input of the IF ADC within a specified range. The attenuation is applied by step of 6 dB (or 12 dB) up to a maximum of 54 dB, starting at a received signal power of about -58 dBm (with default radio register settings).

Figure 285. AGC block diagram

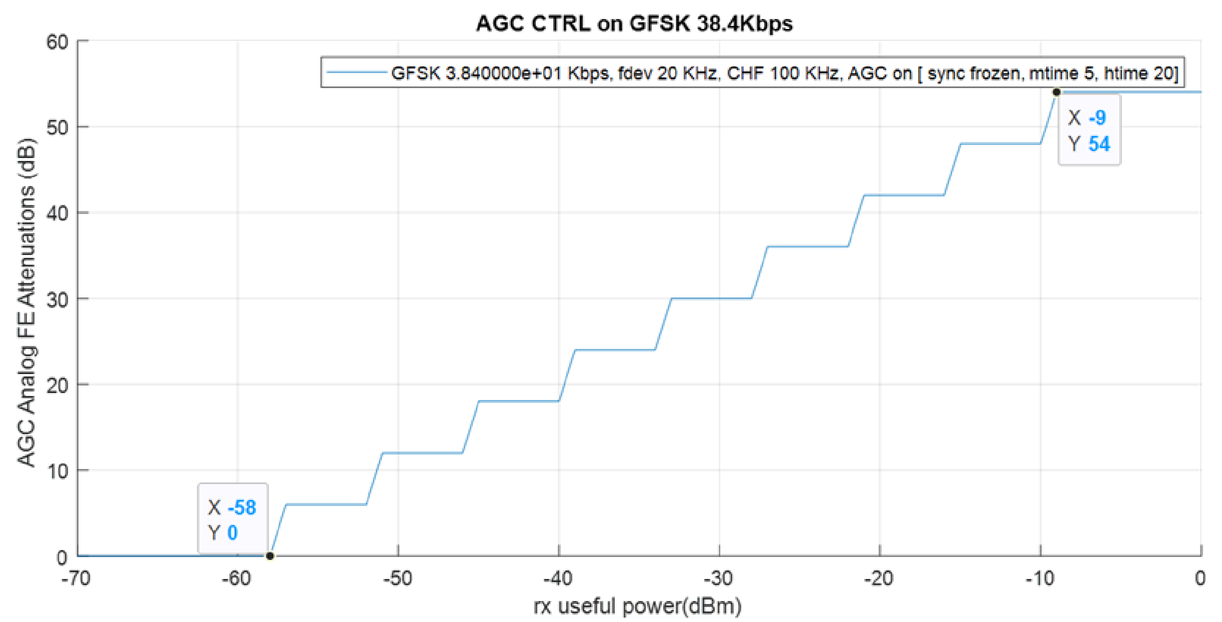


From an implementation point of view, the (peak) signal amplitude is measured, using the Robertson approximation, in the digital domain after the ADC and compared to a low threshold and a high threshold. The bits used for the comparison are the 7 MSB of the measured level:

- if the amplitude is above the high threshold, the attenuation is increased sequentially until the amplitude goes below it,
- if the amplitude is below the low threshold, the attenuation is decreased sequentially until the amplitude goes above it.

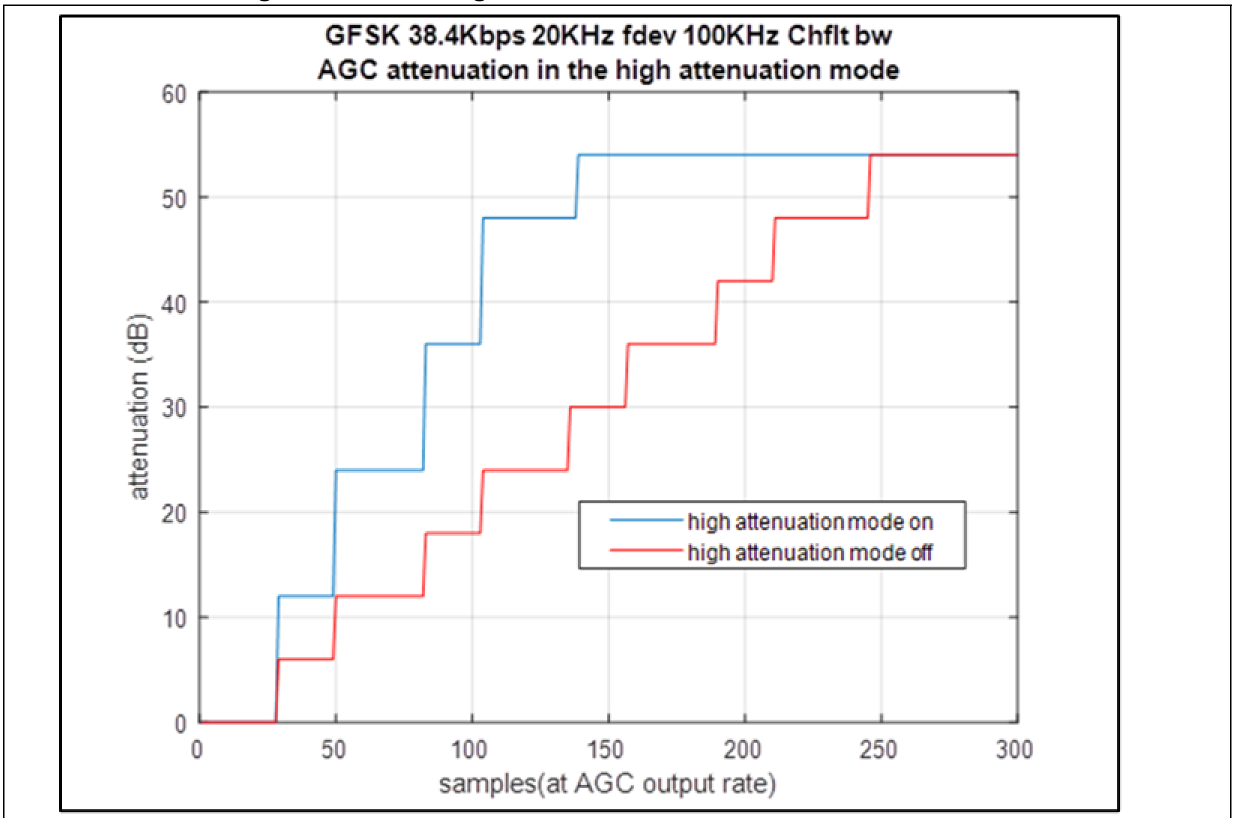
Figure 286 shows the AGC attenuation versus the RX useful power (in dBm) for a GFSK modulation case.

Figure 286. AGC attenuation example



A high attenuation mode is available: when enabled, the steps of attenuation are doubled when the high threshold is exceeded. This high attenuation mode is faster than the normal mode (signal step) in case of short preamble as shown in the next figure (1 byte case and 0 dBm of RX useful power).

Figure 287. AGC high attenuation mode versus normal mode



AGC configuration

The AGC controller can be configured through different bit fields located in the Radio registers. The main bit fields are listed below but the complete parameters can be directly checked through the all the AGCx Radio registers.

- AGC enabling:
  - AGC_EN bit in the *AGC0_CTRL register (AGC0_CTRL)*
  - enabled by default
- High attenuation mode enabling:
  - AGC_HIGH_ATTEN_MODE bit in the *AGC2_CTRL register (AGC2_CTRL)*
  - enabled by default
- Definition of the low and high thresholds:
  - AGC_MIN_THR[3:0} and AGC_MAX_THR[3:0] bit fields in the *AGC1_CTRL register (AGC1_CTRL)*
- Selection of the attenuation at start (min or max):
  - AGC_START_MAX_ATTEN bit in the *AGC2_CTRL register (AGC2_CTRL)*
  - min attenuation selected at start by default
- Possibility to freeze the attenuation on SYNC word detection event:
  - AGC_FREEZE_ON_SYNC bit in the *AGC2_CTRL register (AGC2_CTRL)*
  - enabled by default
- Possibility to freeze the attenuation when the signal goes below the freeze programmable threshold:
  - AGC_FREEZE_ON_STEADY bit in the *AGC2_CTRL register (AGC2_CTRL)*
  - disabled by default
- Set the measurement interval during which the signal is determined:
  - AGC_MEAS_TIME[3:0] bit field in the *AGC2_CTRL register (AGC2_CTRL)*
  - the actual time is

$$T_{AGC\ meas} = 2^{(AGC\_MEAS\_TIME + 2)} / f_{SYS}$$

    ranging from 250 ns to 8.192 ms,
  - for (G-)FSK modulations, the measurement time is normally set to a few µs in order to achieve a fast settling of the algorithm
  - for ASK/OOK modulations, the measurement time must be larger than the duration of the expected longest train of ‘0’ symbols to avoid unstable behavior
- Define the interval between two peak measurements (hold time):
  - AGC_HOLD_TIME[5:0] bit field in the *AGC0_CTRL register (AGC0_CTRL)*
  - sets the interval between two peaks measurements in number of signal samples,
  - the actual time is

$$T_{AGC\ hold} = AGC\_HOLD\_TIME / f_{SYS}$$

  - this time lag is necessary to consider the time propagate the attenuation setting through the RF path and see the impact on the AD data.

The AGCx_ATTEN Radio registers (with x=0 to 9) contains the settings for each step from step0 (= no attenuation) to step 9 (= max attenuation).

The analog RF chain contains two points along the chain to attenuate:

- attenuation on PGA stage:
  - 5 possible steps at this point,
  - controlled by the PGA_AGCGAIN_x[2:0] bit field in the AGC_ATTENx radio registers,
  - coded in binary format (1 represents 6 dB of attenuation, 5 represents the maximum attenuation, that is, 30 dB)
- attenuation on LNA+BOM stage:
  - 4 possible steps at this point,
  - controlled by the ATTEN_AGCGAIN_x[3:0] bit field in the AGC_ATTENx radio registers,
  - coded in thermometric format (“0001” represents 6 dB of attenuation, “1111” represents the maximum attenuation, that is, 24 dB)

In addition, the number of steps applied on the RX analog chain can be read in the AGC_WORD[3:0] bit field of the *RX_INDICATOR register (RX_INDICATOR)* (status register).

29.11.2 DC removal block

The DC block function removes the DC component that may arise in the analog signal path. The bandwidth of the DC block filter can be programmed according to the following table.

Table 132. DC attenuation per sample versus the filter order n (for F_{SYS}=16 MHz)

n (start or track gain)	Bandwidth (in kHz)	DC attenuation per sample
0	541	-Inf
1	498	-24.08
2	455	-18.06
3	414	-14.54
4	372	-12.04
5	332	-10.10
6	294	-8.52
7	257	-7.18
8	221	-6.02
9	187	-5.00
10	156	-4.08
11	126	-3.25
12	97	-2.50
13	71	-1.80
14	45	-1.16
15	22	-0.56

Note: If the system clock is not 16 MHz, the actual bandwidth of the DC filter can be obtained by multiplying the values in the above table by the factor f_{CLK} / 16e6.

In order to achieve the both a fast settling time and a small signal degradation, the band of the filter is set to a relatively large value for a short time and then narrowed.

The DC filter is controlled by the following parameters in the Radio registers:

- Start gain:
  - START_GAIN[3:0] bit field in the DCREM_CTRL0 register (DCREM_CTRL0) register,
  - sets the initial bandwidth of the filter, which is used during the start period,
  - default value is the recommended value (8).
- Track gain:
  - TRACK_GAIN[3:0] bit field in the *DCREM_CTRL0 register (DCREM_CTRL0)*
  - sets the final bandwidth of the filter, which is used after the start period,
  - default value is the recommended value (14 = 0xE).

29.11.3 I/Q mismatch compensation

The MR_SubG IP embeds an adaptive digital algorithm that could compensate I/Q mismatch on the analog signal path. Since the I/Q mismatch is expected to be quite constant on the short term, the calculated compensation value can be reused for a next reception (to speed up the correction).

I/Q mismatch compensation presentation

The IQC correction can be used with different configurations:

- no compensation calculated nor applied during the reception,
- I/Q mismatch compensation calculated during a reception with initialization value for the algorithm:
  - equal to 0
  - equal to last compensation result,
  - equal to a value provided by the SW
- I/Q mismatch compensation applied using a static value (no dynamic adjustment) coming from a recirculation register:
  - can be the last compensation result
  - can be a value provided by the SW.

As the RX_Top (and so the IQC block) are put under reset between each reception, there is a need to record the correction result at the end of a RX to be able to reuse it as initialization value for the next calculation. The Radio controller takes in charge this role by implementing a recirculation register allowing to store either the last calculation result or an external value provided by the SW through the LOAD_IQC_INIT action bit and the IQC_CORRECT_IN[23:0] bit field in the IQC_CONFIG static register.

Note: The Radio Controller stores the IQC block results only at the end of RX including if the ending is generated by a SABORT command (to cover the direct through GPIO/Buffers modes use-cases).

The recirculation register content:

- is visible in the *IQC_CORRECTION_OUT register (IQC_CORRECTION_OUT)* (status register)
- is applied to the IQC block correction input (to be used by the algorithm if REUSE_CORRECTION bit is set)
- is loaded with the IQC block output result on a RX ending **if IQC_ENABLE bit is set**,
- is loaded with IQC_CORRECT_IN[23:0] bit field on write 1 action in the IQC_LOAD_INIT bit.

Note: The IQC_CORRECT_OUT register is updated only:
- at the end of a RX (normal RX end in normal mode or SABORT for other RX modes)
- on a LOAD_IQC_INIT command (then updated with the IQC_CORRECT_IN[23:0] value)

The IQ compensation can be used with different configuration (managed through the *IQC_CORRECTION_OUT register (IQC_CORRECTION_OUT)* (status register), summarized in *Table 133*.

Table 133. IQC correction available configurations

IQC_ENABLE	REUSE_CORRECTION	Associated behavior
0	0	The I/Q correction is effectively disabled: no compensation is applied.
1	0	The I/Q adaptive correction is initialized with 0 for each packet.
1	1	The I/Q adaptive correction is initialized with the value present in the recirculation register (result of last received packet or IQC_CORRECT_IN[23:0] value loaded through IQC_INIT_LOAD action bit).
0	1	The I/Q correction is performed using the value provided by the recirculation register and is NOT adaptively adjusted.

IQC block algorithm configuration

The IQC block algorithm is controlled by the following parameters:

- Adaptation enabling:
  - IQC_ENABLE bit in the IQC_CONFIG static register
  - enables the adaptive I/Q compensation algorithm
- Start the coefficient adaptation from a specified value (previous calibration result / SW loaded value):
  - REUSE_CORRECTION bit in the IQC_CONFIG static register,
  - when set, the algorithm coefficient starts on the value provided by the recirculation register
- Gain configuration for fast period
  - FAST_GAIN[3:0] bit field in the *IQC_CTRL0 register (IQC_CTRL0)* (radio register)
  - sets the gain of the correction loop in fast mode
  - a lower value corresponds to a larger gain
  - setting a larger gain makes the loop faster but increases the noise
- Gain configuration for slow period
  - SLOW_GAIN[3:0] bit field in the *IQC_CTRL0 register (IQC_CTRL0)* (radio register),
  - sets the gain of the correction loop in slow mode
  - a lower value corresponds to a larger gain
  - setting a larger gain makes the loop faster but increases the noise
- Fast period duration:
  - FAST_TIME[3:0] bit field in the *IQC_CTRL3 register (IQC_CTRL3)* (radio register)
  - duration of the fast mode
  - the actual time is

$$\text{Fast Period} = \frac{4}{f_{SYS}} \times 2^{FAST\_TIME}$$

- QPD attack parameter:
  - QPD_ATTACK[7:0] bit field in the *IQC_CTRL1 register (IQC_CTRL1)* (radio register)
  - attack coefficient for the QPD detector
  - default value is the recommended one (8)
- QPD decay parameter:
  - QPD_DECAY[7:0] bit field in the *IQC_CTRL2 register (IQC_CTRL2)* (radio register)
  - decay coefficient for the QPD detector,
  - default value is the recommended one (8)

29.11.4 RSSI

The Received Signal Strength Indicator is a measure of the received signal power at the antenna measured in the channel filter bandwidth.

The measured power is reported in step of 0.5 dB from 0 to 511 and is offset in such a way that 0 means -120 dBm corresponds to about 82.

The result of the measured is available in two separate bit fields located in the *RX_INDICATOR register (RX_INDICATOR)* (status register):

- RSSI_LEVEL_RUN[8:0]: continuous level of the measured RSSI
- RSSI_LEVEL_ON_SYNC[8:0]: measured RSSI captured on SYNC word detection event

The conversion from RSSI_LEVEL_xx value to dBm is given below.

Equation 7 – RSSI dBm conversion formula

$$RSSI_{in\_dBm} = \frac{RSSI\_LEVEL\_xx}{2} - 96 - Gain_{RX\_Chain}$$

with $Gain_{RX\_Chain}$ typically equal to 65

The RSSI measurement takes into account the AGC steps currently applied and integrate the information in the provided result (by adding 6 dB by step). This means the RSSI read in the register already computed this AGC step compensation and can be considered directly as the power on the antenna.

The RSSI estimation is controlled by the following parameter in the Radio registers:

- Filter coefficient:
  - RSSI_FLT[3:0] bit field in the *RSSI_FLT register (RSSI_FLT)*
  - sets the bandwidth of a low IIR pass filter
  - a lower value gives a faster settling of the measurement but a lower precision,
  - the default value is the recommended one (14 = 0xE)

29.11.5 Carrier sense

The carrier sense functionality can be used to detect if any signal is being received. The detection is based on the measured RSSI value.

Two operational mode can be selected for the carrier sensing, controlled through the CS_MODE[1:0] bit field in the *AS_QI_CTRL register (AS_QI_CTRL)* (static register):

- static carrier sensing
  - selected when CS_MODE[1:0] = “00”,
  - the carrier sense signal:

    is asserted when the measured RSSI goes above a programmed threshold specified in the RSSI_THR[8:0] bit field of the AS_QI_CTRL register

    is de-asserted when the measured RSSI goes 3 dB below the same programmed threshold
- dynamic carrier sensing:
  - the carrier sense signal:

    is asserted if the signal is above the programmed threshold and a fast power increase is detected.

    is de-asserted if a power fall of the same amplitude is detected so a fall of:
  - The fast increase and fall amplitude can be:

    6 dB when CS_MODE[1:0] = “01”

    12 dB when CS_MODE[1:0] = “10”

    18 dB when CS_MODE[1:0] = “11”

The rising edge of the carrier sense signal set the CS_F bit in the RFSEQ_IRQ_STATUS register and an associated interrupt is available when enabled.

The carrier sense information is also the criteria used by the Fast RX termination mode (refer *Section 29.14.4: Fast RX termination* section to for details) and one of the stop condition for the RX Timer (refer to *Section 29.14.3: RX Timer* section for details).

29.11.6 Antenna switching

The antenna switching feature allows controlling an external switch in order to select the antenna providing the highest measured RSSI among two antennas.

When the antenna switching mode is enabled, the two antennas are repeatedly switched during the reception of the preamble for each packet, until the carrier sense threshold is reached. From this point, the antenna with the highest power is selected and the switch position is frozen.

This feature requires:

- to have a preamble must be sufficiently long to allow the algorithm choosing the final antenna,
- to select the static carrier sensing mode (CS_MODE[1:0] = “00” in AS_QI_CTRL register),
- to have access to the external antenna switch component through a GPIO of the SoC.

The algorithm is controlled by the following parameters:

- antenna switching enabling
  - AS_ENABLE bit in the dynamic *ADDITIONAL_CTRL register (ADDITIONAL_CTRL)*
  - disabled by default.
- measure time
  - AS_MEAS_TIME[2:0] bit field in the AS_QI_CTRL static register,
  - controls the time interval for RSSI measurement
  - the actual measurement time (duration of selection for each antenna) is:

$$T_{\text{AS meas time}} = \frac{16 \times 2^{\text{CHFL\_E}} \times 2^{\text{AS\_MEAS\_TIME}}}{F_{\text{SYS}}}$$

29.11.7 Automatic frequency compensation (AFC)

In the frequency modulation-based modes, the AFC algorithm allows to compensate, within certain limits, a relative frequency error between the transmitting device and the receiving device cause by crystal inaccuracies for instance.

Due to the demodulation algorithm employed, any frequency error results in a DC offset in the demodulated signal before slicing. The basic operating principle of the AFC is to detect the minimum and maximum signal frequencies and to calculate a correction to remove the DC offset.

Such correction supports two modes:

- slicer correction mode (default mode): applied at the slicer level as an offset compensation
  - this mode allows a quick recovery of the frequency error but does not prevent part of the received signal power to be cut by the channel filter
  - this mode is recommended for the normal operations
- IF correction mode (optional mode): used to adjust the second IF conversion stage frequency
  - this mode adjusts the signal frequency before entering the channel filter thus avoiding power loss but requires a longer period to settle.

In parallel of applying the correction, the AFC block also provides the estimated frequency error information in the AFC_CORRECTION[7:0] bit field of the *QI_INFO register (QI_INFO)* (status register).

If the frequency is known to be constant (for example, communication always occurs between the same pair of devices), this value can directly be used to correct the programmed Intermediary Frequency in the IF_OFFSET_xx[12:0] bit fields of the *IF_CTRL register (IF_CTRL)* (static register, with xx = DIG or ANA).

A further control on the output sign permanence of AFC corrected signal has been implemented to improve the capability to receive long series of 0s or 1s. If a permanence of sign is detected, using a proper counter with a configurable counter threshold, the AFC frequency/amplitude offset is not updated, quite like an opening the loop of the AFC mechanism. If a sign change is detected the AFC return in its normal working mode.

In addition, a reinitialization option has been introduced to improve the AFC behavior on frames with PREAMBLE smaller than 64 bits. The option is controlled through the AFC3_CONFIG Radio register new [7:6] bit field called AFC_REINIT_OPTION[1:0]:

- “00”: no reinitialization allowed
- “01”: reinitialization based on RSSI ramping
- “10”: reinitialization based on abnormal Fdev
- “11”: reinitialization based on both RSSI and Fdev behaviors (default configuration)

In order to guarantee both fast lock and smooth tracking, the AFC has a fast and slow mode.

The AFC starts in fast mode as soon as the signal threshold is passed and switches to slow mode after a programmable period.

The AFC is controlled by the following parameters located in Radio registers:

- AFC enabling:
  - AFC_EN bit in the *AFC2_CONFIG register (AFC2_CONFIG)*
  - enabled by default.
- Selection of the AFC mode:
  - AFC_MODE bit in the *AFC2_CONFIG register (AFC2_CONFIG)*
  - slicer correction mode selected by default.
- Selection of AFC the start event:
  - AFC_INIT_MODE bit in the *AFC3_CONFIG register (AFC3_CONFIG)*
  - sets the start event of the AFC:

    when 0: the AFC starts once the measured RSSI is above the programmed threshold

    when 1: the AFC starts after the SYNC word detection and the first estimation of frequency offset is done by the sync correlator
  - RSSI above threshold start mode is selected by default.
- Possibility to freeze the AFC correction on SYNC word detection event:
  - AFC_FREEZE_ON_SYNC bit in the *AFC2_CONFIG register (AFC2_CONFIG)*
  - when set, the value of the frequency correction is no more updated after SYNC word detection vent
  - set by default.
- Selection of the fast period duration:
  - AFC_FAST_PERIOD[7:0] bit field in the *AFC1_CONFIG register (AFC1_CONFIG)*
  - sets the length of the fast period in number of samples
  - if set to 0, the fast period ends with the SYNC word detection, that is, the fast gain is used up to the SYNC word detection and the slow gain is used after the SYNC detection.
  - the recommended value is such that the fast period equals the preamble length. Since the algorithm operates typically on 2 samples per symbol, the programmed value should be twice the number of preamble symbols.
- AFC fast gain
  - AFC_FAST_GAIN_LOG2[3:0] bit field in the *AFC0_CONFIG register (AFC0_CONFIG)*
  - sets the loop gain for the fast mode
- AFC slow gain
  - AFC_SLOW_GAIN_LOG2{3:0] bit field in the *AFC0_CONFIG register*

(AFC0_CONFIG)
- sets the loop gain for the slow mode
- AFC peak detection leakage selection
  - AFC_PD_LEAKAGE[4:0] bit field in the AFC2_CONFIG register
  - sets the decay speed of the min / max frequency peak detector
  - *Note: default value is 8 while the recommended value is 4*
- AFC sign permanence check enabling:
  - AFC_SIGN_PERM_CHECK bit in the AFC3_CONFIG register
  - disabled by default
- AFC sign permanence check mechanism counter threshold:
  - AFC_TH_SIGN_THR[3:0] bit field in the AFC3_CONFIG register
  - default value is 10

In addition, the AFC uses the RSSI threshold (RSSI_THR[8:0] bit field in AS_QI_CTRL static register) information.

29.11.8 Symbol timing recovery

The MR_SubG IP supports two different algorithms for the timing recovery. The selection of the algorithm is done with the CLKREC_ALGO_SEL bit in the *CLKREC_CTRL1 register (CLKREC_CTRL1)* (radio register):

- CLKREC_ALGO_SEL = 0: a first order algorithm is used (shortly referred to as DLL)
- CLKREC_ALGO_SEL = 1: a second order algorithm is used (shortly referred to as PLL)

In addition, the settings of the following parameters affect the behavior of the clock recovery algorithms:

- Post filter length:
  - PSTFLT_LEN bit in the *CLKREC_CTRL0 register (CLKREC_CTRL0)* (radio register)
  - controls the length of the demodulator post-filter
  - setting this bit to ‘1’ may improve the demodulation performance but requires a slower recovery
  - default value is the recommended one (0).
- RSSI threshold:
  - RSSI_THR[8:0] in the static *AS_QI_CTRL register (AS_QI_CTRL)*
  - sets the minimum signal power above which the timing recovery is started.

DLL mode (default mode)

The DLL algorithm, based on a first order loop, is only able to control the delay of the local bit-timing generator in order to align it to the received bit period. If there is an error between the actual received bit and the nominal one, the relative edges drift over time and the algorithm periodically applies a delay correction to recover. Since in presence of long sequences of zeroes or ones it is not possible to estimate any timing error, the loop ends to lose lock if the period error is large (greater than 3%).

The convergence speed of the loop is controlled by the K_P parameter with a smaller value yielding a faster loop. This KP parameter is defined in CLKREC_P_GAIN_FAST / CLKREC_P_GAIN_SLOW bit field of the CLKREC_CTRL0 / CLKREC_CTRL1 Radio registers.

Recommended value for K_P for all modulations and data rate is 1 or 2.

PLL mode

The PLL algorithm tracks the phase error of the local timing generator relative to received bit period and controls both frequency and phase to achieve the timing lock. Once the relative period error has been estimated and corrected (for instance during the preamble phase), even in presence of long sequences of zeroes or ones, the loop is able to keep lock.

In order to improve the performance of the algorithm, two sets of gain coefficients (K_I and K_P) can be configured to be used before and after the SYNC word detection:

- CLKREC_I_GAIN_FAST and CLKREC_P_GAIN_FAST: used before the SYNC word detection,
- CLKREC_I_GAIN_SLOW and CLKREC_P_GAIN_SLOW: used after the SYNC word detection.

29.12 Radio register manager (RRM)

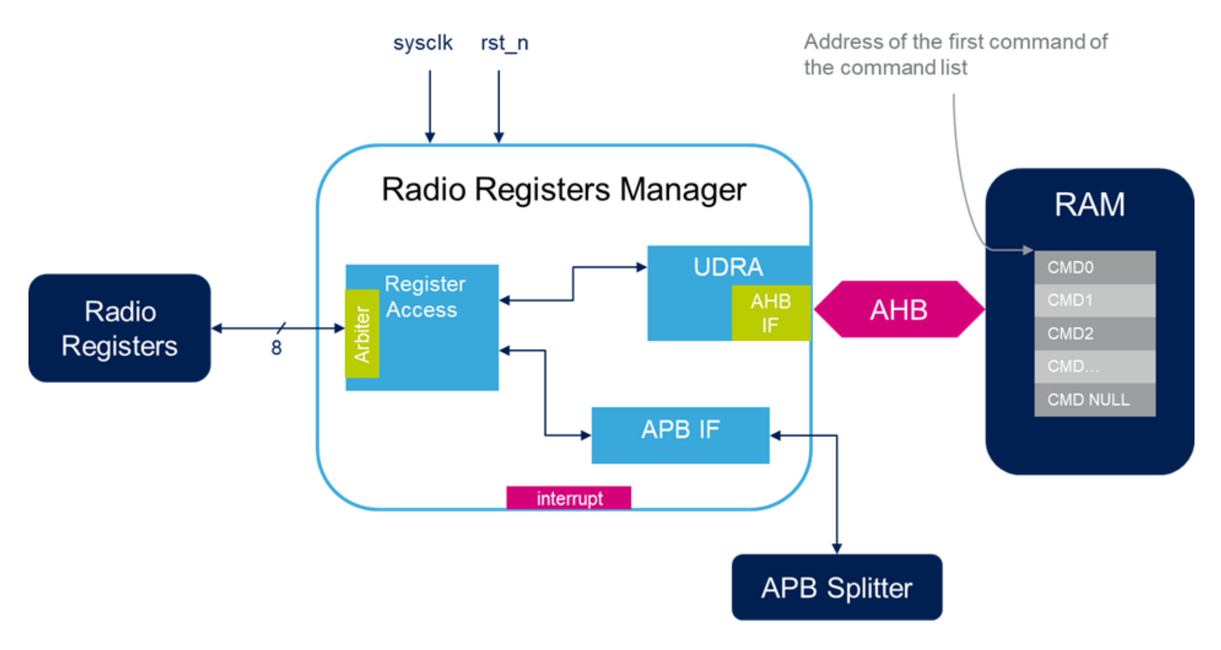
The radio register manager (RRM) is a hardware block managing the radio registers accesses. It is mainly used for the radio register automatic restoration after a low power mode exit.

29.12.1 Overview

The RRM embeds three features:

- Radio register direct access through APB
  - the SW can directly read and write the Radio Registers as classical APB registers
- Radio register access through UDRA using RAM link list
  - the HW can restore automatically some Radio Registers values thanks to a link list in RAM
  - this restoring has an interest for low power exit chained with a RF transfer
- an arbiter to manage concurrent accesses between APB and UDRA

Figure 288. RRM overview



29.12.2 UDRA

The Unified Direct Register Access block is an AHB master that allows the software preparing some commands in a command list located in the retention RAM. Those commands execute read from and write into the radio registers.

The main interest of the UDRA is to restore some specific Radio registers after a Low Power mode sequence to start a RF sequence while the CPU is still booting and not yet available to manage.

Some interruption flags are linked to the UDRA block in the RRM:

- on a command start event,
- on a command end event.

UDRA usage

The sequence to use the UDRA is the following:

- the software writes a command list in the retention RAM,
- the software provides to the UDRA through the Misc *RRM_UDRA_CTRL register (RRM_UDRA_CTRL)*:
  – a CMDLIST_PTR_OFFSET[15:0] bit field representing the address pointer of this command list (16-bit offset versus SoC RAM base address).

Note: There is no 32-bit aligned address constraint on the RRM command list pointer.

  – a CMDLIST_PTR_VALID bit to inform if a command list is present or not in RAM
- on any MR_SubG IP reset exit, the RRM UDRA checks the CMDLIST_PTR_VALID bit value to start (if 1) or not (if 0) the execution of the command list
- the SW can also launch the execution of the command list in RAM by writing 1 in the RRM_CMD_REQ action bit located in the Misc. *RRM_UDRA_CTRL register (RRM_UDRA_CTRL)*.

UDRA command list format in RAM

The command list is a group of commands that can be:

- single Radio register write,
- burst Radio registers write (registers with consecutive addresses),
- single Radio register read,
- burst Radio registers read (registers with consecutive addresses)

Note: As the Radio registers can be read directly through APB by the CPU, the read command can be considered as useless at SW point of view.

The write and read command format is described in *Table 134*. **The SW must take care to always ends the command list by the NULL command. Else, the UDRA behavior is unpredictable.**

Note: Radio registers are considered as 8-bit registers on UDRA side. For this reason, the addressing is on 8-bit and address offset between 2 registers is +1 in UDRA mapping. Both address mappings (direct APB and UDRA) are visible in the Table 130: RADIO_REG_BLOCK register list.

For burst read/write, only the first radio register address has to be provided. Then, the address is automatically incremented by 1 at HW level on each new register access inside the burst.

Table 134. UDRA command format in RAM

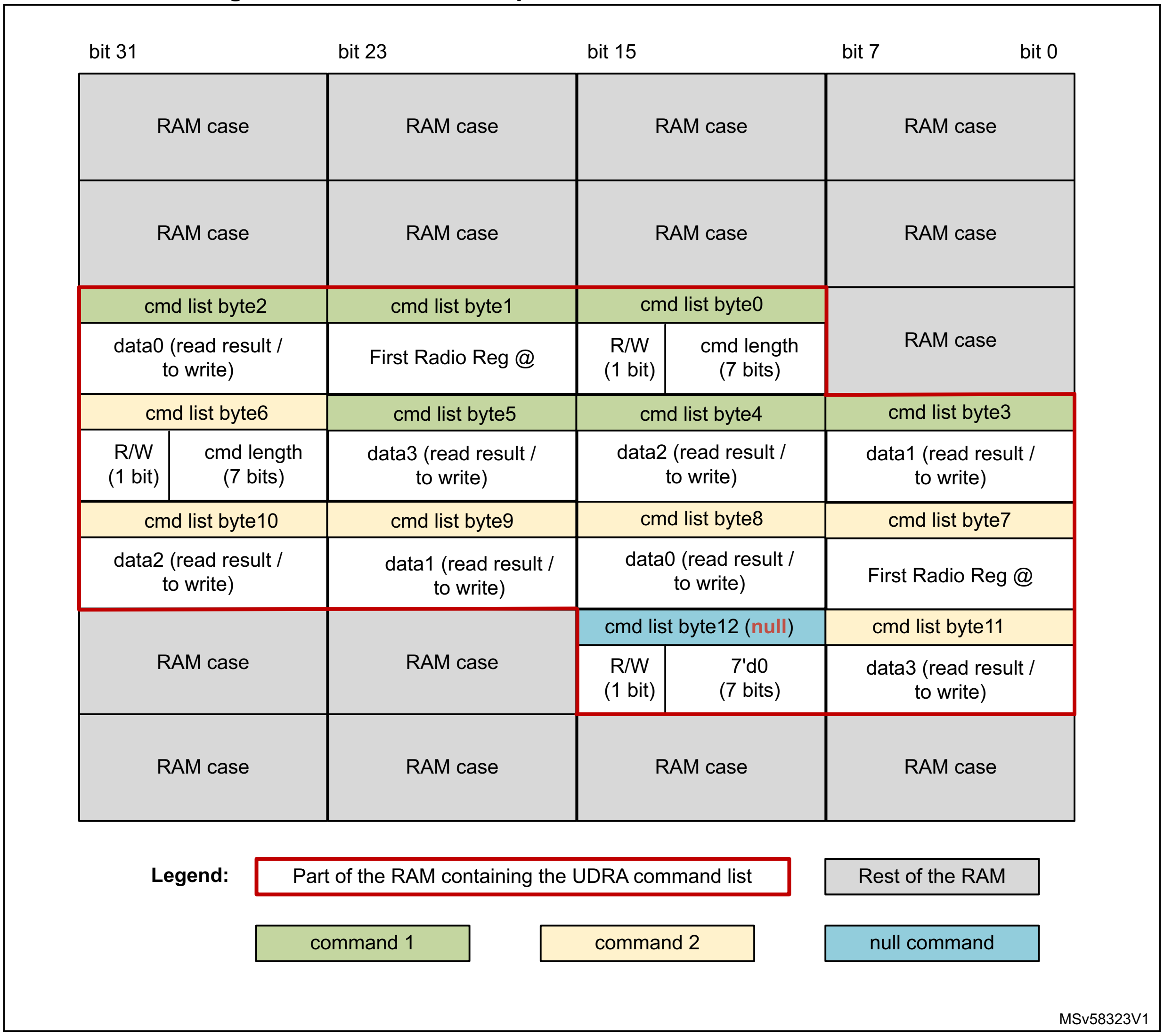
Byte number	Address in RAM	Byte value	Description
1	command_base_addr	0x--	bit7: 1=write / 0=read bit[6:0] = number of data to write or to read. n = number of data for the example in this table.
2	command_base_addr+1	8-bit address	Address of a Radio Register following the 8-bit address mapping (see *Table 130: RADIO_REG_BLOCK register list*)
3	command_base_addr+2	1st data	If write command: write first 8-bit data to be written. If read command: location where the first 8-bit read data is available at the end of the command execution.
4	command_base_addr+3	2nd data	Optional (depends on number of data to write/read). If write command: write second 8-bit data to be written. If read command: location where the second 8-bit read data is available at the end of the command execution.
…			
n+2	command_base_addr+(n+1)	nth data	Optional (depends on number of data to write/read). If write command: write nth 8-bit data to be written. If read command: location where the nth 8-bit read data is available at the end of the command execution.
n+3	command_base_addr+n+2	0x--	Optional: possible to chain other commands. bit7: 1=write / 0=read bit[6:0] = number of data to write or to read.

Table 134. UDRA command format in RAM

Byte number	Address in RAM	Byte value	Description
n+4	command_base_addr+n+3	8-bit address	Address of a Radio Register following the 8-bit address mapping (see *Table 130: RADIO_REG_BLOCK register list*)
n+5	command_base_addr+n+4	1st data	If write command: write first 8-bit data to be written. If read command: location where the first 8-bit read data is available.
…			
last	command_base_addr+last-1	0x00 / 0x80	MANDATORY. The null command (command with null length) must be added at the end of the command list. This is needed by the state machines of the UDRA to be informed they reach the end of the list.

Figure 289 shows a visualization of an UDRA command list in a SoC RAM.

Figure 289. Overview of a possible UDRA command list in RAM



Basic examples:

- Write AFC1_CONFIG=0x20 and AFC2_CONFIG=0xC4 (consecutive registers):

@command_base_addr = 0x82;Write 2 data

@command_base_addr+1 = 0x09;AFC1_CONFIG_ADDR;

@command_base_addr+2 = 0x20;1st data to write in AFC1_CONFIG

@command_base_addr+3 = 0xC4;2nd data to write in AFC2_CONFIG

@command_base_addr+4 = 0x00;null command

29.12.3 Direct APB access

This block allows the software to do direct read/write in the Radio registers through APB standard accesses. The Radio registers mapping is available in *Section 29.10: Radio register descriptions*.

The Radio registers control the parameters for the analog and few modulator/demodulator features.

Note: The radio registers are 8-bit only so the APB registers bit field[31:8] part is padded with 0.

29.12.4 RRM arbiter

The RRM embeds a round-robin arbiter to manage concurrent accesses to Radio registers by both APB and UDRA.

After reset, on first conflict arbitration, the arbiter gives the token to the UDRA. Then round-robin mechanism provides alternatively the token to APB and UDRA.

Recommendation:

The software must not write some radio registers through direct APB access if they are also modified through commands in RAM (through UDRA block). In this case, there is a risk of multi drivers in parallel and to lose coherency (no way to know which requester wrote the last).

29.13 Radio FSM

The Radio FSM manages the analog RFSUBG block startup and stop sequences depending on the requested command (TX, RX, calibrations...)

The Radio FSM is the sub-block that generates the PLL_LOCK_FAIL_F flag located in the *RFSEQ_IRQ_ENABLE register (RFSEQ_IRQ_ENABLE)*.

29.13.1 Radio FSM sequence

This paragraph lists the main steps in Radio FSM sequence.

- Exit IDLE to reach WAIT_ACTIVE2 as soon as a LOCK/RX/TX request is active
- Switch from WAIT_ACTIVE2 to ACTIVE2 if/when the system clock is confirmed to be on the accurate clock
- Goes from ACTIVE2:
  - to LOCKONRX (resp. LOCKONTX) state if the active request is a LOCK_RX (resp. LOCK_TX)
  - to TX state if the active request is TX command
  - to RX state if the active request is a RX command
- At the end of a RX or a TX, the Radio FSM manages several possible scenarios:
  - returns to LOCKONRX or LOCKONTX state to be able to restart a new transfer in the same direction and at the same frequency as the previous one (if BACK2LOCKON bit is set)
  - returns to ACTIVE2 (if BACK2ACTIVE2 bit is set) or IDLE until there is a need to start a new transfer

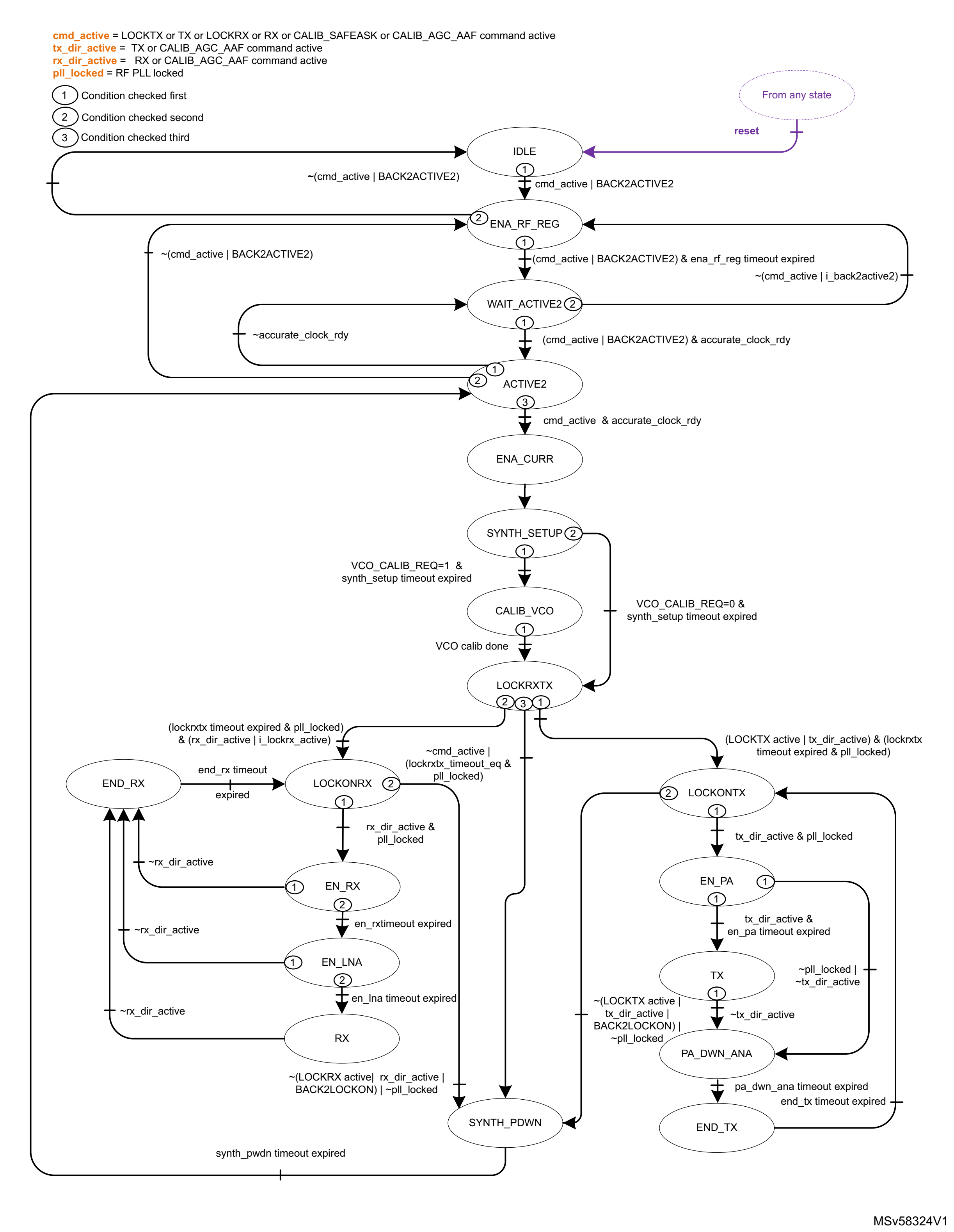
Note: It is also possible to send the Radio FSM to ACTIVE2 by setting the BACK2ACTIVE2 bit in the dynamic COMMAND register (COMMAND).

The current state information is available in the RADIO_FSM_INFO register (RADIO_FSM_INFO) (status register).

Figure 290 provides an overview of the Radio FSM states sequence.

Some transitions are triggered by hardware signals while others are managed through timeout windows programmable through the Radio registers. The paragraph Timeout durations used by the Radio FSM lists the different timeouts and where to program them. However, the default values should be the already needed ones and are not supposed to be modified.

Figure 290. Radio FSM overview



29.13.2 Radio FSM states overview

This section describes briefly the different states of the Radio FSM:

Table 135. Radio FSM state details

State	Goal of the state	Event to exit the state	Next state
IDLE	Default state when no RF activity is required	LOCK or TX or RX or calibration or back2active2 request active.	ENA_RF_REG
ENA_RF_REG	Enable the RF Regulator	No more command active and back2active2 bit low	IDLE
		RF_REG timeout expiration (to guarantee the RF_REG is ON)	WAIT_ACTIVE2
WAIT_ACTIVE2	Wait for confirmation the system clock is an accurate clock.	Accurate clock is present and a command or back2active2 is still active	ACTIVE2
		No more command active and back2active2 bit low	ENA_RF_REG
ACTIVE2	Indicates accurate clock is present	No more accurate clock present	WAIT_ACTIVE2
		No more command active and back2active2 bit low	ENA_RF_REG
		Accurate clock is present and a command is active	ENA_CURR
ENA_CURR	Enable the reference current block	One clock cycle state	SYNTH_SETUP
SYNTH_SETUP	Start the PLL analog block and wait for the RF regulator stabilization	SYNTH_SETUP timeout expiration and VCO calibration request is set.	CALIB_VCO
		SYNTH_SETUP timeout expiration and no VCO calibration is requested (external calibration used).	LOCKRXTX
CALIB_VCO	Calibration of the PLL	VCO calibration end signal received	LOCKRXTX
LOCKRXTX	Wait for the PLL lock information	LOCKRXTX timeout expiration and CALIB_SAFEASK or TX or LOCKTX command active and (PLL locked or RFFSM_IGNORE_PLL_LOCK bit = 1)	LOCKONTX
		LOCKRXTX timeout expiration and CALIB_AGC_AAF or RX or RX_IQBIST or LOCKRX command active and (PLL locked or RFFSM_IGNORE_PLL_LOCK bit = 1)	LOCKONRX
		No more command active or (PLL not locked at the end of the timeout and RFFSM_IGNORE_PLL_LOCK bit = 0)	SYNTH_PWDN

Table 135. Radio FSM state details (continued)

State	Goal of the state	Event to exit the state	Next state
LOCKONTX	State with PLL locked. This state can be/ – one clock cycle transition when CALIB_SAFEASK or TX command is active – a final state on LOCKTX request	CALIB_SAFEASK or TX command is active	EN_PA
		(No more CALIB_SAFEASK or LOCKTX or TX command active and BACK2LOCKON bit = 0) or PLL not locked	SYNTH_PDWN
EN_PA	Start the PA and wait during the DAC precharge	(PLL no more locked and RFFSM_IGNORE_PLL_LOCK bit = 0) or No more CALIB_SAFEASK or TX command active	PA_DWN_ANA
		EN_PA timeout expiration and CALIB_SAFEASK or TX command active	TX
TX	Transmission phase	End of TX or end of Safe-ASK calibra-tion	PA_DWN_ANA
PA_DWN_ANA	Delay to absorb the analog DAC dis-charge	PA_DWN_ANA timeout expiration	END_TX
END_TX	Delay to let the RF regulator stabilise after analog PA clock stop	END_TX timeout expiration	LOCKONTX
LOCKONRX	State with PLL locked. This state can be – one clock cycle transition when CALIB_AGC_AAF or RX or RX_IQBIST request is active – a final state on LOCKRX request	(PLL no more locked and RFFSM_IGNORE_PLL_LOCK bit = 0) and RX / RX_IQBIST / CALIB_AGC_AAF command active	EN_RX
		(No more CALIB_AGC_AAF or RX or RX_IQBIST command active and BACK2LOCKON bit = 0) or PLL not locked	SYNTH_PWDN
EN_RX	Start the analog blocks linked to RX chain (Mixer, RXADC, LNA, etc.) and the PGA precharge	No more CALIB_AGC_AAF or RX or RX_IQBIST command active	SYNTH_PWDN
		EN_RX timeout expiration	EN_LNA
EN_LNA	Disable the PGA precharge. *Note: the name of this state is no more relevant as LNA is already started in EN_RX state. It is used to let the RX chain stabilize after PGA precharge disable.*	No more CALIB_AGC_AAF or RX or RX_IQBIST command active	SYNTH_PWDN
		EN_LNA timeout expiration	RX

Table 135. Radio FSM state details (continued)

State	Goal of the state	Event to exit the state	Next state
RX	Phase of reception	End of RX or AGC/AAF calibration or RX_IQBIST sequences	END_RX
END_RX	Delay to let the RF regulator stabilise after analog RX chain stop	END_RX timeout expiration	LOCKONRX

29.13.3 Timeout durations used by the Radio FSM

The Radio FSM embeds a counter that uses different timeout duration (defined in the radio registers) to exit some states. The different timeouts are listed in the table below.

Table 136. Timeout versus Radio FSM state exit

State to exit	Register / bit field for timeout	Default value
EN_RF_REG	RF_FSM0_TIMEOUT[7:0] = ENA_RFREG_TIMER	0x0 = 0 µs
WAIT_ACTIVE2	No timeout. Exit on conditions (refer to *Table 135: Radio FSM state details* for the list of conditions)	N/A
ACTIVE2	No timeout. Exit on conditions (refer to *Table 135: Radio FSM state details* for the list of conditions)	N/A
ENA_CURR	No timeout. Exit immediately (1 clock cycle duration)	N/A
SYNTH_SETUP	RF_FSM1_TIMEOUT[7:0] = SYNTH_SETUP_TIMER	0x06 = 6 µs
CALIB_VCO	No timeout. Exit on end of calibration signal.	N/A
LOCKRXTX	If a VCO calibration is done: RF_FSM2_TIMEOUT[7:0] = VCO_CALIB_LOCK_TIMER If no VCO calibration: RF_FSM3_TIMEOUT[7:0] = VCO_LOCK_TIMER	0x50 = 80 µs 0x28 = 40 µs
LOCKONTX	No timeout. Exit on conditions (refer to *Table 135: Radio FSM state details* for the list of conditions)	N/A
EN_PA	RF_FSM5_TIMEOUT[7:0] = EN_PA_TIMER	0x19 = 25 µs
TX	No Timeout. Exit on conditions (refer to *Table 135: Radio FSM state details* for the list of conditions)	N/A
PA_DWN_ANA	RF_FSM6_TIMEOUT[7:0] = PA_DWN_ANA_TIMER	0x19 = 25 µs
TX_END	RF_FSM10_TIMEOUT[7:0] = END_TX_TIMER	0x06 = 6 µs
LOCKONRX	No timeout. Exit on conditions (refer to refer to *Table 135: Radio FSM state details* for the list of conditions)	N/A
EN_RX	RF_FSM4_TIMEOUT[7:0] = EN_RX_TIMER	0x0A = 10 µs
EN_LNA	RF_FSM7_TIMEOUT[7:0] = ENA_LNA_TIMER	0x05 = 5 µs
RX	No timeout. Exit on conditions (refer to *Table 135: Radio FSM state details* for the list of conditions)	N/A
RX_END	RF_FSM9_TIMEOUT[7:0] = END_RX_TIMER	0x06 = 6 µs
SYNTH_PDWN	RF_FSM8_TIMEOUT[7:0] = SYNTH_PDWN_TIMER	0x0A = 10 µs

Note: All the default values present in the radio registers about those timeouts are already adapted to the need and must not be modified for normal usage of the radio.

This leads to the following global durations:

- IDLE to TX: 111 µs if VCO calibration, 71 µs if no VCO calibration
- ACTIVE2 to TX: 111 µs if VCO calibration, 71 µs if no VCO calibration
- TX to ACTIVE2: 41 µs
- IDLE to RX: 101 µs if VCO calibration, 61 µs if no VCO calibration
- ACTIVE2 to RX: 101 µs if VCO calibration, 61 µs if no VCO calibration
- RX to ACTIVE2: 16 µs

Note: On cut2, IDLE to ACTIVE2 transition is only few 16 MHz clock cycles (LDO_RF timeout duration reduced to 0 by default)

29.14 Radio controller

The radio controller manages:

- the commands,
- the RX Timer,
- the RFSEQ_IRQ_STATUS flag generation when linked to a RX/TX transfer,
- the update signal to store the RX information in read-only global registers,
- the IQC correction value selection,
- the Time Capture feature.

29.14.1 Commands

The MR_SubG IP supports several commands to launch RF operations (transfers or calibrations).

Any command is launched by the action of writing its opcode in the global dynamic *COMMAND register (COMMAND)*.

The available commands are:

Table 137. Command list

Command name	Description	Command ID
NOP	No action This "no" command can be requested at any time, whatever the on-going command or in IDLE. *Note: to be used in case an intermediate SeqAction is needed at sequencer level without requesting any command or when the user just wants to send the Radio FSM in ACTIVE2 state by setting the BACK2ACTIVE2 bit sharing the same register.*	0x0
TX	Start a TX sequence	0x1
RX	Start a RX sequence	0x2
LOCKTX	Start a RF sequence up to PLL locked based on TX frequency	0x4
LOCKRX	Start a RF sequence up to PLL locked based on RX frequency	0x3
SABORT	Stop any on-going RX/TX/LOCKRX/LOCKTX command	0x5
CALIB_SAFEASK	Launch a PA Safe-ASK calibration to get the max safe PA code to be used	0xA
RELOAD_RX_TIMER	Reload with a new timeout and new stop conditions and restart the RX Timer	0x6
CALIB_AGC_AAF	Start needed HW features to run an AGC_ATTEN trim or an AAF cut off frequency trim sequence at SW level.	0xB

Note: The COMMAND register keeps its content until the SW writes something else. this means the latest requested command Opcode stays readable event after the command is over.

Most of the commands are exclusive and cannot be launched in parallel except for few ones. Launching a new command while the previous one is not completed may lead to a rejected command or an ignored command. The table below provides a summary of the treatment of concurrent command requests.

Table 138. Concurrent command behavior overview

Active command	New command request							
	LOCKRX	LOCKTX	RX	TX	SABORT	CALIB_AGC_AAF	CALIB_SAFEASK	RELOAD_RX_TIMER
None	**executed** lockrx_active =1	**executed** locktx_active =1	**executed** rx_active=1	**executed** tx_active=1	**rejected**	**executed**	**executed**	**ignored**
LOCKRX (lockrx_active=1)	**ignored** lockrx_active =1	**rejected** lockrx_active =1 locktx_active =0	**executed** rx_active=1 lockrxactive =0	**rejected** lockrx_active =1 tx_active=0	**executed** lockrx_active=0	**executed**	**rejected**	**ignored**
LOCKTX (lockrx_active=1)	**rejected** locktx_active =1 lockrx_active =0	**ignored** locktx_active =1	**rejected** locktx_active =1 rx_active =0	**executed** tx_active=1 rx_active =0	**executed** locktx_active=0	**rejected**	**executed**	**ignored**
RX (rx_active=1)	**rejected** rx_active=1 lockrx_active =0	**rejected** locktx_active =0 rx_active=1	**ignored** rx_active=1	**rejected** rx_active=1 tx_active=0	**executed** rx_active=0	**rejected**	**rejected**	**executed**
TX (tx_active=1)	**rejected** lockrx_active =0 tx_active=1	**rejected** tx_active=1 locktx_active =0	**rejected** rx_active=0 tx_active=1	**ignored** tx_active=1	**executed** tx_active=0	**rejected**	**rejected**	**ignored**
SABORT (sabort_req)	**rejected**	**rejected**	**rejected**	**rejected**	**ignored**	**rejected**	**rejected**	**ignored**
CALIB_AGC_AAF	**rejected**	**rejected**	**rejected**	**rejected**	**executed** rx_active=0 calib_agc_aaf_req =0	**ignored**	**rejected**	**ignored**
CALIB_SAFEASK	**rejected**	**rejected**	**rejected**	**rejected**	**executed** tx_active=0 calib_ask_req=0	**rejected**	**ignored**	**ignored**
RELOAD_RX_TIMER (reload pulse of A clock cycle)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table legend:

- executed: the new command is considered and executed in parallel of the on-going one.
- ignored: the new command is not considered, and no specific information is raised towards the SW.
- rejected: the new command is not considered, and the COMMAND_REJECTED_F flag (associated to an interrupt if enabled) is raised in the *RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)* to inform the SW.

TX command

One the APB write with Opcode = 0x1 occurs in the COMMAND dynamic register, the Radio Controller:

- requests the Radio FSM to in TX state,
- enables the Data Buffer management block

The user is informed the TX command is over thanks to the TX_DONE_F (and associated interrupt if enabled) available in the RFSEQ_IRQ_STATUS register once the frame has been transmitted.

For specific configurations, the TX command needs the SW interference to stop thanks to the SABORT command. This may happen in the following configurations:

- transmission in direct through GPIO mode,
- transmission in Direct through Buffer when PCKTLEN = 0.

In those cases, the HW has no knowledge of the moment to stop and so, waits for a SW decision to abort.

The TX command can only be requested when no other command is active except a LOCKTX command. Otherwise, the TX command is rejected (except in case of concurrency with another on-going TX command in which case it is ignored).

RX command

One the APB write with Opcode = 0x2 occurs in the COMMAND dynamic register, the Radio Controller:

- requests the Radio FSM to in X state,
- enables the Data Buffer management block

The user is informed the RX command is over thanks to flags (and associated interrupt if enabled) available in the *RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)*:

- RX_OK_F flag
  – when a frame has been received without CRC error,
- RX_CRC_ERROR_F flag
  – when a frame has been received but a mismatch is detected between the received CRC an the internally calculated CRC,
- RX_TIMEOUT_F flag
  – when the RX sequence has been stopped by the HW on a RX Timer timeout,
- FAST_RX_TERM_F flag
  – when the Fast termination mode is enabled and the measured RSSI stays below the programmed threshold during the Fast RX termination window.

For some specific configurations, the RX command needs the SW interference to stop thanks to the SABORT command. This may happen in the following configurations:

- reception in Normal mode but no RX Timeout programmed (RX_TIMEOUT[22:0]=0). Then, the RX command stays active until a frame is received
- reception in Direct through GPIO mode
- reception in Direct through Buffer mode

In these cases, the HW has no knowledge of the moment to stop and waits for a SW decision to abort.

The RX command can only be requested while no other command is active except a LOCKRX command. Otherwise, the RX command is rejected (except in case of concurrency with another on-going RX command in which case it is ignored).

LOCKTX command

Once the APB write with Opcode = 0x3 occurs in the COMMAND dynamic register, the Radio controller requests the Radio FSM to go to LOCKONTX state.

From there, the Radio FSM can be:

- sent to TX state by a TX command or a CALIB_SAFEASK command,
- sent back to ACTIVE2 (or IDLE if BACK2ACTIVE2 bit is low) by a SABORT.

The aim of staying in LOCKONTX state can be:

- to limit the latency to on TX (only PA startup step left before to reach TX),
- to stabilize the XO frequency versus temperature increase generated by the RF startup by staying long enough in this state.

Note: In LOCKONTX state the power consumption add-on is not transparent as the RF PLL is running.

This command can only be requested while no other command is active, else it is rejected (except for a concurrent LOCKTX command in which case it is ignored).

LOCKRX command

Once the APB write with Opcode = 0x4 occurs in the COMMAND dynamic register, the Radio controller requests the Radio FSM to go to LOCKONRX state.

From there, the Radio FSM can be:

- sent to RX state by a RX command or a CALIB_AGC_AAF command,
- sent back to ACTIVE2 (or IDLE if BACK2ACTIVE2 bit is low) by a SABORT.

The aim of staying in LOCKONRX state can be:

- to limit the latency to on RX (only RX chain startup step left before to reach RX),
- to stabilize the XO frequency versus temperature increase generated by the RF startup by staying long enough in this state.

Note: In LOCKONRX state the power consumption add-on is not transparent as the RF PLL is running.

This command can only be requested while no other command is active, else it is rejected (except for a concurrent LOCKRX command in which case it is ignored).

SABORT command

The SABORT command (Opcode = 0x5) is used to stop any other active command.

Note: The RELOAD_RX_TIMER command is not concerned as it lasts 1 fast clock period.

The SABORT command informs the SW once it is completed:

- either by a COMMAND_REJECTED_F flag if no command is active when the SABORT is requested,
- or by a SABORT_DONE_F flag when the SABORT has been executed.

Note: Launching a second SABORT command before the confirmation of the previous one (rejected or done) leads randomly (depending on the state of the HW when the second command is issued) either to a command rejected or to an ignored command for which the SW never receives any acknowledge.

The SABORT_DONE_F flag indication is raised only once the interrupted command is effectively over. So, a latency is present between the SABORT command request and the done information. The SABORT is considered as done once the Radio FSM goes back to its default state (IDLE or ACTIVE2 or LOCKONxX depending on BACK2ACTIVE and BACK2LOKCON bits).

RELOAD_RX_TIMER command

Writing 0x6 in the COMMAND_ID[3:0] bit field generates a reload on the RX timer if this timer is enabled (corresponding to Radio FSM in RX state).

Before issuing the RELOAD_RX_TIMER command, the SW may:

- program a new timeout target through the RX_TIMEOUT[22:0] bit field in the dynamic RX_TIMER register,
- program new stop condition(s) through the RX_TIMER[31:28] bits.

The reload action must be requested while the Radio FSM is in RX state:

- if the RX Timer is currently active, it is reloaded on-the-fly with a new timeout and/or stop condition targets,
- if the RX Timer is stopped (due to a previous stop condition match or because initially programmed with 0), it restarts using the new timeout and/or new stop condition targets.

Refer to *Section 29.14.3: RX Timer* for details.

CALIB_SAFEASK command

The aim of the SafeASK calibration is to detect the operation mode of the output transistor of the class-E PA: linear vs saturation or ron zone vs current source zone. This allows to get the maximum output power on the antenna before PA enters in ohmic zone (or ron zone) causing deleterious AM to PM conversion during the ASK modulation.

The sequence to run a safe ASK calibration is the following:

1. Program the radio IP as for as for an ASK/OOK TX in FIR mode:
   – select the ASK/OOK modulation in MOD0_CONFIG[22:20] = MOD_TYPE[2:0],
   – select the FIR mode in PA_CONFIG register (see section *PA_MODE = 01*)
2. Define the calibration window by providing:
   – the minimum power to try in PA_LEVEL_3_0[7:0] = PA_LEVEL0[7:0] bit field
   – the maximum power to target in PA_LEVEL_7_4[31:24] = PA_LEVEL7[7:0] bit field
3. Define the result format (dB or linear) through the PA_CONFIG[13] = LIN_NLOG
4. Launch the calibration through the CALIB_SAFEASK command (0x0A Opcode):
   – the Radio Controller requests the Radio FSM to go to TX state,
   – once in TX state is reached, a PA SafeASK calibration is launched,
   – the Radio FSM goes back to its default state once it receives the “end of calibration” information from the PA interface block
5. The user is informed the CALIB_SAFEASK command is completed thanks to the SAFEASK_CALIB_DONE_F flag (and associated interrupt if enabled) available in the RFSEQ_IRQ_STATUS register.
6. The result of the calibration is available in the PA_CODEMAX[7:0] bit field of the *PA_SAFEASK_OUT register (PA_SAFEASK_OUT)* (status register).

The command can only be requested while no other command is active except a LOCKTX command. Otherwise, the CALIB_SAFEASK command is rejected (except in case of concurrency with another on-going CALIB_SAFEASK command in which case it is ignored).

CALIB_AGC_AAF command

The goal of the CALIB_AGC command is to help the SW in the trimming process of LNA+BOM AGC stage to guarantee 6dB by step.

The AGC calibration is described in a dedicated application note.

The user is informed the CALIB_AGC_AAF command is completed thanks to the AGC_AAF_CALIB_DONE_F flag (and associated interrupt if enabled) available in the RFSEQ_IRQ_STATUS register.

The result of the RSSI measurement is available in the RSSI_LEVEL_RUN[8:0] bit field of the *RX_INDICATOR register (RX_INDICATOR)* (status register).

The command can only be requested while no other command is active except a LOCKRX command. Otherwise, the CALIB_AGC_AAF command is rejected (except in case of concurrency with another on-going CALIB_AGC_AAF command in which case it is ignored).

29.14.2 BACK2ACTIVE2 and BACK2LOCKON bits

Some complementary bits have been added on MSB part of the dynamic *COMMAND register (COMMAND)*. They offer options about the default state of the Radio FSM in some scenarios.

BACK2ACTIVE2 bit

This bit is used to send the Radio FSM from IDLE to ACTIVE2 even if no command is active. If this bit is high, the default state of the Radio FSM is ACTIVE2 instead if IDLE.

BACK2ACTIVE is useful to reduce the latency on RF command execution by removing the RF regulator settling time defined in the *RF_FSM0_TIMEOUT register (RF_FSM0_TIMEOUT)* (radio register).

BACK2LOCKON bit

This bit is used to request the Radio FSM:

- to stay in LOCKONTX state when coming back from TX state,
- to stay in LOCKONRX state when coming back from RX state.

This bit has no impact on the Radio FSM until it reaches the LOCKONTX or LOCKONRX state.

The BACK2LOCKON bit has been added in case it may help for some specific scenarios that would like to chain RX (respectively TX) sequences at the same frequency with very low latency in between.

Note: If the user wants to switch from RX to TX (resp. from TX to RX) or to have a channel hopping / frequency change, this bit must not be set as in this case the Radio FSM must go back to ACTIVE2 and restart the Sigma delta block and PLL with new frequency target.

Recommendation: this bit must be used as few as possible and only if it brings a real added value to the targeted scenarios.

29.14.3 RX Timer

The RX Timer is used to abort automatically a RX phase if the defined stop condition does not occur before the programmed timeout expires.

The possible stop conditions can be selected in the dynamic *RX_TIMER register (RX_TIMER)* dynamic register:

- CS information (RSSI over threshold)
  - the RX timer stops when the carrier_sense (CS) is raised (in parallel this signal also sets the CS_F flag in the *RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)*)
  - the carrier_sense signal is raised when the RSSI measured in the antenna is greater than the RSSI threshold programmed by the SW (in the AS_QI_CTRL[8:0] = RSSI_THR[8:0])
  - the stop condition is enabled by setting the RX_TIMER[28] = RX_CS_TIMEOUT_MASK bit.
- PQI information
  - the RX timer stops when the valid preamble information is raised (in parallel this signal also sets the PREAMBLE_VALID_F flag in the *RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)*)
  - the valid preamble signal is raised when a valid preamble is detected by the demodulator (or if PQI_THR[3:0] bit field is 0 in the AS_QI_CTRL static register),
  - the stop condition is enabled by setting the RX_TIMER[29] = RX_PQI_TIMEOUT_MASK bit.
- SQI information
  - the RX timer stops when the valid SYNC signal is raised (in parallel this signal also sets the SYNC_VALID_F flag in the *RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)*)
  - the valid SYNC signal is raised when a valid SYNC word is detected by the demodulator,
  - the stop condition is enabled by setting the RX_TIMER[30] = RX_SQI_TIMEOUT_MASK bit.
- An additional bit is present to use a combination of the above stop conditions instead of single information: the RX_TIMER[31] = RX_OR_nAND_SELECT:
  - RX_OR_nAND_SELECT = 0: a AND is applied on the enabled stop conditions

    For example, if RX_CS_TIMEOUT_MASK and RX_PQI_TIMEOUT_MASK bits are set, the stop condition is raised when both flags are high.
  - RX_OR_nAND_SELECT = 1: a OR is applied on the enabled stop conditions

    For example,The RX Timer starts are soon as the Radio FSM enters the RX state, except if the timeout duration has been programmed to 0.

The RX timer stops when it matches a stop condition or on a timeout expiration.

As long as the Radio FSM is in RX state, the RX Timer can be restarted through the RELOAD_RX_TIMER command. This command can also be used while the timer is currently running, to update the timeout target or the stop condition mask.

Concerning the timeout duration:

- it is programmable through the RX_TIMEOUT[22:0] bit field of the RX_TIMER dynamic register,
- its time unit is the interpolated absolute time (16 x slow clock frequency).

Table 139. RX Timer duration versus slow clock frequency

Slow clock frequency	Period unit	Maximum timeout duration
24 kHz (min)	2.6 µs (384 kHz period)	21.84 s
32 kHz (typ)	1.95 µs (512 kHz period)	16.38 s
49 kHz (max)	1.27 µs (784 kHz period)	10.7 s

To disable the RX Timer, the user must program 0 as target timeout duration.

The SW is informed about a RX Timer event as follows:

- RX_TIMEOUT_F:
  - raised when the RX Timer stops due to a timeout expiration
  - in this case, the reception phase is aborted by HW and the Radio FSM goes back to its requested default state
- RXTIMER_STOP_CDT_F:
  - raised when the RX Timer stops due to a match with the stop condition(s) mask,
  - in this case, the reception goes on until a valid frame is received
  - the RX Timer can be restarted with the RELOAD_RX_TIMER command if the Radio FSM is still in RX state which may lead to a further timeout event or stop condition match event.

An example of the pair RX Timer / RELOAD_RX_TIMER command could be:

1. Launch a RX command with a RX Timer programmed with a duration1 and stop condition based on CS_F
2. If the SW receives a RXTIMER_STOP_CDT_F event, the radio has detected power on the antenna but no guarantee it is a valid frame
3. the SW relaunches the RX Timer with a new duration2 based on valid PREAMBLE detection stop condition
4. if the SW receives a RXTIMER_STOP_CDT_F event, the radio has detected a valid preamble but no guarantee that the coming SYNC word matches
5. the SW relaunches the RX Timer with a new duration3 based on valid SYNC word detection stop condition

This allows chained timeout sequences to be proposed.

29.14.4 Fast RX termination

Fast RX termination is a feature allows stopping quickly the reception if no relevant power is measured on the antenna.

The Fast RX Termination mode is enabled through the FAST_CS_TERM_EN bit in the dynamic *FAST_RX_TIMER register (FAST_RX_TIMER)* register.

When enabled, the behavior is the following:

- a RX command is requested in the COMMAND register,
- once the Radio FSM is in RX state, the RX_Top starts an internal timer programmable through the FAST_RX_TIMEOUT[7:0] bit field in the FAST_RX_TIMER register,
- on fast timer expiration, the Radio Controller checks the carrier sense information:
  - if set, the Radio controller starts the RX Timer and rest of the RX sequence is managed as explained in the RX Timer section
  - if not set, the Radio controller stops the RX sequence and the FAST_RX_TERM_F flag in the RFSEQ_IRQ_STATUS register is set to explain the reason of the RX termination.

This mode can be used for sniff scenarios where the goal is to stop the reception and goes back in low power mode if no relevant power is seen on the antenna at wakeup and to retry later.

Although both fast RX termination and RX timers can be used to keep or stop the radio in RX state, depending on the presence - or otherwise - of power on the antenna, they behave differently.

The fast RX termination timer always runs until the timeout expires before it checks the carrier sense real-time information:

- If the carrier sense information is high (power on the antenna is higher than programmed threshold): the radio stays in RX state and starts the RX Timer (if enabled).
- If the carrier sense information is low (no power detected in the antenna): the radio exits the RX state.

Otherwise, the RX Timer stops as soon as it detects a stop condition and makes the radio exit the RX state only if its timeout expires (no stop condition met during the timeout window)

For details on carrier sense signal management, refer to *Section 29.11.5: Carrier sense*.

29.14.5 Hardware analog failure information

The analog elements may report some dysfunction during a RF sequence that can lead to a status / interrupt information up to a HW abort of the RF transfer, depending on the malfunction.

The malfunctions are reported in the following stats registers:

- the *RFSEQ_STATUS_DETAIL register (RFSEQ_STATUS_DETAIL)* providing the exact problem source
- the *RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)* through HW_ANA_FAILURE_F bit which is raised as soon as one of the RFSEQ_STATUS_DETAIL bits linked to an analog issue is raised. This bit is associated to an interrupt if enabled.

Those analog malfunctions can be:

- PLL lock failure:
  - The Radio FSM monitors the PLL lock information at the end of the LOCKRXTX state and during the LOCKONRX, LOCKONTX and EN_PA states. If the PLL is not locked or unlocks during this monitoring, the Radio FSM aborts the sequence and goes back to its default state.
  - This information is treated by the Radio Controller as a termination of the associated command.
  - The SW is informed of the event:

    by the PLL_LOCK_FAIL_F bit of the RFSEQ_STATUS_DETAIL register,

    by the HW_ANA_FAILURE_F flag in the RFSEQ_IRQ_STATUS register and associated interrupt if enabled.
- PLL unlock event
  - If the PLL unlocks (PLL lock information falling edge detection):

    the PLL_UNLOCK_F bit of the RFSEQ_STATUS_DETAIL register is raised

    the HW_ANA_FAILURE_F bit of the RFSEQ_IRQ_STATUs register is raised
  - if it occurs outside the LOCKONRX, LOCKONTX and EN_PA Radio FSM states , this event has no impact on the rest of the sequence. It is the responsibility of the SW to interrupt the RF transfer on this information.
- PLL calibration error
  - If the calibrator block indicates a calibration error at the end of the PLL calibration phase:

    the PLL_CALAMP_F bit of the RFSEQ_STATUS_DETAIL register is raised if the calibration error reason comes from the amplitude calibration

    the PLL_CALFREQ_F bit of the RFSEQ_STATUS_DETAIL register is raised if the calibration error reason comes from the frequency calibration

    the HW_ANA_FAILURE_F bit of the RFSEQ_IRQ_STATUS register is raised
  - this event has no impact on the rest of the sequence. It is the responsibility of the SW to decide to interfere or not on the on-going RF sequence.

The PLL lock failure is the only event that creates an HW abort reaction from the Radio FSM and so from the Radio Controller.

29.14.6 Time capture feature

The Time Capture feature allows recording the interpolated absolute time in the TIME_CAPTURE[31:0] read-only status register when a specific event is detected.

The enabling of the feature and the selection of the trigger event are done through the ADDITIONAL_CTRL[22:20] = TIME_CAPTURESEL[2:0] bit field as follows:

- TIME_CAPTURESEL[2:0] = 000:
  - the feature is disabled, no time capture occurs
- TIME_CAPTURESEL[2:0] = 001:
  - the interpolated absolute time is latched on end of transmission information,
  - the trigger signal is the end of TX information provided by the TX_Top block (before TX done flag is set and the Radio FSM goes back to ACTIVE2/IDLE state)
- TIME_CAPTURESEL[2:0] = 010:
  - the interpolated absolute time is latched on end of reception information whatever the CRC result (RX OK or CRC error),
  - the trigger signal is the end of reception signal provided by the RX_Top block (before RX done or timeout or CRC error flag is set and the Radio FSM goes back to ACTIVE2/IDLE state)

Note: If the RX is stopped by a SABORT command, no Time Capture occurs in this configuration.

- TIME_CAPTURESEL[2:0] = 011:
  - the interpolated absolute time is latched on the SYNC word detection event,
  - the trigger signal is the same as the setting the SYNC_VALID flag
- TIME_CAPTURESEL[2:0] = 1xx:
  - Reserved for future used (no action required)

The trigger selection is exclusive so only one trigger source can be selected at a time.

The TIME_CAPTURE register keeps the recorded time until a new valid Time Capture event occurs.

29.15 Data buffers

The data to transmit (when in TX) and the received data (when in RX) are stored in the RAM of the device embedding the MR_SubG IP. The Data Buffer Manager (DBM) is an AHB master interfacing between the RAM of the device and the TX_Top / RX_Top blocks of the radio.

The Data Buffers are used in transmission or reception except:

- for direct through GPIO mode for both TX and RX,
- for PN mode in TX.

29.15.1 Data buffer mechanism

The MR_SubG IP uses the RAM of the device instead of internal FIFOs to manage the data to transmit and received data.

The RAM usage is the following:

- the MR_SubG IP uses two Data Buffers: DATABUFFER0 and DATABUFFER1,
- both buffers have a common programmable size (in byte),
- the location of each Data Buffer in RAM is defined by a programmable 32-bit aligned pointer,
- the DBM (Data buffer Manager) always does 32-bit RAM accesses:
  – in TX, read only full words in RAM, does not use the extra MSByte if buffer size in not modulo 4,
  – in RX, write full words even if buffer size is not modulo 4 (see associated warning below).
- the two Data Buffers are used successively, in a rotation way: the DBM always starts a transfer from the DATABUFFER0, then once fully used (read for a TX, written for a RX), switches to the DATABUFFER1. The DBM then switches back to the DATABUFFER0 once the DATABUFFER1 is also fully used.
- Each time a Data Buffer is used, the SW is informed by status flags (DATABUFFER0_USED_F and DATABUFFER1_USED_F), associated to an interrupt if enabled. An intermediate information can be generated thanks to a programmable threshold to be informed before the “fully used” criteria.
- It is the responsibility of the SW to treat the used Data buffer content (refill in case of TX, content read in case of RX) before the DBM reuses it. This can also be mitigated by defining the most appropriate size of the buffer to fit the targeted use-case.

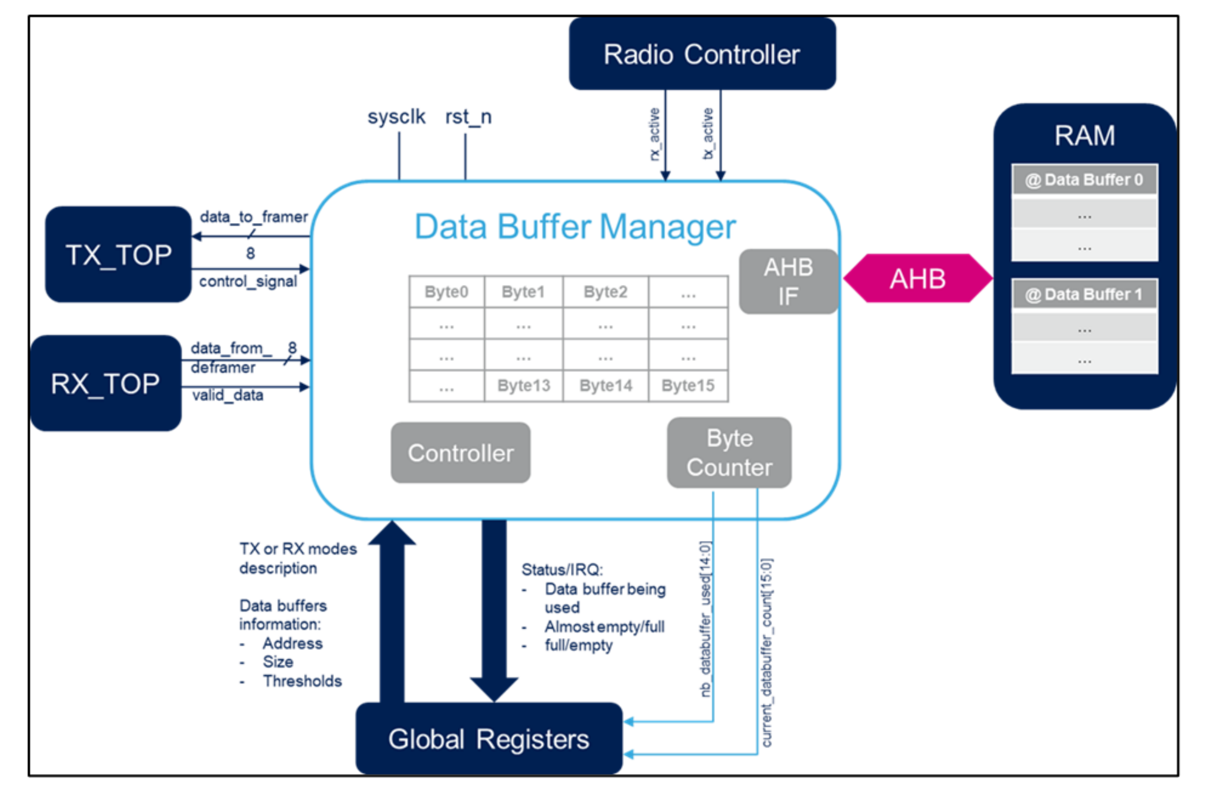
Warning: For SW to avoid RAM corruption in reception:

- **The DATABUFFER size can have a byte granularity to switch from one DATABUFFER to another when the exact number of bytes have been reached**
- **However, the buffers allocated in RAM must always have a word (32-bit) granularity as the DBM always accesses the RAM word by word. The extra bytes in RAM are dummy but written anyway so these dummy bytes must be allocated in RAM.**

29.15.2 Data buffer manager overview

The following figure provides an overview of the DBM.

Figure 291. Data Buffer Manager overview



The DBM:

- reads the data to transmit in RAM during a transmission,
- writes the received data into the RAM during a reception,
- provides status information to the SW like which buffer is currently under used, the number of Data buffers already used, the information the programmed threshold is reached, etc.

The DBM embeds a 16-byte internal FIFO to be able to absorb potential latencies on the AHB traffic without losing bytes. This internal FIFO leads to some specific behaviors described below.

In transmission, the DBM prefetches data in RAM. The first prefetch occurs on TX command request (COMMAND register writing) and its size (up to 16 bytes) depends on selected transmission mode and packet length information, as shown in *Table 140*.

Table 140. DBM initial internal FIFO prefetch

Transmission mode (PCKT_CTRL[8:7] = TX_MODE[1:0])	Packet length PCKLEN_CONFIG[15:0]	Prefetch behavior
Normal mode	0	No prefetch as no payload
	< 16	prefetch PCKLEN[15:0] bytes
	≥ 16	prefetch 16 bytes
Direct through buffers	0	prefetch 16 bytes (as infinite TX up to SABORT)
	< 16	prefetch PCKLEN[15:0] bytes
	≥ 16	prefetch 16 bytes
Others (PN and Direct through GPIOs)	Don't care	No prefetch

Then a new word (4 bytes) is prefetch each time a slot is free in the internal FIFO until the number of bytes defined by the packet length is reached (except for infinite direct through buffers configuration).

29.15.3 Programming procedure

The SW must program the following information before to use the Data Buffers:

- Data Buffers size (from 0 to 65535 bytes) in the *DATABUFFER_SIZE register (DATABUFFER_SIZE)*
- Data buffers location in the *DATABUFFER0_PTR register (DATABUFFER0_PTR)* and the *DATABUFFER1_PTR register (DATABUFFER1_PTR)*.

Note: The base address of the Data Buffer in RAM must be 32-bit aligned.

Some status bit fields, sometimes associated to an interrupt (if enabled), provide information to the SW about the on-going usage of the Data Buffers:

- DATABUFFER0_USED_F and DATABUFFER1_USED_F:
  - located in the *RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)*
  - associated to an interrupt (if enabled)
  - set each time the DBM has completed the read in TX or the write in RX inside the Data buffer and started reading the other DATABUFFER
  - need to be cleared by SW writing a 1 in the associated bit.
- CURRENT_DATABUFFER:
  - located in the *DATABUFFER_INFO register (DATABUFFER_INFO)* (status register)
  - indicates which Data Buffer is currently under read or write by the DBM.
- NB_DATABUFFER_USED[14:0]:
  - located in the *DATABUFFER_INFO register (DATABUFFER_INFO)* (status register)
  - indicates the number of Data Buffers already fully used since the beginning of the transfer (on-going or latest if the transfer is over).
- CURRENT_DATABUFFER_COUNT[15:0]:
  - located in the DATABUFFER_INFO status register,
  - indicates the number of bytes used in the last Data Buffer used (could be 0 if the transfer ended with a number of bytes that is modulo the Data Buffers size).

At the end of the transfer, the SW can determine the number of bytes read or stored in the Data buffers following the formula:

(NB_DATABUFFER_USED[14:0] x DATABUFFER_SIZE[15:0]) + CURRENT_DATABUFFER_COUNT[15:0]

In addition, the SW can define some thresholds (common to both Data Buffers):

- RX_ALMOST_FULL_THR[15:0]
  - almost full threshold in bytes for the Data buffers on a reception,
  - located in the *DATABUFFER_THR register (DATABUFFER_THR)*
  - associated to an interrupt if enabled,
  - set when the CURRENT_DATABUFFER_COUNT[15:0] = RX_ALMOST_FULL_THR[15:0]
- TX_ALMOST_EMPTY_THR[15:0]:
  - almost empty threshold in bytes for the Data Buffers in transmission,
  - located in the DATABUFFER_THR register,
  - associated to an interrupt if enabled,
  - set when the CURRENT_DATABUFFER_COUNT[15:0] = TX_ALMOST_EMPTY_THR[15:0]

Warning: Due to the DBM 32-bit accesses only in RAM, if defined less than DATABUFFER_SIZE – 4, the threshold flags may happen simultaneously to the DATABUFFER_USEDx_F flag.

29.15.4 Error handling

The DBM manages a DBM FIFO error.

This error flag can be raised for two different reasons:

1. the Data buffers size has been programmed to 0 while:
   - Transmission mode is Normal mode (TX_MODE[1:0]=”00”) and packet length is not null
   - or Transmission mode is direct through Buffer (TX_MODE[1:0] = “01”)
2. the internal FIFO encounters an overflow (in reception) or an underflow (in transmission).

Regarding the second case: the DBM embeds a 16-bytes internal FIFO to store locally the received data or the next data to transmit in case of high latency on the AHB accesses in the device. The sizing of the FIFO has been studied to fit with the targeted device structure.

However, if the latency generated by the SoC is very too long, it may happen that:

- in reception: the FIFO is full when the RX_Top block provides a new byte to store (overflow)
- in transmission: the FIFO is empty when the TX_Top block requests a new byte to transmit (underflow).

In all cases, if the error is detected, the DBM_FIFO_ERROR_F bit is set in the RFSEQ_STATUS_DETAIL register.

When this bit is set, it also sets the AHB_ACCESS_ERROR_F bit in the RFSEQ_IRQ_STATUS register, which can be associated to an interrupt.

29.16 Sequencer

The Sequencer allows autonomous RF transfers management / chaining without CPU interfering.

Some example of scenarios are proposed in *Section 29.23: Uses-cases: examples of scenarios with the Sequencer*.

29.16.1 Sequencer overview

The role of the Sequencer is to run scenarios prepared by the CPU to achieve some application use-cases like Listen Before Talk, Auto Acknowledge, Auto-Retransmission, and so on.

To build and execute a scenario, the SW must:

1. Prepare in RAM a chained list of actions (called SeqActions in this document) through the GlobalConfiguration and ActionConfiguration RAM tables (see *Section 29.16.4: Sequencer RAM tables* for details).
2. Provide the location of the GlobalConfiguration as an offset of the device RAM base address through the retained *SEQ_GLOBALTABLE_PTR register (SEQ_GLOBALTABLE_PTR)*.
3. From there, the SW can decide when to start the execution of the sequence by writing 1 in the action bit GEN_SEQ_TRIGGER located in the Misc *SEQUENCER_CTRL register (SEQUENCER_CTRL)*.
4. Once the initiator trigger is sent, the Sequencer manages the scenario through the execution of the chained action tables. The sequence mainly consists in:
   – load the static registers with values in the GlobalConfiguration RAM table if requested (automatic after a reset/low power sequence and on SW request through a ForceReload bit in RAM tables)
   – Read the ActionConfiguration table pointed by the GlobalConfiguration table and execute the scenario:

     Load the dynamic registers with ActionConfiguration table content

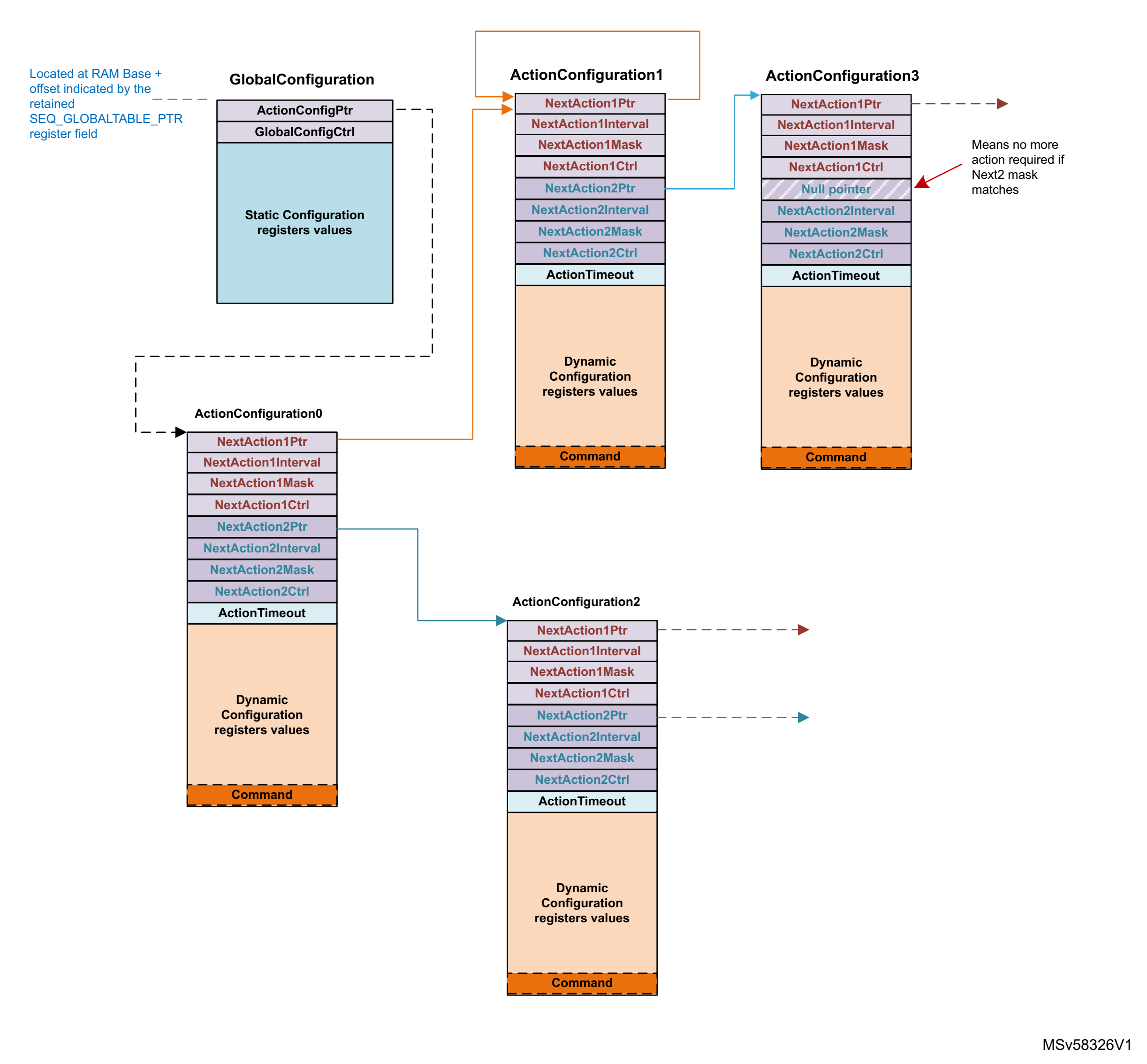
     Monitor the events versus two enable masks

     When an event enabled by one of the two masks is active, get information about the next SeqAction (new ActionConfiguration table pointer, time interval before execution, low power mode at SoC level allowed or not during this inter-SeqAction interval

The SW may program some interrupts to be informed on specific steps along the sequence (RX done implying some received data to process, TX done that may imply a low power phase for the device, and so on.)

Figure 292 is a generic representation of RAM tables used in chained sequences.

Figure 292. Sequencer actions list example



29.16.2 Sequence description

In this document, any phase from a trigger event on Sequencer to the end of the selected ActionConfiguration RAM table is called a SeqAction. An application use-case is an assembly of SeqActions through RAM tables configuration and mask definition for future actions.

The Sequencer starts its activity only on a trigger event which could be:

- an APB write action in the GEN_SEQ_TRIGGER bit (only option to start a scenario as only the Sequencer can modify the *RFIP_WAKEUPTIME register (RFIP_WAKEUPTIME)* register and RFIP_WAKEUP_EN bit)
- a RFIP wakeup timer event
- an internal trigger generated by the Sequencer itself if the next action is requested to be started immediately after the on-going one.

On this trigger event, the Sequencer:

- gets information from the GlobalConfiguration RAM table:
  - if requested (ForceClearEvents bit is set), resets the internal sequencer event status information,
  - if requested (after a reset or if ForceReload bit is set), loads the Static configuration registers with the values provided at the bottom of the table,
  - gets full address pointer on the ActionConfiguration entity to execute
- then it gets the sequence information in the pointed ActionConfiguration RAM table containing the following information:
  - Two different next actions are possible depending on the result / status generated by the on-going action. knowing each of the two next actions description is defined symmetrically by:

    An associated ActionConfiguration address pointer,

    A mask based on seq_event_status[31:0] internal bus (respecting *RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)* bit mapping) to consider the next action is the selected one.

    A delay to introduce before to execute this next action (done through an HW programming of the RFIP wakeup timer by the Sequencer),

    The ForceClearEvents, ForceReload and SleepEn bits associated to the selected next action configuration are written back in the common GlobalConfiguration RAM table
  - The Dynamic global registers are overwritten by the Sequencer using the values provided in the ActionConfiguration table

Note: The latest Dynamic register is the COMMAND register: so as soon as the Sequencer overloads this register, the associated command is launched by the Radio Controller (except if NOP opcode is used)

  – a timeout (ActionTimeout) used as a watchdog to limit the waiting window until a match occurs for one or the other next action is managed by the Sequencer; this timeout monitoring is started just after the overload of the Dynamic registers.

- Once a Next action mask match occurs:
  – The Sequencer updates few bit fields of the GlobalConfiguration (pointer on ActionConfiguration table, ForceClearEvents, ForceReload and SleepEn bits)
  – The Sequencer gets the next pointer address from the selected next action and writes it back in GlobalConfiguration

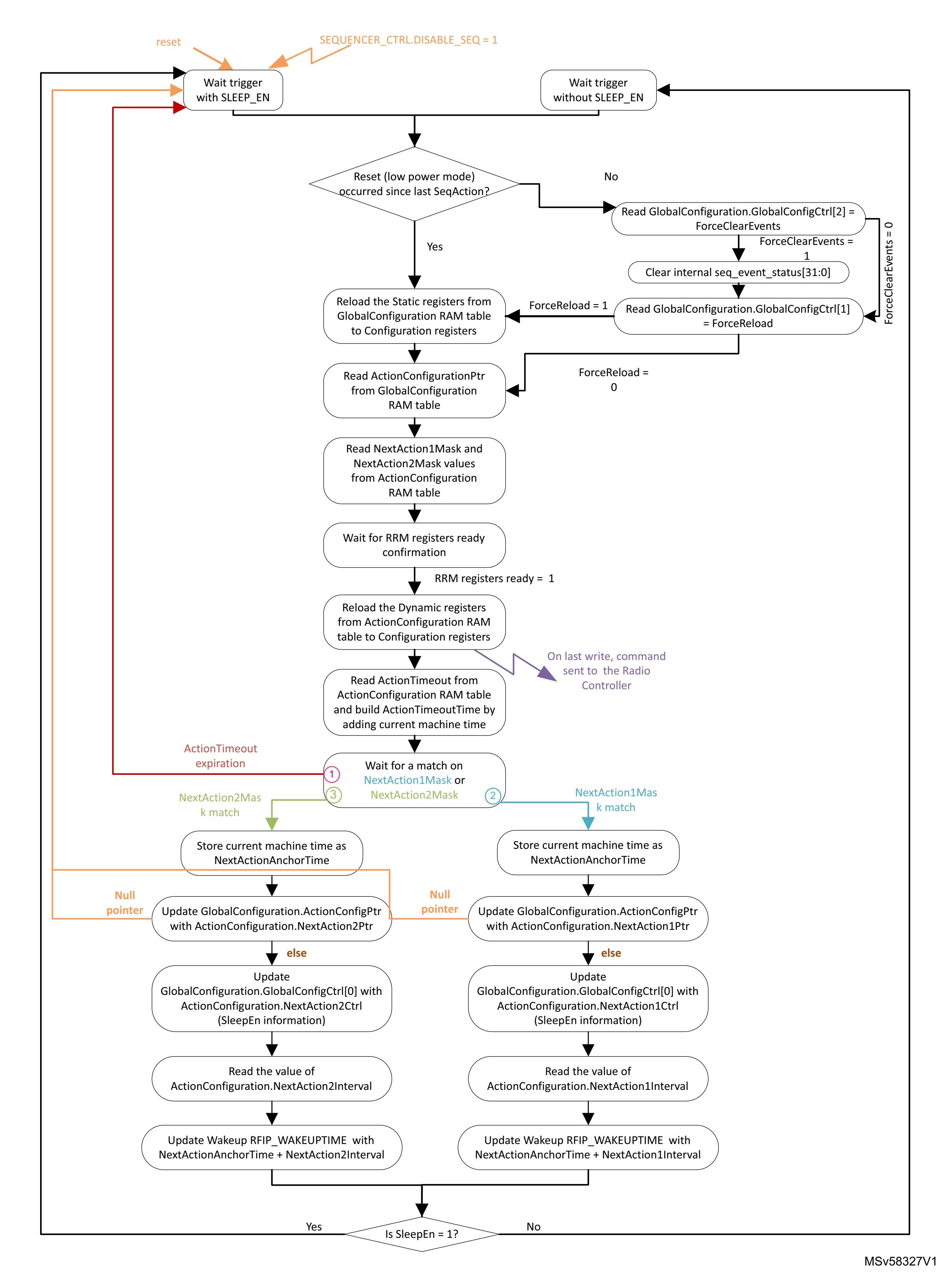
Note: A null pointer for next action means the sequence is complete (no more chained SeqAction requested). Then the Sequencer returns to the wait trigger state with low power mode allowed (SleepEn=1). Only a manual trigger through the GEN_SEQ_TRIGGER bit restarts a sequence as the Wakeup timer has not been reprogrammed in this case by the Sequencer.

- The Sequencer checks the delay for next action trigger:
  – if a null interval is requested, the Sequencer generates an internal trigger to restart immediately from the first step (“gets information from the GlobalConfiguration table”)
  – If a non-null interval is requested, the Sequencer reprograms the Wakeup timer for a future trigger event and goes in a “wait for trigger” state. The Sequencer allows the low power mode of the device during this interval or not depending on the NextActionXCtrl.SleepEn bit value (X=1 or 2 depending on matched mask).
- If the ActionTimeout expires before any Next action mask match occurs, then the Sequencer stops without reprogramming any future trigger and goes in “wait for trigger with sleep enable” state (low power mode allowed).

When the Sequencer completes a SeqAction due to an ActionTimeout event or due to a match on a NextAction pointing on a null address pointer, the SEQ_F status flag (associated to an interrupt if enabled) is raised in RFSEQ_IRQ_STATUS register. A complementary information is available in the RFSEQ_STATUS_DETAIL register to indicate if the SeqAction completed on a next action mask match with null pointer (SEQ_COMPLETE_F flag) or on an ActionTimeout event (SEQ_ACTIONTIMEOUT_F flag).

Figure 293 summarizes the flow of execution by the Sequence of a SeqAction.

Figure 293. SeqAction flow at Sequencer level



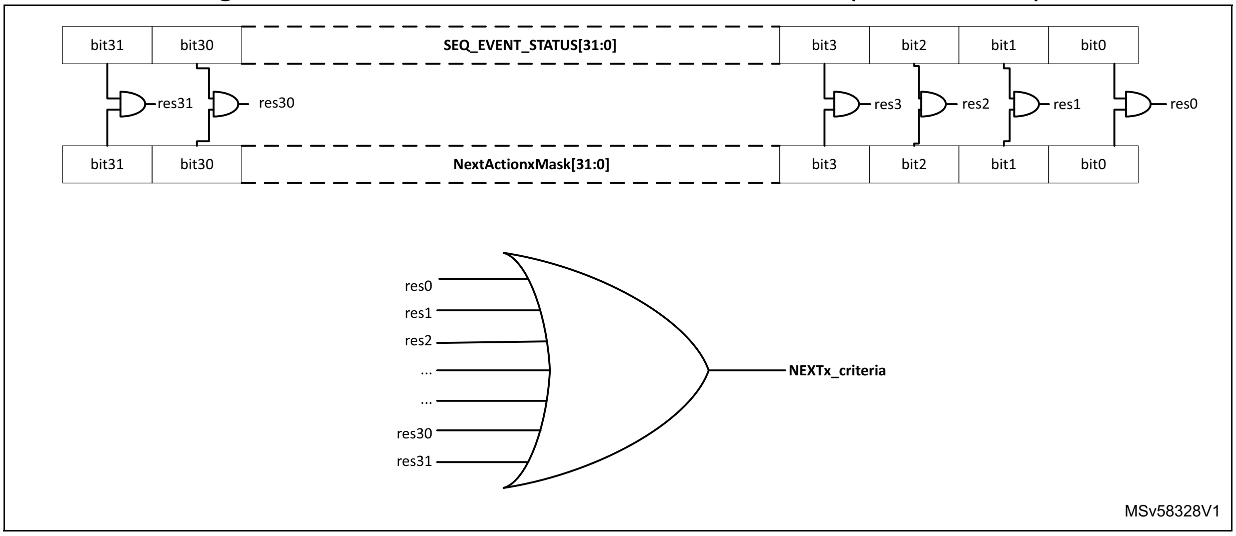
29.16.3 Next action mask management

The sequence is based on chained SeqActions.

Each SeqAction can select the next SeqAction among a list of two depending on the current SeqAction effects / consequences. The criteria to select the next SeqAction is based on a mask called NextActionXMask with X=1 or 2.

This principle of this mask is to be mapped bit by bit on an internal seq_event_status bus using the same bit mapping as the *RFSEQ_IRQ_STATUS register (RFSEQ_IRQ_STATUS)* . A mask match is valid if at least one event enabled by the mask is at 1 in the internal seq_event_status bus.

Figure 294. NextActionxMask mechanism overview (with x = 1 or 2)



A mask match event occurs as soon as:

- a seq_event_status bit is 1
- and its associated bit in NextActionxMask mapping is also 1

Note: A null mask is considered as always match

If a match occurs simultaneously on both NextAction1Mask and NextAction2Mask, the NextAction1Mask is selected

The seq_event_status bus is built inside the Sequencer according to the following rules:

- some events are not applicable
- some events are real-time events: the current state is used so the event can be active at a moment and no more active later without any clear action (for instance the CS flag

indicating if there is a power on the antenna dynamically rises or falls according to current power received by the antenna)

- some events are recorded and stay high even if the conditions have changed. In this case, the event bit is cleared:
  - on a SeqAction exit due to an ActionTimeout event,
  - on a SeqAction entry if the ForceClearEvents bit is set in the Control word of the GlobalConfiguration RAM table,
  - on a mask match event only if the associated bit in the matching mask (NextAction1Mask or NextAction2Mask) is set.

Note: The event recording is always active, except when the Sequencer is disabled or in "wait trigger with sleep enable" state.

The table below summarizes the bit mapping of the seq_event_status bus and the treatment associated to each bit.

Table 141. Sequencer internal events bus overview

Bit position	Name / Feature	Treatment
31	AGC_AAF calibration done	Recorded
30	SAFE-ASK Calibration done	Recorded
29	Reserved, no event associated	Not applicable
28	RRM command end	Recorded
27	RRM command started	Recorded
26	Reserved, no event associated	Not applicable
25	Reserved, no event associated	Not applicable
24	HW Analog failure	Recorded
23	Reserved, no event associated	Not applicable
22	AHB Access Error (DATABUFFER_SIZE=0 or DBM internal FIFO overflow/underflow)	Real-time event
21	TX almost empty DATABUFFER1	Real-time event
20	TX almost empty DATABUFFER0	Real-time event
19	RX almost full DATABUFFER1	Real-time event
18	RX almost full databuffer0	Real-time event
17	DATABUFFER1 used	Real-time event
16	DATABUFFER0 used	Real-time event
15	Reserved, no event associated	Real-time event
14	Valid SYNC word detected	Real-time event
13	Preamble detected	Real-time event
12	Carrier Sense detected (power on the antenna over programmed RSSI threshold)	Real-time event
11	Reserved, no event associated	Not applicable
10	Reserved, no event associated	Not applicable

Table 141. Sequencer internal events bus overview

Bit position	Name / Feature	Treatment
9	Command rejected	Recorded
8	SABORT done	Recorded
7	RX Timer Stop conditions match	Recorded
6	Reserved, no event associated	Not applicable
5	Reserved, no event associated	Not applicable
4	Fast RX Termination	Recorded
3	CRC error on reception	Recorded
2	Timeout on reception	Recorded
1	Reception OK	Recorded
0	Transmission done	Recorded

After the last dynamic register overload, once a match occurs, the Sequencer launches the end of SeqAction process (write back in GlobalConfiguration table and potential programming of the RFIP wakeup timer). Then the Sequencer waits for a new trigger to restart a new SeqAction.

29.16.4 Sequencer RAM tables

The RAM tables must be located in the RAM bank(s) that stay in retention during SoC low power mode sessions to be able to run scenarios chaining SeqActions with interleaved low power sequences.

The RAM tables used by the Sequencer are split in two categories:

- GlobalConfiguration table:
  - this RAM table is unique
  - this RAM table is always used by the Sequencer at the beginning of each SeqAction (on trigger event)
  - the main role of this table is to point on the ActionConfiguration RAM table concerned by the trigger event and to optionally preset some specific elements
- ActionConfiguration tables:
  - there are several tables
  - each table is linked to a specific SeqAction
  - the main role of these tables is to configure and launch a command and to define the possible following SeqAction depending on the consequences (flags) of the command

Tips for users

This paragraph proposes some recommendations for few fields of the RAM tables.

- GlobalConfiguration initilization for a new SeqActions chain:
  - Set ForceClearEvents and ForceReloadStatic bits to 1 to remove any possible history coming from potential previous sequence (chained SeqAction)
  - Then those 2 bits are written back at the end of each SeqAction according to NextActionCtrl bit fields defined in the matched next action
- Use 0x04000000 value as a “never match” mask
  - The bit 26 of the seq_event_status bus is stuck to 0 so this match cannot happen
  - Note that the bit 26 of the RFSEQ_STATUS_IRQ is the SEQ_F flag so this bit is not expected to become valid/used in the seq_event_status bus in future evolutions.
- Use 0x00000000 as an “always match” mask
  - the Sequencer automatically matches a null mask

29.17 GlobalConfiguration RAM table overview

The GlobalConfiguration RAM table is always read by the Sequencer after any trigger event whatever the on-going sequence (just starting or already few intermediary action steps executed). The GlobalConfiguration table is unique and shared by all the SeqActions.

The GlobalConfiguration RAM table could be summarized as follows:

- Start address of this table is defined in the SEQ_GLOBALTABLE_PTR global Retained register as an offset of the SoC RAM base address
- Start address of this table must be 32-bit aligned in RAM
- Unique and shared by all the SeqActions
- Always read first after each trigger event
- It contains:
  - In WORD 0: the pointer on the ActionConfiguration RAM table to execute,
  - In WORD1:

    A bit (ForceClearEvents) to clear the internal seq_event_status bus before to start the SeqAction

    A bit (ForceReload) to enable/disable a force reload of the Static registers even if no reset occurs since the last reload

    A bit (SleepEn) indicating if the low power mode was allowed during the wait interval since last SeqAction execution (result of the previous SeqAction write back operation)
  - A Static register table to be used to override all the Static configuration registers

At the end of the SeqAction execution, the Sequencer updates the 2 first words of the GlobalConfiguration table according to the next action address pointer and information present in the WORD1 of the under execution ActionConfiguration RAM table which are:

- Address of the ActionConfiguration to be executed on next trigger event
- Allowance or not of low power mode until next action trigger event
- Enable/disable the seq_event_status bus clearance on entry of next SeqAction
- Enable/disable a force of the static register reload at the beginning of the next action

Table 142. GlobalConfiguration RAM table

Word	Byte addr	R/W by seq	7	6	5	4	3	2	2	1	0
0x00	0x0	R/W	ActionConfigPtr[7:0]								
	0x1		ActionConfigPtr[15:8]								
	0x2		ActionConfigPtr[23:16]								
	0x3		ActionConfigPtr[31:24]								
0x01	0x4	R/W	-						Force clear events	Force reload	Sleep enable
	0x5		-								
	0x6		-								
	0x7		-								
0x02	0x8	R	StaticConfigReg0[7:0]								
	0x9		StaticConfigReg0[15:8]								
	0xA		StaticConfigReg0[23:16]								
	0xB		StaticConfigReg0[31:24]								
0x03	0xC	R	StaticConfigReg1[7:0]								
	0xD		StaticConfigReg1[15:8]								
	0xE		StaticConfigReg1[23:16]								
	0xF		StaticConfigReg1[31:24]								
...	0x10	R	...								
	0x11		...								
	0x12		...								
	0x13		...								
0x10	0x40	R	StaticConfigRegE[7:0]								
	0x41		StaticConfigRegE[15:8]								
	0x42		StaticConfigRegE[23:16]								
	0x43		StaticConfigRegE[31:24]								

29.18 GlobalConfiguration RAM table register descriptions

Table 143. GlobalConfiguration RAM table registers

Address Offset	Register name	Description
0x00	WORD0	Word0 register (ActionConfigPtr)
0x04	WORD1	Word1 register (GlobalConfigCtrl)
0x08 to 0x40	WORDN	WordN register with N = 2 to 0x10

29.18.1 Word 0 register (WORD0)

Address offset: 0x0000

Reset value: 0x0000 0000

Word 1 register

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

Bits 31:0 **ACTIONCONFIGPTR[31:0]**: Action Configuration address Pointer.

Contains the address of the ActionConfiguration table to execute for this trigger event.

The address pointed by this word must be 32-bit aligned.

Note: this value must be initialized by the SW, then it is overloaded automatically by the Sequencer to prepare the next SeqAction depending on the Next1Mask / Next2Mask result.

29.18.2 Word1 register (WORD1)

Address offset: 0x0004

Reset value: 0x0000 0000

Word1 register[a]

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

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

Bit 2 **FORCECLEAREVENTS**: Clear seq_event_status internal bus request.

Defines if the Sequencer must clear its internal seq_events_statusbus on SeqAction entry (before to execute the ActionConfiguration table):

0: do not modify internal seq_event_status bus (keep memory about any potential events not cleared on previous SeqAction exit or that occurred since the last SeqAction exit)

1: clear the seq_event_status bus (erase memory of past events before this SeqAction)

Bit 1 **FORCERELOAD**: Static register force reload request.

Defines in which cases the Sequencer reloads the Static configuration registers:

0: limits the reload of the Static configuration registers to power off detection case

1: notifies the Sequencer to reload the Static configuration in any cases.

Note: the Sequencer automatically reloads the static registers if it detects the MR_SubG IP has been power off since last trigger event / SeqAction execution.

Bit 0 **SLEEPEN**: Sleep allowed after SeqAction execution. Indicates if a low-power sequence is allowed, until the next SeqAction triggered event.

0: the Sequencer does not allow the low power mode between the previous SeqAction and the next one

1: the Sequencer allows the device going in low power mode until the next SeqAction trigger occurs.

a. This word is written back by the Sequencer in RAM at the end of each SeqAction execution to prepare the Sequencer behavior up to next trigger event.

29.18.3 Word N register N = 2 to 16 (WORDN)

Address offset: 0x08 + (4 x X) with X=0 to 14)

Reset value: 0x0000 0000

Word N register

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

Bits 31:0 **STATIC_CONFIG_REGX**: Static configuration register with X from 0 to 14.

Contains the value to be written by the Sequencer into the Static register X.

29.19 ActionConfiguration RAM table overview

The ActionConfiguration RAM table contains the dynamic part of the SeqAction and is potentially different for each step of the sequence. There can be as many ActionConfiguration tables as needed by the application use-case.

The ActionConfiguration RAM table is summarized as follows:

- start address of the table is defined in the WORD0 of the GlobalConfiguration table for the first SeqAction; then the Sequencer uses the address pointer provided by the ActionConfiguration tables to reprogram the GlobaConfiguration WORD0 at the end of each SeqAction
- start address of these tables must be 32-bit aligned
- as many tables as needed for the full scenario can be defined
- it contains:
  – in WORD 0 to WORD3: the information linked to Next Action1 (pointer address of the corresponding ActionConfiguration RAM table, interval to wait before to run it, mask to match the Next Action1 and Next Action1 control
  – in WORD 4 to WORD7: the equivalent information linked to Next Action2 (pointer address of the corresponding ActionConfiguration RAM table, interval to wait before to run it, mask to match the Next Action2 and Next Action2 control
  – in WORD8: the ActionTimeout duration
  – the rest of the table (WORD9 to WORD13) contains a Dynamic configuration registers table to reprogram the Dynamic register which are intended to change between the sequenced actions.

Table 144. ActionConfiguration RAM table

Word	Byte addr	R/W by Seq	Bitfield (8 bits)
0x00	0x0	R	NextAction1Ptr[7:0]
	0x1		NextAction1Ptr [15:8]
	0x2		NextAction1Ptr [23:16]
	0x3		NextAction1Ptr [31:24]
0x01	0x4	R	NextAction1Interval[7:0]
	0x5		NextAction1Interval[15:8]
	0x6		NextAction1Interval[23:16]
	0x7		NextAction1Interval[31:24]
0x02	0x8	R	NextAction1Mask[7:0]
	0x9		NextAction1Mask[15:8]
	0xA		NextAction1Mask[23:16]
	0xB		NextAction1Mask[31:24]

Table 144. ActionConfiguration RAM table

Word	Byte addr	R/W by Seq	Bitfield (8 bits)
0x03	0xC	R	NextAction1Ctrl[7:0]
	0xD		NextAction1Ctrl[15:8]
	0xE		NextAction1Ctrl[23:16]
	0xF		NextAction1Ctrl[31:24]
0x04	0x10	R	NextAction2Ptr[7:0]
	0x11		NextAction2Ptr [15:8]
	0x12		NextAction2Ptr [23:16]
	0x13		NextAction2Ptr [31:24]
0x05	0x14	R	NextAction2Interval[7:0]
	0x15		NextAction2Interval[15:8]
	0x16		NextAction2Interval[23:16]
	0x17		NextAction2Interval[31:24]
0x06	0x18	R	NextAction2Mask[7:0]
	0x19		NextAction2Mask[15:8]
	0x1A		NextAction2Mask[23:16]
	0x1B		NextAction2Mask[31:24]
0x07	0x1C	R	NextAction2Ctrl[7:0]
	0x1D		NextAction2Ctrl[15:8]
	0x1E		NextAction2Ctrl[23:16]
	0x1F		NextAction2Ctrl[31:24]
0x08	0x20	R	ActionTimeout[7:0]
	0x21		ActionTimeout [15:8]
	0x22		ActionTimeout [23:16]
	0x23		ActionTimeout [31:24]
0x09	0x24	R	DynamicConfigReg0[7:0]
	0x25		DynamicConfigReg0[15:8]
	0x26		DynamicConfigReg0[23:16]
	0x27		DynamicConfigReg0[31:24]
0x0A	0x28	R	DynamicConfigReg1[7:0]
	0x29		DynamicConfigReg1[15:8]
	0x2A		DynamicConfigReg1[23:16]
	0x2B		DynamicConfigReg1[31:24]

Table 144. ActionConfiguration RAM table

Word	Byte addr	R/W by Seq	Bitfield (8 bits)
...	0x..	R	...
	0x..		...
	0x..		...
	0x..		...
0x13	0x4C	R	DynamicConfigRegA[7:0] (= COMAND[7:0])
	0x4D		DynamicConfigRegA[15:8] (= COMAND[15:8])
	0x4E		DynamicConfigRegA[23:16] (= COMAND[23:16])
	0x4F		DynamicConfigRegA[31:24] (= COMAND[31:24])

Table 145. ActionConfiguration RAM table registers list

Address Offset	Register name	Description
0x00	WORD0	Word0 register
0x04	WORD1	Word1 register
0x08	WORD2	Word2 register
0x0C	WORD3	Word3 register
0x10	WORD4	Word4 register
0x14	WORD5	Word5 register
0x18	WORD6	Word6 register
0x1C	WORD7	Word7 register
0x20	WORD8	Word8 register
0x24 to 0x4C	WORDN	WordN registers with N = 0x9...0x13

29.20 ActionConfiguration RAM table register descriptions

29.20.1 Word 0 register (WORD0)

Address offset: 0x0000

Reset value: 0x0000 0000

Word 0 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NEXTACTION1PTR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEXTACTION1PTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **NEXTACTION1PTR[31:0]**: NextAction1 address Pointer.

Contains the address of the ActionConfiguration table to execute on next trigger event in case of NextAction1Mask match. This address must be 32-bit aligned.

Note: this value is written by the Sequencer in the GlobalConfiguration WORD0 at the end of the on-going SeqAction if the NextAction1Mask matches. if null, the Sequencer does not write back any next SeqAction information in the GlobalConfiguration table nor Wakeup Timer new target.

29.20.2 Word 1 register (WORD1)

Address offset: 0x0004

Reset value: 0x0000 0000

Word 1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NEXTACTION1INTERVAL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEXTACTION1INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **NEXTACTION1INTERVAL[31:0]**: NextAction1 Interval duration (in interpolated absolute time ticks unit).

Contains the delay before to execute the NextAction1 in case of NextAction1Mask match.

– Note:
when a NextAction1 mask match event occurs, this value is added to the current 32-bit interpolated time and the result of this addition is written in the Retained RFIP_WAKEUPTIME register (RFIP_WAKEUPTIME) by the Sequencer at the end of the on-going SeqAction,

– if null, the Sequencer starts immediately the next SeqAction after RAM write back phase.
Remark: if NEXTACTION1CTRL.SleepEn=1, the user must not program an interval shorter than SOC_WAKEUP_OFFSET.

29.20.3 Word 2 register (WORD2)

Address offset: 0x0008

Reset value: 0x0000 0000

Word 2 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NEXTACTION1MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEXTACTION1MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **NEXTACTION1MASK[31:0]**: NextAction1 Mask.

Contains the mask to be applied on the seq_event_status internal bus by the Sequencer to detect the on-going SeqAction must be stopped and NextAction1 is selected for the next step.

If at least one bit at 1 in the mask is also equal to 1 in the seq_event_status bus (at same position), then a match occurs.

29.20.4 Word 3 register (WORD3)

Address offset: 0x0004

Reset value: 0x0000 0000

Word3 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NEXT1FORCECLEAREVENTS	NEXT1FORCERELOAD	NEXT1SLEEPEN
													rw	rw	rw

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

Bit 2 **NEXT1FORCECLEAREVENTS**: Clear seq_event_status internal bus request bit to write back in GlobalConfiguration.GlobalConfigurationControl[2] at the end of the on-going SeqAction in the case of NextAction1Mask match.

Setting this bit means the user requests a seq_event_status bus clear on next SeqAction start if the next action mask 1 matches.

Bit 1 **NEXT1FORCERELOAD**: Static register force reload request bit to write back in GlobalConfiguration.

GlobalConfigurationControl[1] at the end of the on-going SeqAction in case of NextAction1Mask match.

Setting this bit means the user requests a Static registers reload on next SeqAction start if the next action mask 1 matches.

Bit 0 **NEXT1SLEEPEN**: Sleep allowed after SeqAction execution bit to write back in GlobalConfiguration.

GlobalConfigurationControl[0] at the end of the on-going SeqAction in case of NextAction1Mask match.

Setting this bit means the user allows a device Low Power sequence between the end of the on-going SeqAction and the next SeqAction start if the next action mask 1 matches.

Note: The whole 32-bit word is written back in the GlobalConfigurationControl word in RAM.

29.20.5 Word 4 register (WORD4)

Address offset: 0x0010

Reset value: 0x0000 0000

Word 4 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NEXTACTION2PTR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEXTACTION2PTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **NEXTACTION2PTR**: NextAction2 address pointer.

Contains the address of the ActionConfiguration table to execute on next trigger event in case of NextAction2Mask match. This address must be 32-bit aligned.

Note:

this value is written by the Sequencer in the GlobalConfiguration WORD0 at the end of the on-going SeqAction if the NextAction2Mask matches.

If null, the Sequencer does not write back any next SeqAction information in the GlobalConfiguration table nor Wakeup Timer new target.

29.20.6 Word 5 register (WORD5)

Address offset: 0x0014

Reset value: 0x0000 0000

Word 5 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NEXTACTION2INTERVAL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEXTACTION2INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **NEXTACTION2INTERVAL[31:0]**: NextAction2 Interval duration (in interpolated absolute time ticks unit).

Contains the delay before to execute the NextAction2 in case of NextAction1Mask match.

Note:

when a NextAction2 mask match event occurs, this value is added to the current 32-bit interpolated time and the result of this addition is written in the Retained RFIP_WAKEUPTIME register (RFIP_WAKEUPTIME) by the Sequencer at the end of the on-going SeqAction,

if null, the Sequencer starts immediately the next SeqAction.

Remark: if NEXTACTION2CTRL.SleepEn=1, the user must not program an interval shorter than SOC_WAKEUP_OFFSET.

29.20.7 Word 6 register (WORD6)

Address offset: 0x0008

Reset value: 0x0000 0000

Word 6 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NEXTACTION2MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NEXTACTION2MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **NEXTACTION2MASK[31:0]**: NextAction2 Mask.

Contains the mask to be applied on the seq_event_status internal bus by the Sequencer to detect the on-going SeqAction must be stopped and NextAction2 is selected for the next step.

If at least one bit at 1 in the mask is also equal to 1 in the seq_event_status bus (at same position), then a match occurs.

29.20.8 Word 7 register (WORD7)

Address offset: 0x001C

Reset value: 0x0000 0000

Word7 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NEXT2FORCECLEAREVENTS	NEXT2FORCERELOAD	NEXT2SLEEPEN
													rw	rw	rw

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

Bit 2 **NEXT2FORCECLEAREVENTS**: Clear seq_event_status internal bus request bit to write back in GlobalConfiguration.

GlobalConfigurationControl[2] at the end of the on-going SeqAction in case of NextAction2Mask match.

Setting this bit means the user requests a seq_event_status bus clear on next SeqAction start if the next action mask 2 matches.

Bit 1 **NEXT2FORCERELOAD**: Static register force reload request bit to write back in GlobalConfiguration.

GlobalConfigurationControl[1] at the end of the on-going SeqAction in case of NextAction1Mask match.

Setting this bit means the user requests a Static registers reload on next SeqAction start if the next action mask 2 matches.

Bit 0 **NEXT2SLEEPEN**: Sleep allowed after SeqAction execution bit to write back in GlobalConfiguration.

GlobalConfigurationControl[0] at the end of the on-going SeqAction in case of NextAction1Mask match.

Setting this bit means the user allows a device Low Power sequence between the end of the on-going SeqAction and the next SeqAction start if the next action mask 2 matches.

Note: The whole 32-bit word is written back in the GlobalConfigurationControl word in RAM.

29.20.9 Word 8 register (WORD8)

Address offset: 0x0020

Reset value: 0x0000 0000

Word 8 register

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

Bits 31:0 **ACTIONTIMEOUT[31:0]**: Action timeout (in interpolated absolute time ticks).

Contains the maximum duration to wait between the end of Dynamic registers reload action and a match on Next1 or Next2 action. If the timeout expires, the Sequencer stops monitoring the NextAction1Mask and NextAction2Mask and returns in the “wait with sleep enable“ state.

If the timeout event occurs, the SEQ_ACTIONTIMEOUT_F flag in the RFSEQ_STATUS_DETAIL register and the SEQ_F flag in the RFSEQ_IRQ_STATUS are set.

Note: only a manual trigger (APB write) can then restart as no RFIP_WAKEUPTIME reprogramming done by the Sequencer in this case).

Note: null value means no timeout on next actions masks monitoring.

29.20.10 Word N register N = 2 to 16 (WORDN)

Address offset: 0x24 + (4 x X) with X=0 to 10)

Reset value: 0x0000 0000

Word N register

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

Bits 31:0 **DYNAMIC_CONFIG_REGX[31:0]**: Dynamic configuration register with X from 0 to 10.

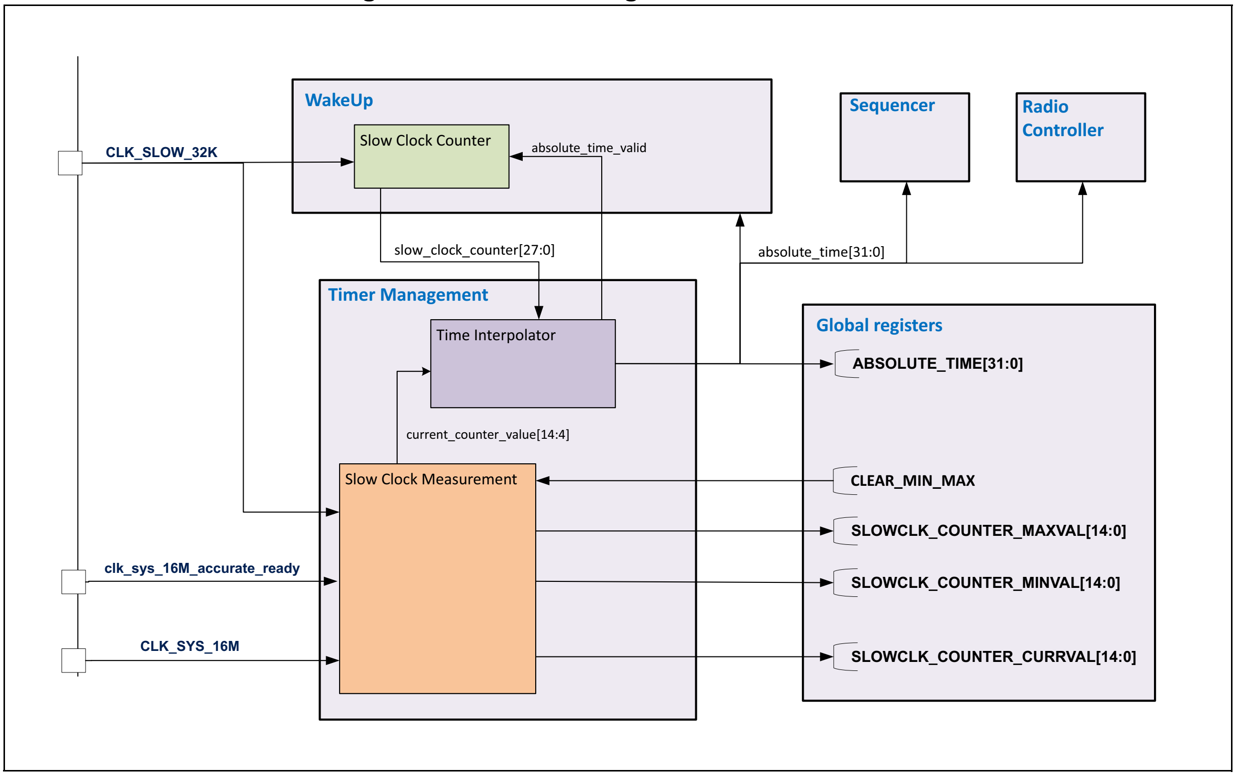
Contains the value to be written by the Sequencer into the Dynamic register X.

29.21 Time management unit (TMU)

The time Management unit oversees two main features:

- compute a 32-bit interpolated absolute time,
- handle the slow clock period measurement.

Figure 295. Time Management Unit overview



29.21.1 Slow clock period measurement

the slow clock internal oscillator (LSI) of the device is not necessarily at 32 kHz. According to the LSI specification, it could provide a frequency between 24 kHz and 49 kHz. For this reason, the SW needs to have a mean to measure on each device the real slow clock period.

The MR_SubG IP embeds a slow clock period measurement block to help the SW calculating this information.

The slow clock measurement principle is to count the number of fast clock periods inside a window of 32 slow clock periods.

The main points to consider are:

- The fast clock frequency is XO frequency / 3 so typically 16 MHz. But it can also be 15.666 MHz with a XO at 47 MHz or 16.666 MHz with a XO at 50 MHz.
- The slow clock measurement is always running and refresh every 32 slow clock cycles
  - the measurement block counts only if the device indicates the fast clock tree is accurate. In *the STM32WL30xx/31xx/33xx*, it corresponds to have a clock source

provided directly by the XO (HSE mode) or by the RC64MPLL block locked on the XO (HSIPLL mode)

- The slow clock counter stops and is reset back to 0 each time the accurate information is cleared by the clock controller of the device.
- The slow clock measurement block also provides some statistics by recording the minimum and maximum period seen in a slow clock cycle. This may show a drift in time for instance. Those statistics are automatically cleared when the device resets the accurate clock information and can also be cleared by the SW.

The results of the calculations are available in the some Global Misc registers:

- SCM_COUNTER_CURRVAL[14:0]:
  - located in the *SCM_COUNTER_VAL register (SCM_COUNTER_VAL)*
  - provides the number of 16 MHz fast clock cycles seen in a 32 slow clock period window
  - read to 0 after a POR or after the loss of the system clock accuracy information; then kept to 0 until the first valid calculation is available (32 slow clock cycles after system clock accurate information).
- SCM_COUNTER_MINVAL[14:0]:
  - located in the *SCM_MIN_MAX register (SCM_MIN_MAX)*
  - provides the minimum SCM counter value seen since the counter is ON or since the last SW clearing action
  - the value can be cleared by setting the CLEAR_MIN_MAX action bit (located in the SCM_MIN_MAX register).

Note: This bit clears both min and max value (statistics part) but does not impact the current counter value.

  - read to 0x7FFF after POR or loss of system clock accuracy or SW clearing action.
- SCM_COUNTER_MAXVAL[14:0]:
  - located in the *SCM_MIN_MAX register (SCM_MIN_MAX)*
  - provides the maximum SCM counter value seen since the counter is ON or since the last SW clearing action
  - the value can be cleared by setting the CLEAR_MIN_MAX action bit (located in the SCM_MIN_MAX register).

Note: This bit clears both min and max value (statistics part) but does not impact the current counter value.

  - read to 0x0000 after POR or loss of system clock accuracy or SW clearing action.

The slow clock period is:

$$\mathrm{Period}_{\mathrm{SLOW\_CLOCK}} = \frac{\mathrm{SCM\_COUNTER\_CURRVAL} \times \mathrm{Period}_{f_{\mathrm{SVS}}}}{32}$$

29.21.2 Time interpolation

The Time Management unit block also generates an interpolated absolute time to the other blocks that could need this information. This interpolated frequency is 16 x slow clock frequency.

The interpolated absolute time is a 32-bit value built by the concatenation of:

- the 28-bit absolute time based on a slow clock counter (incremented on each rising edge of the slow clock),
- the 4-bit interpolated value based on an interpolation counter running on the MR_SubG IP system clock.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Slow clock counter part																												Interpolated counter part			

As for the slow clock measurement and associated statistics, **the interpolated time is only running / available when the device clock controller confirms the system clock is accurate**. The interpolated time is available 3 slow clock periods after the combination slow clock present and system clock accuracy confirmation (against 32 slow clock periods for measurement and statistics). The reason is the interpolation starts when the slow clock is present even if the system clock is not yet accurate.

Note: When the system clock is accurate, the time interpolation system uses the slow clock period measurement data to better tune the interval between each interpolated tick.

In any case, the rising edge of the slow clock is used to increment the MSB part and re-start the LSB interpolated part. In this process, the TMU guarantees that no missing code can happen during the interpolation:

- if the slow clock is faster than 32 kHz, the 4-bit interpolation reaches 0xF before the next slow clock rising edge:
  - the 0xF is maintained until the slow clock rising edge occurs,
  - MSB part is incremented as soon as the slow clock rising edge is detected,
  - LSB part re-starts from 0x0 as soon as the slow clock rising edge is detected
- if the slow clock is slower than 32 kHz, the 4-bit interpolation has not yet reached 0xF when the next slow clock rising edge occurs:
  - the time interpolator increments the 4-bit counter every 16 MHz system clock until it reaches 0xF
- Once 0xF is reached:
  - MSB part is incremented,
  - LSB part re-starts from 0x0

The MR_SubG IP sub-blocks using the time interpolation are:

- Global registers:
  - to provide the interpolated absolute time in the Misc *ABSOLUTE_TIME register (ABSOLUTE_TIME)*.
- Radio controller:
  - to manage the RX Timeout
- Sequencer:
  - to build the value to program in the Wakeup block for next SeqAction (= current interpolated absolute time + NextActionxInterval) and load it in RFIP_WAKEUPTIME register
  - to manage the ActionTimeout
- Wakeup:
  - to generate the trigger event to the Sequencer (by comparison with the RFIP_WAKEUPTIME register value)
  - to generate the wakeup interrupt event for the CPU (by comparison with the CPU_WAKEUPTIME register value)

29.22 Wakeup and sleep management

The MR_SubG IP manages information towards the SoC concerning sleep allowance and wakeup request.

29.22.1 Sleep management

The sleep management block is a very basic block providing to the SoC a sleep allowance signal. The goal is to inform the power controller of the SoC if the MR_SubG IP agrees to have a low power mode activated on the SoC or, on the contrary, if some on-going or incoming activities prevent to allow a low power phase at SoC level.

The criteria to allow the low power mode at RFIP point of view is when no activity is on-going or expected to start soon. This is summarized by:

- the Radio FSM is in IDLE state,
- the Sequencer is in the “wait trigger with sleep enable” state.

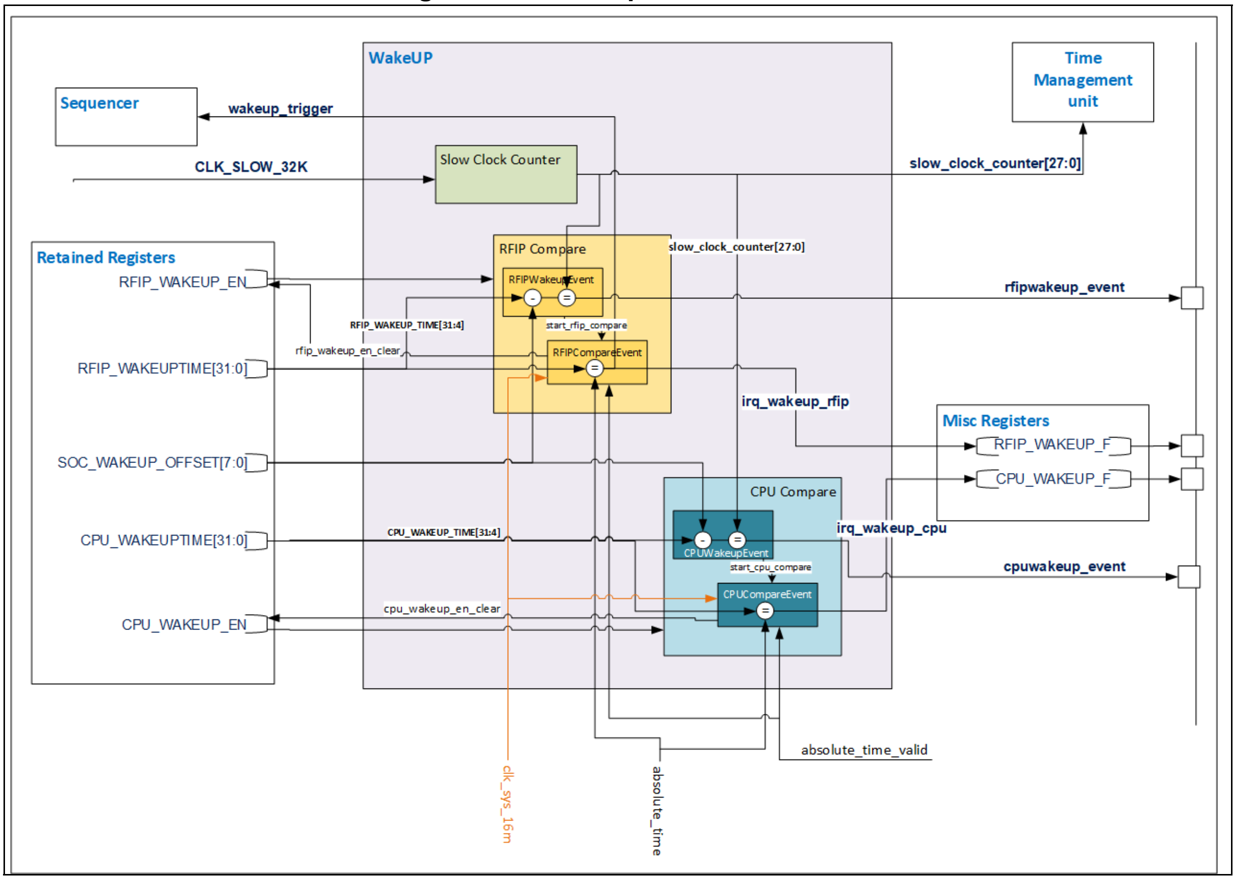
29.22.2 Wakeup block

The Wakeup block of the MR_SubG IP is in the always-on power domain of the device, meaning it stays powered ON during the low power mode phase of the SoC.

The Wakeup block is mainly in charge of:

- the absolute time counter (28-bit),
- the wakeup request for the SoC power controller,
- the wakeup request for the Sequencer and for the CPU.

Figure 296. Wakeup block overview



Slow clock counter management

The Wakeup block embeds a counter clocked by the slow clock, used:

- internally (of the Wakeup block) to generate the SoC wakeup events towards the Power Controller,
- by the Time Management Unit block to manage the ABSOLUTE_TIME[31:4] part.

The slow clock counter:

- (Re)starts from 0 each time a reset occurs on the slow clock tree (corresponding to a POR or an intended SW reset of the MR_SubG IP through the reset manager of the SoC),
- increments on every slow clock rising edge as soon as it is no more under reset,
- rotates back to 0 when it reaches 0xFFFFFFF maximum value.

There is no dedicated action bit / mechanism to reload the counter with 0 or any pre-programmed value.

Wakeup management

The Wakeup block manages two sources of wakeup for the SoC:

- RFIP wakeup: targeting a SoC wakeup event for a SeqAction to be executed by the Sequencer,
- CPU wakeup: possibility to wake up the SoC at a specific absolute time. In this case, the goal is to propose to the SW to share the usage of an always-on timer, already present in the MR_SubG IP and available (if the radio IP is enabled).

The wake-up management is done in two steps:

1. send a wakeup event to the Power controller of the SoC to restart power and clocks,
2. send a wakeup event to the final target (Sequencer or CPU) once the power and clocks are recovered.

The principle is to wake up the SoC before to wake-up the final target to let time to the power, clocks and interpolated absolute time to settle. This is possible thanks to:

- embedded 28-bit slow clock counter providing the absolute time (without LSB interpolation),
- xx_WAKEUPTIME[31:0] registers (xx can be RFIP or CPU),
- SOC_WAKEUP_OFFSET[7:0] bit field in the (Retained) *WAKEUP_CTRL register (WAKEUP_CTRL)*
- xx_WAKEUP_EN bits in the WAKEUP_CTRL retained register (xx can be RFIP or CPU)

The SW has to program a static / fixed value in the SOC_WAKEUP_OFFSET[7:0] bit field. This duration should represent the delay at Soc level from the wakeup event raised on the Power Controller and the moment power, clock and accuracy on system clock are restored.

Then the SW only manages the wanted time for wakeup event on final target (Sequencer or CPU) without taking care of the SoC wakeup sequence delay. The wakeup block uses the SOC_WAKEUP_OFFSET information to anticipate the wakeup on the Power Controller versus the final target time programmed in xx_WAKEUPTIME.

Figure 297 and *Figure 298* show the timeline and contributors versus events.

Figure 297. Sequencer wakeup event generation

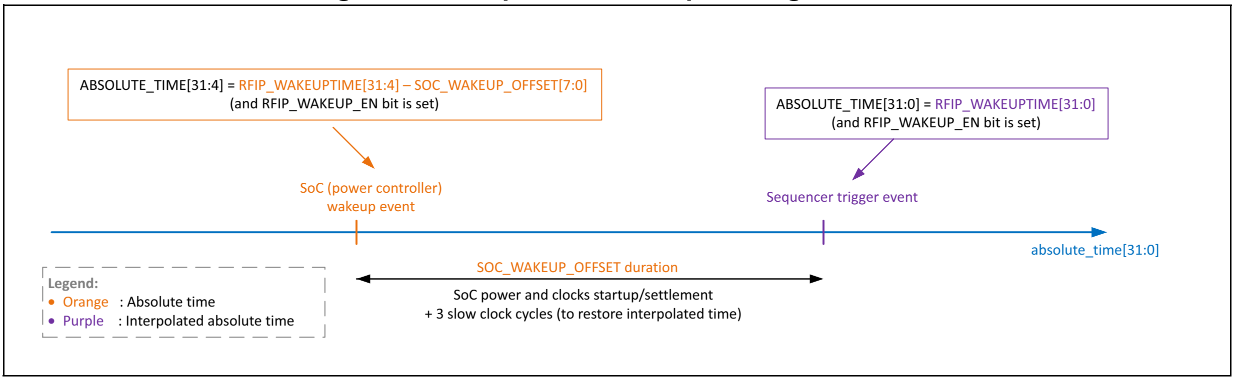
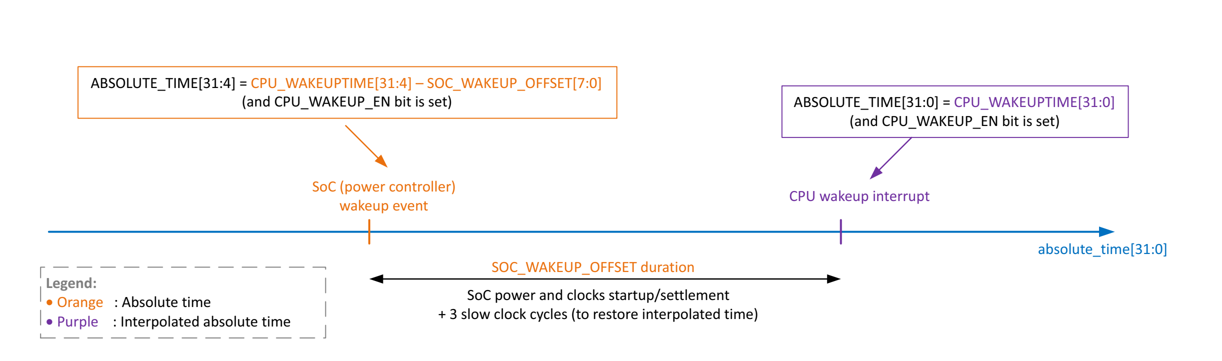


Figure 298. CPU wakeup event generation



The Sequencer wakeup management programming is managed by the Sequencer itself. The SW cannot write directly in the *RFIP_WAKEUPTIME register (RFIP_WAKEUPTIME)*, not in the RFIP_WAKEUP_EN bit in the *WAKEUP_CTRL register (WAKEUP_CTRL)* register. Only the Sequencer can modify those bit fields. The SW must program the wanted behavior and wakeup target time in the GlobalConfiguration and ActionConfiguration RAM tables used by the Sequencer in a manner that low power mode and wakeup events are generated as wanted. Refer to *Section 29.16.2: Sequence description*.

29.23 Uses-cases: examples of scenarios with the Sequencer

This section aims at proposing few scenarios to show how to use the Sequencer in association with the CPU software.

Those scenarios are only example and the user can create the use-cases another way if he prefers and creates other scenarios.

29.23.1 Reception with auto-acknowledge

This use-case is quite simple and is a good start to understand how to chain actions with the help of the Sequencer.

This scenario is the following:

Launch a reception:

- if RX done (Mask1) at the end, launch a TX with the acknowledge information
- if RX Timeout or CRC error (Mask2) at the end, then relaunch a new RX after a defined delay

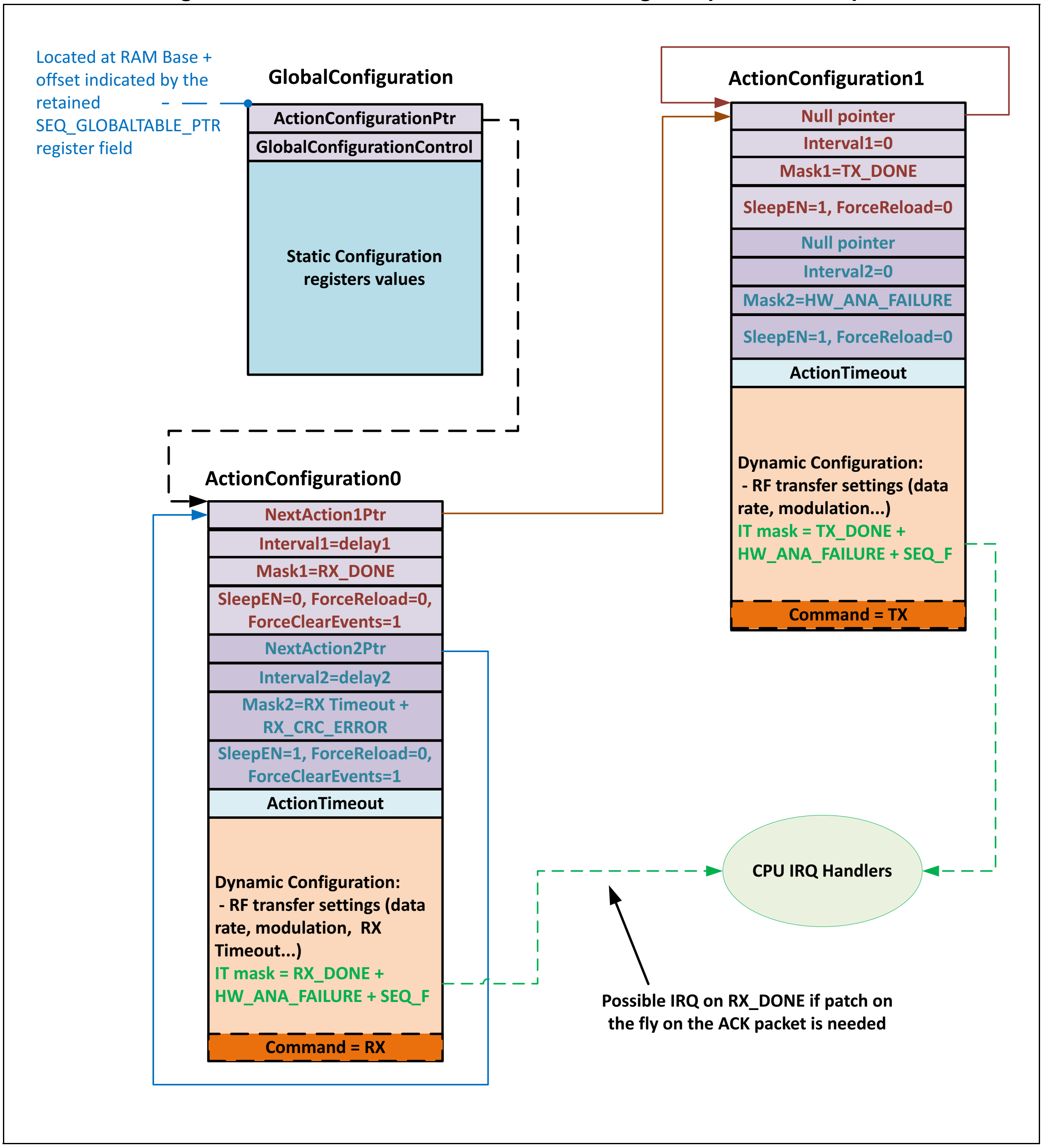
This scenario uses only2 ActionConfiguration RAM tables:

- ActionConfiguration0 table to generate the RX:
  - NextAction1 is a TX to be chained, after a defined delay if RX_DONE flag is raised,
  - NextAction2 is a replay of current ActionConfiguration0 after a defined delay (with allowance of the Low Power mode between each RX phase if the defined delay is long enough to justify it)
  - the CPU could need an IRQ on RX_DONE to modify / prepare the acknowledge content before it is sent
- ActionConfiguration1 table to generate the TX:
  - NextAction1 uses a mask corresponding to TX_DONE and a null pointer as this SeqAction represents the last element of the chained sequence.
  - NextAction2 could use a mask with the HW_ANA_FAILURE flag to monitor a potential RF PLL lock issue that would prevent any TX_DONE flag. This action also defines a null pointer as next action as it represents the last element of the chain.
  - the CPU needs the TX_DONE interrupt to be informed when the RF transmission is over and/or the SEQ_F flag to be informed the sequence is over whatever the cause (TX_DONE, HW_ANA_FAILURE or ActionTimeout)

Note: If the delay between each RX re-trial is not fixed, then:

- *it is necessary to create as many ActionConfiguration tables as targeted RX phases with a different delay in between,*
- *or to use the CPU interrupt on RX Timeout to update the ActionConfiguration0 RAM table with a new interval before the next SeqAction starts.*

Figure 299. RX with Auto-Ack use-case through Sequencer example



29.23.2 Sniff mode

The sniff mode consists in going in RX, check if a frame is currently on the air. If yes, then go on the RX, if not, then go back to a low power mode and retry a bit later.

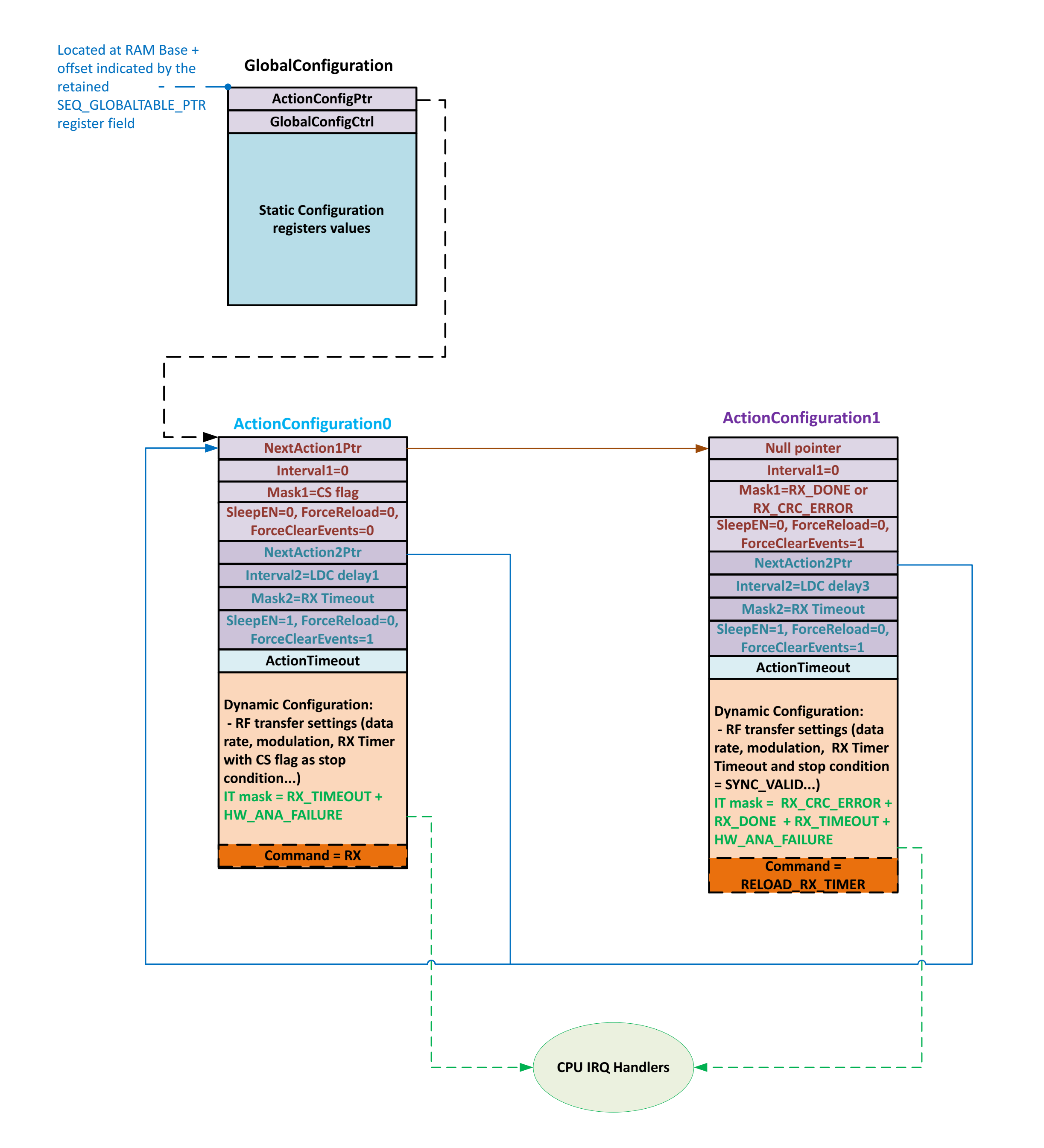
An example of implementation of the sniff mode scenario is proposed below.

Enter reception with RX Timer programmed to stop on CS flag [SeqAction0]:

- If RX timeout occurs, then no power on the antenna, go back to sleep and retry later
- if CS flag detected, reload the RX Timer to stop on valid SYNC word detection [SeqAction1]:
  - if RX Timeout, then go back to sleep and retry later from the beginning
  - if RX done (with or without CRC error), end of the scenario. the SW has to treat the reception result and decide what is the next scenario.

Figure 300 shows the RAM table associated to this scenario.

Figure 300. Sniff mode use-case through Sequencer example



This scenario involves 2 ActionConfiguration tables:

- ActionConfiguration0 table generates the RX command with RX Timer on CS flag (corresponding to RSSI above threshold on the antenna):
  - NextAction1 is selected if CS flag occurs. It targets a Reload RX Timer (while the on-going RX is still active) with a new stop condition based on valid SYNC detection, to be chained immediately (no interval or small delay)
  - NextAction2 is selected if a RX Timeout occurs. It is a replay of current ActionConfiguration0 after a defined delay (with allowance of the Low Power mode between each RX phase if the defined delay is long enough to justify it)
  - the CPU may enable some interrupts like:

    RX_TIMEOUT to be informed a Low Power mode is possible from a radio point of view

    HW_ANA_FAILURE to be informed an issue occurs on analog radio side during RX command start-up and that the scenario may be compromised.
- ActionConfiguration1 table generates a RELOAD_RX_TIMER command with SYNC detection as stop condition:
  - NextAction1 is selected if a RX done occurs (with or without CRC error). It corresponds to end of sequence/no next action (null pointer).
  - NextAction2 is selected if a RX Timeout occurs. It points on ActionConfiguration0 after a defined delay (with allowance of the Low Power mode between each RX phase if the defined delay is long enough to justify it) to restart the sequence from the beginning.
  - the CPU may enable some interrupts like:

    RX_TIMEOUT to be informed a Low Power mode is possible from a radio point of view

    RX_DONE and RX_CRC_ERROR to know the sequence is over on radio side and to treat the reception data.

Note: RX_DONE/RX_CRC_ERROR events are a final step for the scenario.

The NextActionX in the description above takes the color of the pointed ActionConfiguration table.

29.23.3 Listen before talk (LBT)

The Listen Before Talk principle is to check that no other device is currently transmitting on the air before to transmit from our device. This can be summarized by starting the TX scenario by reception phases used to check the power on the antenna and to transmit only once there is no power on the air.

The scenario proposed here consists in checking up to 3 times that no power is on the antenna to do the TX. An example of implementation through the Sequencer is proposed below:

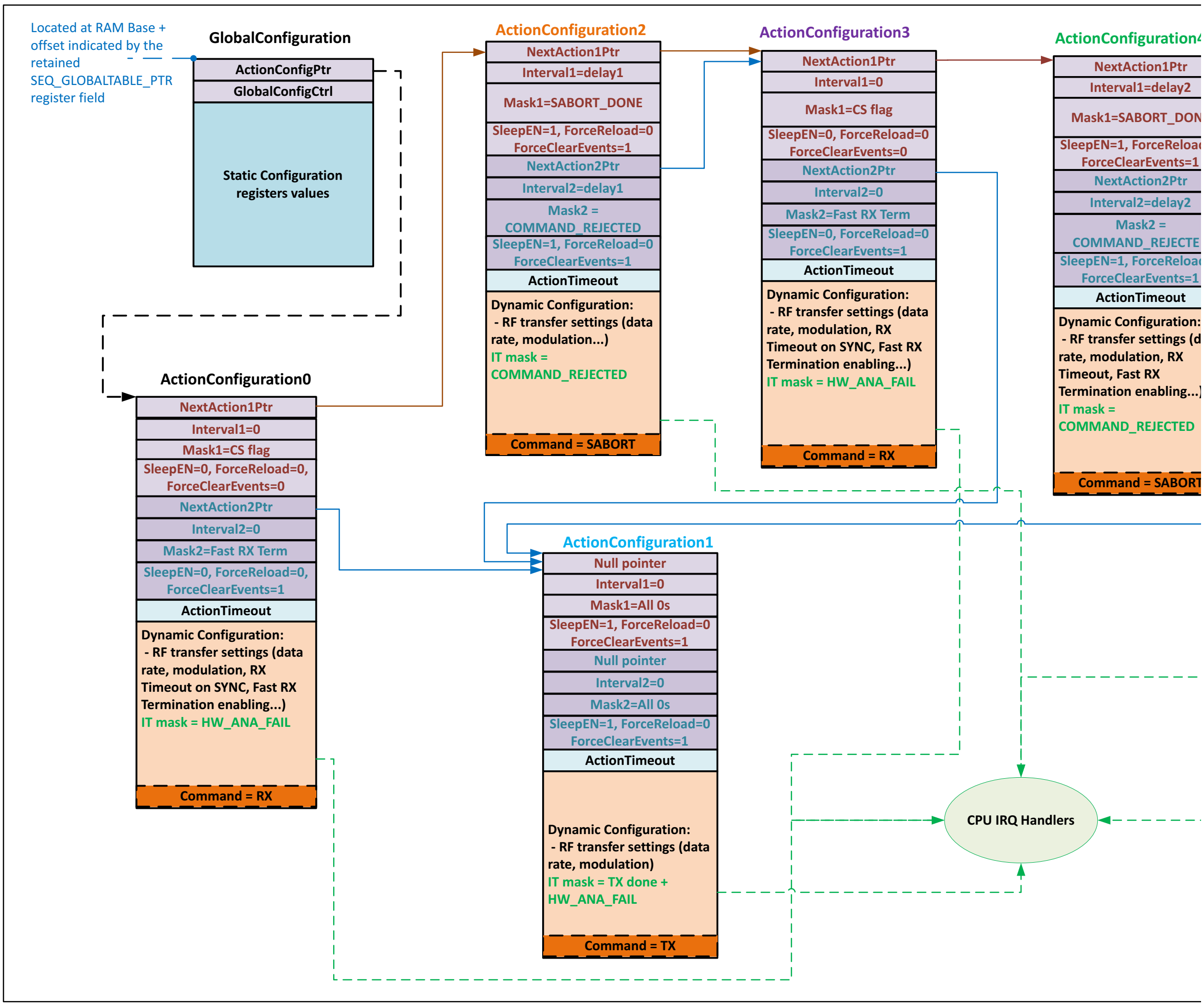
1. Starting by a RX with Fast RX Termination mode enabled
2. Once the RX is launched:
   a) if the FAST_RX_TERM flag is raised, this means no power on the antenna → the device can transmit its frame by issuing a TX
   b) if a CS flag is raised, the device must interrupt the on-going RX by a SABORT command
3. After three RX with CS flag raised, the sequence stops trying and informs the CPU through the RX_TIMEOUT (the SW programs a very small RX Timeout on the last RX trial).
4. On the other end, if one of the three RX attempts is successful (Fast RX termination flag), the CPU is informed by interrupt on the TX_DONE flag once the transmission is over.

The sequence and its associated SeqActions are represented in the figure below with GlobalConfiguration and ActionConfiguration RAM tables.

The CPU may enable some interrupts (depending on the SeqAction) like:

- CS flag to manage a Low Power mode entry between SeqAction separated by long delays,
- RX_TIMEOUT to be informed the scenario ends with no transmission because the radio always detected traffic on the air,
- TX_DONE to be informed the transmission well occurred,
- HW_ANA_FAILURE to be informed if an issue occurs on analog radio side during RF transfer start-up.

Figure 301. LBT use-case through Sequencer example



This scenario involves 6 ActionConfiguration tables:

- ActionConfiguration0 table generates the RX command with Fast RX Termination enabled:
  - NextAction1 is selected if CS flag occurs (meaning power measured on the antenna). It targets a SABORT command (to stop the on-going RX sequence) with no Low Power mode allowed, to be chained immediately (no interval or small delay)
  - NextAction2 is selected if a Fast RX Term occurs (meaning no power measured on the antenna). It targets a TX command after no delay or a small delay (with no Low Power mode allowed)
  - the CPU may enable the HW_ANA_FAILURE interrupt to be informed in case of analog radio issue.
- ActionConfiguration1 table generates a TX command:
  - NextAction1 is a null pointer and no specific mask is selected as this is the end point of the sequence (CPU to manage the TX_DONE interrupt)
  - NextAction2 is a null pointer and no specific mask is selected as this is the end point of the sequence (CPU to manage the TX_DONE interrupt)
  - the CPU needs to enable the TX_DONE interrupt to be informed when the transmission is done and may enable the HW_ANA_FAILURE interrupt to be informed in case of analog radio issue
- ActionConfiguration2 table generates a SABORT command:
  - NextAction1 is selected if SABORT_DONE flag occurs (meaning reception stopped). It targets a RX command, to be chained after a defined delay (with Low Power mode allowed if the defined delay is long enough to justify it).
  - NextAction2 is selected if a COMMAND_REJECTED occurs (meaning SABORT command occurred too late and reception probably stopped by itself on RX_TIMEOUT). It targets the same ActionConfiguration than NextAction1 = RX command after a defined delay (with Low Power mode allowed if the delay is long enough)
  - the CPU may enable the COMMAND_REJECTED interrupt to be informed the RX command ended by itself
- ActionConfiguration3 table generates a RX command with Fast RX Termination enabled (second trial):
  - NextAction1 is selected if CS flag occurs (meaning power measured on the antenna). It targets a SABORT command (to stop the on-going RX sequence), to be chained immediately (no interval or small delay).
  - NextAction2 is selected if a Fast RX Term occurs (meaning no power measured on the antenna). It targets a TX command after no delay or a small delay (with no Low Power mode allowed)
  - the CPU may enable the HW_ANA_FAILURE interrupt to be informed in case of analog radio issue
- ActionConfiguration4 table generates a SABORT command:
  - NextAction1 is selected if SABORT_DONE flag occurs (meaning no more RF activity). It targets a RX command, to be chained after a defined delay (with Low Power mode allowed if the defined delay is long enough to justify it).
  - NextAction2 is selected if a COMMAND_REJECTED occurs (meaning SABORT command occurred too late). It targets the same ActionConfiguration than

NextAction1 = RX command after a defined delay (with Low Power mode allowed if the delay is long enough)

  - the CPU may enable the COMMAND_REJECTED interrupt to be informed the RX command ended by itself

- ActionConfiguration5 table generates a RX command with Fast RX Termination enabled and a small RX Timeout based on SYNC detection stop condition (last trial):
  - NextAction1 is a null pointer that is selected if RX Timeout occurs (meaning power measured on the antenna and the RX aborted automatically soon after thanks to the small RX Timeout on SYNC). It raises the SleepEn bit to allow a Low Power mode.
  - NextAction2 is selected if a Fast RX Term occurs (meaning no power measured on the antenna). It targets a TX command after no delay or a small delay (with no Low Power mode allowed)
  - the CPU may the HW_ANA_FAILURE interrupt to be informed in case of analog radio issue and the RX_TIMEOUT to be informed the scenario is ended without transmission.

Note: The last RX trial is based on a RX Timeout IRQ enable mask instead of the CS flag to let the SW know it is the last iteration. The Sequencer mask can be kept on CS flag or changed to RX Timeout. Using a small timeout based on SYNC valid info allows the HW to automatically abort the on-going RX a short time after the fast RX Termination sequence is ended.

30 Debug support (DBG)

30.1 SW debug features

The STM32WL30xx/31xx/33xx device JTAG ID[31:0] is the following:

- 0000 0010000000100111 00000100000 1
- (0x0202 7041)

The Cortex-M0+ subsystem of the STM32WL30xx/31xx/33xx embeds 4 breakpoints and 2 watchpoints.

STM32WL30xx/31xx/33xx devices embed the ARM serial wire debug port which enables Serial Wire debug (2-wire) to be connected to the CPU (default after power-on reset).

30.2 MCU debug component (DBG)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog and I2C during a breakpoint

30.2.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode: FCLK and HCLK are still active. Consequently, this mode does not impose any restrictions on the standard debug features.
- In DeepStop mode, the DBG_STOP bit must be previously set by the debugger.

This enables the internal RC oscillator clock to feed FCLK and HCLK in DeepStop mode.

30.2.2 Debug support for timers, watchdog and I2C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the I2C, the user can choose to block the SMBUS timeout during a breakpoint.

Note: 1 Only I2C1 is available only on STM32WL31xx/33xx devices.

2 I2C2 is available only on STM32WL33xx devices.

30.3 DBG registers

30.3.1 DBG configuration register (DBG_CR)

This register configures the low-power modes of the MCU under debug.

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support this feature, it is still possible for the user software to write to this register.

Address offset: 0x00

POR Reset: 0x0000 0000 (not reset by system reset)

Only 32-bit access supported

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

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

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

Bit 1 **DBG_STOP**: Allow debug of the CPU in Deepstop mode

0: Normal operation. All clocks are disabled automatically in STOP mode

1: Automatic clock stop disabled. All active CPU clocks and oscillators continue to run during STOP mode, allowing full CPU debug capability. On exit from STOP mode, the clock settings are set to the STOP mode exit state.

Bit 0 **DBG_SLEEP**: Allow debug of the CPU in SLEEP mode

0: Normal operation. All clocks are disabled automatically in SLEEP mode

1: Automatic clock stop disabled. All active CPU clocks and oscillators continue to run during SLEEP mode, allowing full CPU debug capability. On exit from SLEEP mode, the clock settings are set to the SLEEP mode exit state.

30.3.2 DBG APB0 freeze register (DBG_APB0_FZ)

This register configures the clocking of timers, RTC, IWDG, peripherals of the MCU under debug:

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address offset: 0x04

Power on reset (POR): 0x0000 0000 (not reset by system reset)

Only 32-bit access are supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DBG_IWDG_STOP	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM16_STOP	DBG_TIM2_STOP
	rw		rw											rw	rw

Bits 31:15 reserved, must be kept at reset value.

Bit 14 **DBG_IWDG_STOP**: IWDG stop in the CPU debug

0: Normal operation. IWDG continues to operate while the CPU is in debug mode
1: Stop in debug. IWDG is frozen while the CPU is in debug mode.

Bit 13 reserved, must be kept at reset value.

Bit 12 **DBG_RTC_STOP**: RTC stop in CPU debug

0: Normal operation. RTC continues to operate while the CPU is in debug mode
1: Stop in debug. RTC is frozen while the CPU is in debug mode.

Bits 11:2 reserved, must be kept at reset value.

Bit 1 **DBG_TIM16_STOP**: TIM16 stop in the CPU debug

0: Normal operation. TIM16 continues to operate while the CPU is in debug mode
1: Stop in debug. TIM16 is frozen while the CPU is in debug mode.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in the CPU debug

0: Normal operation. TIM2 continues to operate while the CPU is in debug mode
1: Stop in debug. TIM2 is frozen while the CPU is in debug mode.

30.3.3 DBG APB1 freeze register (DBG_APB1_FZ)

This register configures the clocking of the I2C SMBUS peripherals of the MCU under debug:

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address offset: 0x08

Power on reset (POR): 0x0000 0000 (not reset by system reset)

Only 32-bit access are supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C2_STOP	Res.	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
								rw		rw					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

Bit 23 **DBG_I2C2_STOP**(1): I2C2 SMBUS timeout stop in CPU debug
0: Normal operation. I2C2 SMBUS timeout continues to operate while the CPU is in debug mode
1: Stop in debug. I2C2 SMBUS timeou is frozen while the CPU is in debug mode.

Bit 22 reserved, must be kept at reset value.

Bit 21 **DBG_I2C1_STOP**(2): I2C1 SMBUS timeout stop in CPU debug
0: Normal operation. I2C1 SMBUS timeout continues to operate while the CPU is in debug mode
1: Stop in debug. I2C1 SMBUS timeou is frozen while the CPU is in debug mode.

Bits 20:0 reserved, must be kept at reset value.

1. I2C2 is available only on STM32WL33xx devices.
2. I2C1 is available only on STM32WL31xx/33xx devices.

30.4 DBG register map

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the DBG registers base address location in the STM32WL30xx/31xx/33xx.

Table 146. DBG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DBG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DGB_STOP	DBG_SLEEP
	Reset value																															0	0
0x04	DBG_APB0_FZ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM16_STOP	DBG_TIM2_STOP
	Reset value																		0		0											0	0
0x08	DBG_APB1_FZ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C2_STOP(1)	Res.	DBG_I2C1_STOP(2)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0		0																					

1. I2C2 is available only in the STM32WL33xx.
2. I2C1 is available only in STM32WL31xx/33xx devices.

31 Device electronic signature (DESIG)

The device electronic signature is stored in the system memory area of the flash memory module, and can be read using the debug interface or by the CPU.

It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match the characteristics of the microcontroller.

31.1 DESIG registers

Base address: 0x10001EF0

31.1.1 Unique device ID register 1 (UID64_R1)

This unique 64-bit device identifier provides a reference number, unique for a given device and in any context. These bits cannot be altered by the user.

Address offset: 0x00

Reset value: 0xXXXX XXXX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 31:0 **UID[31:0]**: unique serial number (first 4 bytes).

31.1.2 Unique device ID register 2 (UID64_R2)

Address offset: 0x04

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	Res.	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[63:32]**: unique serial number (last 4 bytes).

32 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

33 Revision history

Table 147. Document revision history

Date	Revision	Changes
27-May-2024	1	First release.
15-Nov-2024	2	Updated: – R_{ref} and f_{XO} range to “48 or 50 MHz” – *Section 5: Power controller (PWRC)* (SMPS typos) – *Table 8: GPIO alternate options AF0 - AF3* (corrected LPUART_TX and USART_RX signal names) – *Table 13: System clock details* – *Section 7.3.2: I/O pin alternate function multiplexer and mapping* – *Payload* – Corrected register name in*Section 29.10.50: RF_INFO_OUT register (RF_INFO_OUT)* Added *Section 31: Device electronic signature (DESIG)*
08-Jan-2025	3	Corrected (minor) typos in: – *Figure 3: AHB up/down converter* – *Section 5.5: SMPS step down regulator* – *Section 5.5.1: SMPS bypass on-the-fly (BOF)* Corrected SMPSBOMSEL[1:0] default value in *Control register 5 (PWRC_CR5)* and *Table 13: PWRC register map and reset values*.
24-Feb-2025	4	Extended scope to STM32WL30xx and STM32WL31xx RPNs. Revised terminology in *Section 24: Inter-integrated circuit (I2C) interface*. Corrected typo in *Equation 5 - Data rate programming formula (in ksps)*.
27-Jun-2025	5	Added: – *Table 3: Memory map and peripheral register boundary addresses* – *Table 5: Address remapping depending on the REMAP bit* – *Section 9.7.5: Enabling protection example* – *Section 9.7.6: Write page protection example* Added product and package applicability notices in: – *Introduction* – *Section 4: I/O operating modes* – *Section 8.2: System controller register descriptions* – *Section 20.4: TIM2 registers* Continued...

Table 147. Document revision history

Date	Revision	Changes
27-Jun-2025	5	Made corrections to: – *Table 7: Interrupt vectors* – *Section 5.7: PWRC register map* – *Figure 13: System clock details* – *Section 6.7: RCC register map* – *Section 7.3.2: I/O pin alternate function multiplexer and mapping* – *Configuration register (CONFIG)* – *LCSC interrupt status register (LCSC_ISR)* – *Section 15: LC sensor controller (LCSC)* – *Table 118: RX mode summary* – *SPI status register (SPIx_SR)* – *RSSI_FLT register (RSSI_FLT)* – *Table 114: Frequency bands*