# RM0471
# Reference manual

## Multiprotocol wireless 32-bit MCU Arm®-based Cortex®-M4 with FPU, Bluetooth® Low Energy or 802.15.4 radio solution

## Introduction

This document is addressed to application developers. It provides complete information on how to use the STM32WB50CG/30CE microcontroller memory and peripherals.

These multiprotocol wireless and ultra-low-power devices embed a powerful and ultra-low-power radio compliant with the Bluetooth® Low Energy SIG specification 5.4 and with IEEE 802.15.4-2011. They contain a dedicated Arm® Cortex®-M0+ core for performing the real-time low layer operation, and include ST state of the art patented technology.

The STM32WB50CG provides value line microcontrollers with 1 Mbyte flash memory and 128 Kbyte SRAM.

The STM32WB30CE provides value line microcontrollers with 512 Kbyte flash memory and 96 Kbyte SRAM.

## Related documents

Available from STMicroelectronics website *www.st.com*:

- STM32WB50CG/30CE datasheet
- STM32WB50CG/30CE errata sheet

For information on the Arm® Cortex®-M4 and Cortex®-M0+ cores refer, to the corresponding technical reference manuals, available from the www.arm.com website.

For information on 802.15.4 refer to the IEEE website (www.ieee.org).

For information on Bluetooth® refer to www.bluetooth.com.

# Contents


**1 Documentation conventions . . . . . 51**

1.1 General information . . . . . 51

1.2 List of abbreviations for registers . . . . . 51

1.3 Glossary . . . . . 52

1.4 Availability of peripherals . . . . . 52

**2 System and memory overview . . . . . 53**

2.1 System architecture . . . . . 53

2.1.1 S0: CPU1 (CPU1 Cortex®-M4) I-bus . . . . . 54

2.1.2 S1: CPU1 (CPU1 Cortex®-M4) D-bus . . . . . 54

2.1.3 S2: CPU1 (CPU1 Cortex®-M4) S-bus . . . . . 54

2.1.4 S3: CPU2 (Cortex®-M0+) S-bus . . . . . 55

2.1.5 S4: DMA-bus . . . . . 55

2.1.6 S6: Radio system-bus . . . . . 55

2.1.7 BusMatrix . . . . . 55

2.2 Memory organization . . . . . 56

2.2.1 Introduction . . . . . 56

2.2.2 Memory map and register boundary addresses . . . . . 57

2.2.3 Bit banding . . . . . 61

2.3 Boot configuration . . . . . 61

2.4 CPU2 boot . . . . . 63

2.5 CPU2 SRAM fetch disable . . . . . 63

**3 Embedded flash memory (FLASH) . . . . . 64**

3.1 Introduction . . . . . 64

3.2 FLASH main features . . . . . 64

3.3 FLASH functional description . . . . . 65

3.3.1 Flash memory organization . . . . . 65

3.3.2 Empty check . . . . . 66

3.3.3 Error code correction (ECC) . . . . . 66

3.3.4 Read access latency . . . . . 66

3.3.5 Adaptive real-time memory accelerator (ART Accelerator) . . . . . 68

3.3.6 Flash memory program and erase operations . . . . . 71

3.3.7 Flash main memory erase sequences . . . . . 72
3.3.8 Flash main memory programming sequences . . . . . 74
3.4 FLASH option bytes . . . . . 79
3.4.1 Option bytes description . . . . . 79
3.4.2 Option bytes programming . . . . . 86
3.5 FLASH UID64 . . . . . 88
3.6 Flash memory protection . . . . . 89
3.6.1 Read protection (RDP) . . . . . 89
3.6.2 Proprietary code readout protection (PCROP) . . . . . 93
3.6.3 Write protection (WRP) . . . . . 94
3.6.4 CPU2 security (ESE) . . . . . 95
3.7 FLASH program/erase suspension . . . . . 96
3.8 FLASH interrupts . . . . . 97
3.9 Register access protection . . . . . 97
3.10 FLASH registers . . . . . 98
3.10.1 Flash memory access control register (FLASH_ACR) . . . . . 98
3.10.2 Flash memory key register (FLASH_KEYR) . . . . . 99
3.10.3 Flash memory option key register (FLASH_OPTKEYR) . . . . . 99
3.10.4 Flash memory status register (FLASH_SR) . . . . . 100
3.10.5 Flash memory control register (FLASH_CR) . . . . . 101
3.10.6 Flash memory ECC register (FLASH_ECCR) . . . . . 103
3.10.7 Flash memory option register (FLASH_OPTR) . . . . . 104
3.10.8 Flash memory PCROP zone A start address register (FLASH_PCROP1ASR) . . . . . 106
3.10.9 Flash memory PCROP zone A end address register (FLASH_PCROP1AER) . . . . . 107
3.10.10 Flash memory WRP area A address register (FLASH_WRP1AR) . . . 107
3.10.11 Flash memory WRP area B address register (FLASH_WRP1BR) . . . 108
3.10.12 Flash memory PCROP zone B start address register (FLASH_PCROP1BSR) . . . . . 108
3.10.13 Flash memory PCROP zone B end address register (FLASH_PCROP1BER) . . . . . 109
3.10.14 Flash memory IPCC mailbox data buffer address register (FLASH_IPCCBR) . . . . . 109
3.10.15 Flash memory CPU2 access control register (FLASH_C2ACR) . . . . . 109
3.10.16 Flash memory CPU2 status register (FLASH_C2SR) . . . . . 110
3.10.17 Flash memory CPU2 control register (FLASH_C2CR) . . . . . 112
3.10.18 Secure flash memory start address register (FLASH_SFR) . . . . . 113

3.10.19 Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR) . . . 114
3.10.20 FLASH register map . . . 116

**4 Radio system . . . 118**
4.1 Introduction . . . 118
4.2 Main features . . . 118
4.3 Radio system functional description . . . 119
4.3.1 General description . . . 119

**5 Cyclic redundancy check calculation unit (CRC) . . . 120**
5.1 Introduction . . . 120
5.2 CRC main features . . . 120
5.3 CRC functional description . . . 121
5.3.1 CRC block diagram . . . 121
5.3.2 CRC internal signals . . . 121
5.3.3 CRC operation . . . 121
5.4 CRC registers . . . 123
5.4.1 CRC data register (CRC_DR) . . . 123
5.4.2 CRC independent data register (CRC_IDR) . . . 123
5.4.3 CRC control register (CRC_CR) . . . 124
5.4.4 CRC initial value (CRC_INIT) . . . 125
5.4.5 CRC polynomial (CRC_POL) . . . 125
5.4.6 CRC register map . . . 126

**6 Power control (PWR) . . . 127**
6.1 Power supplies . . . 127
6.1.1 Independent analog peripherals supply . . . 128
6.1.2 Battery backup domain . . . 128
6.1.3 Voltage regulator . . . 130
6.2 Power supply supervisor . . . 130
6.2.1 Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR) . . . 130
6.2.2 Programmable voltage detector (PVD) . . . 131
6.3 CPU2 boot . . . 132
6.4 Low-power modes . . . 134
6.4.1 Run mode . . . 140

6.4.2 Low-power run mode (LP run) . . . . . 140
6.4.3 Entering Low-power mode . . . . . 141
6.4.4 Exiting Low-power mode . . . . . 141
6.4.5 Sleep mode . . . . . 142
6.4.6 Low-power sleep mode (LP sleep) . . . . . 143
6.4.7 Stop0 mode . . . . . 144
6.4.8 Stop1 mode . . . . . 146
6.4.9 Stop2 mode . . . . . 147
6.4.10 Standby mode . . . . . 149
6.4.11 Shutdown mode . . . . . 151
6.4.12 Auto wake-up from Low-power mode . . . . . 152
6.5 Real-time radio information . . . . . 153
6.6 PWR registers . . . . . 154
6.6.1 PWR control register 1 (PWR_CR1) . . . . . 154
6.6.2 PWR control register 2 (PWR_CR2) . . . . . 155
6.6.3 PWR control register 3 (PWR_CR3) . . . . . 156
6.6.4 PWR control register 4 (PWR_CR4) . . . . . 157
6.6.5 PWR status register 1 (PWR_SR1) . . . . . 158
6.6.6 PWR status register 2 (PWR_SR2) . . . . . 159
6.6.7 PWR status clear register (PWR_SCR) . . . . . 159
6.6.8 PWR Port A pull-up control register (PWR_PUCRA) . . . . . 160
6.6.9 PWR Port A pull-down control register (PWR_PDCRA) . . . . . 161
6.6.10 PWR Port B pull-up control register (PWR_PUCRB) . . . . . 161
6.6.11 PWR Port B pull-down control register (PWR_PDCRB) . . . . . 162
6.6.12 PWR Port C pull-up control register (PWR_PUCRC) . . . . . 162
6.6.13 PWR Port C pull-down control register (PWR_PDCRC) . . . . . 163
6.6.14 PWR Port E pull-up control register (PWR_PUCRE) . . . . . 163
6.6.15 PWR Port E pull-down control register (PWR_PDCRE) . . . . . 163
6.6.16 PWR Port H pull-up control register (PWR_PUCRH) . . . . . 164
6.6.17 PWR Port H pull-down control register (PWR_PDCRH) . . . . . 164
6.6.18 PWR CPU2 control register 1 (PWR_C2CR1) . . . . . 165
6.6.19 PWR CPU2 control register 3 (PWR_C2CR3) . . . . . 166
6.6.20 PWR extended status and status clear register (PWR_EXTSCR) . . . 167
6.6.21 PWR register map and reset value table . . . . . 169

**7 Reset and clock control (RCC) . . . . . 171**
7.1 Reset . . . . . 171

7.1.1 Power reset . . . . . 171
7.1.2 System reset . . . . . 171
7.1.3 Backup domain reset . . . . . 173
7.2 Clocks . . . . . 173
7.2.1 HSE clock . . . . . 176
7.2.2 HSI16 clock . . . . . 178
7.2.3 MSI clock . . . . . 178
7.2.4 HSI48 clock . . . . . 179
7.2.5 PLL . . . . . 179
7.2.6 LSE clock . . . . . 180
7.2.7 LSI1 clock . . . . . 181
7.2.8 LSI2 clock . . . . . 181
7.2.9 System clock (SYSCLK) selection . . . . . 182
7.2.10 Clock source frequency . . . . . 182
7.2.11 Clock security system (CSS) on HSE . . . . . 182
7.2.12 Clock security system on LSE (LSECSS) . . . . . 183
7.2.13 LSI source selection . . . . . 183
7.2.14 ADC clock . . . . . 183
7.2.15 RTC clock . . . . . 183
7.2.16 Timer clock . . . . . 184
7.2.17 Watchdog clock . . . . . 184
7.2.18 True RNG clock . . . . . 185
7.2.19 Clock-out capability . . . . . 185
7.2.20 Internal/external clock measurement with TIM16/TIM17 . . . . . 185
7.2.21 Peripheral clocks enable . . . . . 187
7.3 Low-power modes . . . . . 189
7.4 RCC registers . . . . . 191
7.4.1 RCC clock control register (RCC_CR) . . . . . 191
7.4.2 RCC internal clock sources calibration register (RCC_ICSCR) . . . . . 194
7.4.3 RCC clock configuration register (RCC_CFGR) . . . . . 195
7.4.4 RCC PLL configuration register (RCC_PLLCFGR) . . . . . 197
7.4.5 RCC clock interrupt enable register (RCC_CIER) . . . . . 200
7.4.6 RCC clock interrupt flag register (RCC_CIFR) . . . . . 202
7.4.7 RCC clock interrupt clear register (RCC_CICR) . . . . . 203
7.4.8 RCC AHB1 peripheral reset register (RCC_AHB1RSTR) . . . . . 205
7.4.9 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) . . . . . 205
7.4.10 RCC AHB3 and AHB4 peripheral reset register (RCC_AHB3RSTR) . 206

7.4.11 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1) . . . . . 207
7.4.12 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2) . . . . . 208
7.4.13 RCC APB2 peripheral reset register (RCC_APB2RSTR) . . . . . 208
7.4.14 RCC APB3 peripheral reset register (RCC_APB3RSTR) . . . . . 209
7.4.15 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR) . . . . . 210
7.4.16 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR) . . . . . 211
7.4.17 RCC AHB3 and AHB4 peripheral clock enable register (RCC_AHB3ENR) . . . . . 212
7.4.18 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1) . . . 213
7.4.19 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2) . . . 213
7.4.20 RCC APB2 peripheral clock enable register (RCC_APB2ENR) . . . . . 214
7.4.21 RCC AHB1 peripheral clocks enable in Sleep modes register (RCC_AHB1SMENR) . . . . . 215
7.4.22 RCC AHB2 peripheral clocks enable in Sleep modes register (RCC_AHB2SMENR) . . . . . 216
7.4.23 RCC AHB3 and AHB4 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR) . . . . . 217
7.4.24 RCC APB1 peripheral clocks enable in Sleep mode register 1 (RCC_APB1SMENR1) . . . . . 218
7.4.25 RCC APB1 peripheral clocks enable in Sleep mode register 2 (RCC_APB1SMENR2) . . . . . 219
7.4.26 RCC APB2 peripheral clocks enable in Sleep mode register (RCC_APB2SMENR) . . . . . 219
7.4.27 RCC peripherals independent clock configuration register (RCC_CCIPR) . . . . . 220
7.4.28 RCC backup domain control register (RCC_BDCR) . . . . . 222
7.4.29 RCC control/status register (RCC_CSR) . . . . . 224
7.4.30 RCC clock recovery RC register (RCC_CRRCR) . . . . . 226
7.4.31 RCC clock HSE register (RCC_HSECR) . . . . . 226
7.4.32 RCC extended clock recovery register (RCC_EXTCFGR) . . . . . 227
7.4.33 RCC CPU2 AHB1 peripheral clock enable register (RCC_C2AHB1ENR) . . . . . 230
7.4.34 RCC CPU2 AHB2 peripheral clock enable register (RCC_C2AHB2ENR) . . . . . 230
7.4.35 RCC CPU2 AHB3 and AHB4 peripheral clock enable register (RCC_C2AHB3ENR) . . . . . 231
7.4.36 RCC CPU2 APB1 peripheral clock enable register 1 (RCC_C2APB1ENR1) . . . . . 232
7.4.37 RCC CPU2 APB1 peripheral clock enable register 2 (RCC_C2APB1ENR2) . . . . . 233

7.4.38 RCC CPU2 APB2 peripheral clock enable register (RCC_C2APB2ENR) . . . . . 234
7.4.39 RCC CPU2 APB3 peripheral clock enable register (RCC_C2APB3ENR) . . . . . 235
7.4.40 RCC CPU2 AHB1 peripheral clocks enable in Sleep modes register (RCC_C2AHB1SMENR) . . . . . 235
7.4.41 RCC CPU2 AHB2 peripheral clocks enable in Sleep modes register (RCC_C2AHB2SMENR) . . . . . 236
7.4.42 RCC CPU2 AHB3 and AHB4 peripheral clocks enable in Sleep mode register (RCC_C2AHB3SMENR) . . . . . 237
7.4.43 RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 1 (RCC_C2APB1SMENR1) . . . . . 238
7.4.44 RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 2 (RCC_C2APB1SMENR2) . . . . . 239
7.4.45 RCC CPU2 APB2 peripheral clocks enable in Sleep mode register (RCC_C2APB2SMENR) . . . . . 240
7.4.46 RCC CPU2 APB3 peripheral clock enable in Sleep mode register (RCC_C2APB3SMENR) . . . . . 241
7.4.47 RCC register map . . . . . 242

**8 General-purpose I/Os (GPIO) . . . . . 248**

8.1 Introduction . . . . . 248
8.2 GPIO main features . . . . . 248
8.3 GPIO functional description . . . . . 248
8.3.1 General-purpose I/O (GPIO) . . . . . 251
8.3.2 I/O pin alternate function multiplexer and mapping . . . . . 251
8.3.3 I/O port control registers . . . . . 252
8.3.4 I/O port data registers . . . . . 252
8.3.5 I/O data bitwise handling . . . . . 252
8.3.6 GPIO locking mechanism . . . . . 253
8.3.7 I/O alternate function input/output . . . . . 253
8.3.8 External interrupt/wake-up lines . . . . . 253
8.3.9 Input configuration . . . . . 253
8.3.10 Output configuration . . . . . 254
8.3.11 Alternate function configuration . . . . . 255
8.3.12 Analog configuration . . . . . 255
8.3.13 Using the LSE oscillator pins as GPIOs . . . . . 256
8.3.14 Using the GPIO pins in the RTC supply domain . . . . . 256
8.3.15 Using PH3 as GPIO . . . . . 256

8.4 GPIO registers . . . . . 257
8.4.1 GPIO port mode register (GPIOx_MODER) (x = A to C and E and H) . . . . . 257
8.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to C and E and H) . . . . . 258
8.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to C and E and H) . . . . . 258
8.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to C and E and H) . . . . . 259
8.4.5 GPIO port input data register (GPIOx_IDR) (x = A to C and E and H) . . . . . 260
8.4.6 GPIO port output data register (GPIOx_ODR) (x = A to C and E and H) . . . . . 260
8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to C and E and H) . . . . . 261
8.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to C and E and H) . . . . . 261
8.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to C and E and H) . . . . . 263
8.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to C and E and H) . . . . . 264
8.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to C and E and H) . 265
8.4.12 GPIO register map . . . . . 266

**9 System configuration controller (SYSCFG) . . . . . 270**
9.1 SYSCFG main features . . . . . 270
9.2 SYSCFG registers . . . . . 270
9.2.1 SYSCFG memory remap register (SYSCFG_MEMRMP) . . . . . 270
9.2.2 SYSCFG configuration register 1 (SYSCFG_CFGR1) . . . . . 271
9.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1) . . . . . 272
9.2.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2) . . . . . 273
9.2.5 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3) . . . . . 274
9.2.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4) . . . . . 276
9.2.7 SYSCFG SRAM2 control and status register (SYSCFG_SCSR) . . . . 277
9.2.8 SYSCFG configuration register 2 (SYSCFG_CFGR2) . . . . . 278
9.2.9 SYSCFG SRAM2 write protection register (SYSCFG_SWPR1) . . . . . 279
9.2.10 SYSCFG SRAM2 key register (SYSCFG_SKR) . . . . . 279

9.2.11 SYSCFG SRAM2 write protection register 2 (SYSCFG_SWPR2) . . . 279
9.2.12 SYSCFG CPU1 interrupt mask register 1 (SYSCFG_IMR1) . . . . . 280
9.2.13 SYSCFG CPU1 interrupt mask register 2 (SYSCFG_IMR2) . . . . . 280
9.2.14 SYSCFG CPU2 interrupt mask register 1 (SYSCFG_C2IMR1) . . . . . 281
9.2.15 SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2) . . . . . 281
9.2.16 SYSCFG secure IP control register (SYSCFG_SIPCR) . . . . . 282
9.2.17 SYSCFG register map . . . . . 284

**10 Peripherals interconnect matrix . . . . . 286**
10.1 Introduction . . . . . 286
10.2 Connection summary . . . . . 286
10.3 Interconnection details . . . . . 287
10.3.1 From timer (TIM1/TIM2/TIM17) to timer (TIM1/TIM2) . . . . . 287
10.3.2 From timer (TIM1/TIM2) and EXTI to ADC (ADC1) . . . . . 287
10.3.3 From ADC (ADC1) to timer (TIM1) . . . . . 288
10.3.4 From HSE, LSE, LSI, MSI, MCO, RTC to timers (TIM2/TIM16/TIM17) . . . . . 288
10.3.5 From RTC to low-power timers (LPTIM1/LPTIM2) . . . . . 289
10.3.6 From internal analog to ADC1 . . . . . 289
10.3.7 From system errors to timers (TIM1/TIM16/TIM17) . . . . . 289
10.3.8 From timers (TIM16/TIM17) to IRTIM . . . . . 290

**11 Direct memory access controller (DMA) . . . . . 291**
11.1 Introduction . . . . . 291
11.2 DMA main features . . . . . 291
11.3 DMA implementation . . . . . 292
11.3.1 DMA1 . . . . . 292
11.3.2 DMA request mapping . . . . . 292
11.4 DMA functional description . . . . . 292
11.4.1 DMA block diagram . . . . . 292
11.4.2 DMA pins and internal signals . . . . . 293
11.4.3 DMA transfers . . . . . 293
11.4.4 DMA arbitration . . . . . 294
11.4.5 DMA channels . . . . . 294
11.4.6 DMA data width, alignment, and endianness . . . . . 298
11.4.7 DMA error management . . . . . 299
11.5 DMA interrupts . . . . . 300

11.6 DMA registers . . . . . 300
11.6.1 DMA interrupt status register (DMA_ISR) . . . . . 300
11.6.2 DMA interrupt flag clear register (DMA_IFCR) . . . . . 303
11.6.3 DMA channel x configuration register (DMA_CCRx) . . . . . 304
11.6.4 DMA channel x number of data to transfer register (DMA_CNDTRx) . 307
11.6.5 DMA channel x peripheral address register (DMA_CPARx) . . . . . 307
11.6.6 DMA channel x memory address register (DMA_CMARx) . . . . . 308
11.6.7 DMA register map . . . . . 308

**12 DMA request multiplexer (DMAMUX) . . . . . 311**
12.1 Introduction . . . . . .311
12.2 DMAMUX main features . . . . . 312
12.3 DMAMUX implementation . . . . . 312
12.3.1 DMAMUX instantiation . . . . . 312
12.3.2 DMAMUX mapping . . . . . 312
12.4 DMAMUX functional description . . . . . 315
12.4.1 DMAMUX block diagram . . . . . 315
12.4.2 DMAMUX signals . . . . . 316
12.4.3 DMAMUX channels . . . . . 316
12.4.4 DMAMUX request line multiplexer . . . . . 316
12.4.5 DMAMUX request generator . . . . . 319
12.5 DMAMUX interrupts . . . . . 320
12.6 DMAMUX registers . . . . . 321
12.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR) . . . . . 321
12.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR) . . . . . 322
12.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR) . . . . . 322
12.6.4 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR) . . . . . 323
12.6.5 DMAMUX request generator interrupt status register (DMAMUX_RGSR) . . . . . 324
12.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR) . . . . . 324
12.6.7 DMAMUX register map . . . . . 325

**13 Nested vectored interrupt controller (NVIC) . . . . . 327**

13.1 NVIC main features . . . . . 327
13.2 Interrupt block diagram . . . . . 327
13.3 Interrupt and exception vectors . . . . . 328
13.4 Interrupt list . . . . . 333

**14 Extended interrupt and event controller (EXTI) . . . . . 336**
14.1 EXTI main features . . . . . 336
14.2 EXTI block diagram . . . . . 336
14.2.1 EXTI connections between peripherals and CPU . . . . . 338
14.3 EXTI functional description . . . . . 339
14.3.1 EXTI configurable event input wakeup . . . . . 339
14.3.2 EXTI direct event input wakeup . . . . . 341
14.4 EXTI functional behavior . . . . . 341
14.5 EXTI registers . . . . . 343
14.5.1 EXTI rising trigger selection register (EXTI_RTSR1) . . . . . 343
14.5.2 EXTI falling trigger selection register (EXTI_FTSR1) . . . . . 344
14.5.3 EXTI software interrupt event register (EXTI_SWIER1) . . . . . 344
14.5.4 EXTI pending register (EXTI_PR1) . . . . . 345
14.5.5 EXTI rising trigger selection register (EXTI_RTSR2) . . . . . 345
14.5.6 EXTI falling trigger selection register (EXTI_FTSR2) . . . . . 346
14.5.7 EXTI software interrupt event register (EXTI_SWIER2) . . . . . 346
14.5.8 EXTI pending register (EXTI_PR2) . . . . . 347
14.5.9 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1) . . . . . 348
14.5.10 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR1) . . . 348
14.5.11 EXTI CPU wakeup with event mask register (EXTI_EMR1) . . . . . 349
14.5.12 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR1) . . . . . 349
14.5.13 EXTI CPU wakeup with interrupt mask register (EXTI_IMR2) . . . . . 350
14.5.14 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR2) . . . 350
14.5.15 EXTI CPU wakeup with event mask register (EXTI_EMR2) . . . . . 351
14.5.16 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR2) . . . . . 351
14.5.17 EXTI register map . . . . . 352

**15 Analog-to-digital converter (ADC) . . . . . 354**
15.1 Introduction . . . . . 354
15.2 ADC main features . . . . . 354
15.3 ADC functional description . . . . . 356

15.3.1 ADC block diagram . . . . . 356
15.3.2 ADC pins and internal signals . . . . . 357
15.3.3 ADC clocks . . . . . 358
15.3.4 ADC1 connectivity . . . . . 360
15.3.5 Slave AHB interface . . . . . 361
15.3.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) . . . . . 361
15.3.7 Single-ended and differential input channels . . . . . 362
15.3.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT) . . . . . 362
15.3.9 ADC on-off control (ADEN, ADDIS, ADRDY) . . . . . 365
15.3.10 Constraints when writing the ADC control bits . . . . . 366
15.3.11 Channel selection (SQRx, JSQRx) . . . . . 367
15.3.12 Channel-wise programmable sampling time (SMPR1, SMPR2) . . . . . 368
15.3.13 Single conversion mode (CONT = 0) . . . . . 368
15.3.14 Continuous conversion mode (CONT = 1) . . . . . 369
15.3.15 Starting conversions (ADSTART, JADSTART) . . . . . 370
15.3.16 ADC timing . . . . . 371
15.3.17 Stopping an ongoing conversion (ADSTP, JADSTP) . . . . . 371
15.3.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) . . . . . 373
15.3.19 Injected channel management . . . . . 375
15.3.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) . . . . . 376
15.3.21 Queue of context for injected conversions . . . . . 377
15.3.22 Programmable resolution (RES) - Fast conversion mode . . . . . 385
15.3.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP) . . 386
15.3.24 End of conversion sequence (EOS, JEOS) . . . . . 386
15.3.25 Timing diagrams example (single/continuous modes, hardware/software triggers) . . . . . 387
15.3.26 Data management . . . . . 389
15.3.27 Dynamic low-power features . . . . . 394
15.3.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx) . . . . . 399
15.3.29 Oversampler . . . . . 403
15.3.30 Temperature sensor . . . . . 408
15.3.31 VBAT supply monitoring . . . . . 410
15.3.32 Monitoring the internal voltage reference . . . . . 410
15.4 ADC in low-power mode . . . . . 412
15.5 ADC interrupts . . . . . 413

15.6 ADC registers . . . . . 414
15.6.1 ADC interrupt and status register (ADC_ISR) . . . . . 414
15.6.2 ADC interrupt enable register (ADC_IER) . . . . . 416
15.6.3 ADC control register (ADC_CR) . . . . . 418
15.6.4 ADC configuration register (ADC_CFGR) . . . . . 421
15.6.5 ADC configuration register 2 (ADC_CFGR2) . . . . . 425
15.6.6 ADC sample time register 1 (ADC_SMPR1) . . . . . 426
15.6.7 ADC sample time register 2 (ADC_SMPR2) . . . . . 427
15.6.8 ADC watchdog threshold register 1 (ADC_TR1) . . . . . 428
15.6.9 ADC watchdog threshold register 2 (ADC_TR2) . . . . . 428
15.6.10 ADC watchdog threshold register 3 (ADC_TR3) . . . . . 429
15.6.11 ADC regular sequence register 1 (ADC_SQR1) . . . . . 430
15.6.12 ADC regular sequence register 2 (ADC_SQR2) . . . . . 431
15.6.13 ADC regular sequence register 3 (ADC_SQR3) . . . . . 432
15.6.14 ADC regular sequence register 4 (ADC_SQR4) . . . . . 433
15.6.15 ADC regular data register (ADC_DR) . . . . . 433
15.6.16 ADC injected sequence register (ADC_JSQR) . . . . . 434
15.6.17 ADC offset y register (ADC_OFRy) . . . . . 436
15.6.18 ADC injected channel y data register (ADC_JDRy) . . . . . 437
15.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) . . . . 437
15.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) . . . . 438
15.6.21 ADC differential mode selection register (ADC_DIFSEL) . . . . . 438
15.6.22 ADC calibration factors (ADC_CALFACT) . . . . . 439
15.7 ADC common registers . . . . . 439
15.7.1 ADC common status register (ADC_CSR) . . . . . 439
15.7.2 ADC common control register (ADC_CCR) . . . . . 440
15.8 ADC register map . . . . . 442

**16 True random number generator (RNG) . . . . . 445**
16.1 Introduction . . . . . 445
16.2 RNG main features . . . . . 445
16.3 RNG functional description . . . . . 446
16.3.1 RNG block diagram . . . . . 446
16.3.2 RNG internal signals . . . . . 446
16.3.3 Random number generation . . . . . 447
16.3.4 RNG initialization . . . . . 449

16.3.5 RNG operation . . . . . 450
16.3.6 RNG clocking . . . . . 451
16.3.7 Error management . . . . . 451
16.3.8 RNG low-power use . . . . . 452
16.4 RNG interrupts . . . . . 453
16.5 RNG processing time . . . . . 453
16.6 RNG entropy source validation . . . . . 453
16.6.1 Introduction . . . . . 453
16.6.2 Validation conditions . . . . . 453
16.7 RNG registers . . . . . 454
16.7.1 RNG control register (RNG_CR) . . . . . 454
16.7.2 RNG status register (RNG_SR) . . . . . 454
16.7.3 RNG data register (RNG_DR) . . . . . 455
16.7.4 RNG register map . . . . . 456

**17 AES hardware accelerator (AES) . . . . . 457**
17.1 Introduction . . . . . 457
17.2 AES main features . . . . . 457
17.3 AES implementation . . . . . 457
17.4 AES functional description . . . . . 458
17.4.1 AES block diagram . . . . . 458
17.4.2 AES internal signals . . . . . 458
17.4.3 AES cryptographic core . . . . . 458
17.4.4 AES procedure to perform a cipher operation . . . . . 464
17.4.5 AES decryption round key preparation . . . . . 467
17.4.6 AES ciphertext stealing and data padding . . . . . 468
17.4.7 AES task suspend and resume . . . . . 468
17.4.8 AES basic chaining modes (ECB, CBC) . . . . . 469
17.4.9 AES counter (CTR) mode . . . . . 474
17.4.10 AES Galois/counter mode (GCM) . . . . . 476
17.4.11 AES Galois message authentication code (GMAC) . . . . . 481
17.4.12 AES counter with CBC-MAC (CCM) . . . . . 483
17.4.13 AES data registers and data swapping . . . . . 488
17.4.14 AES key registers . . . . . 490
17.4.15 AES initialization vector registers . . . . . 490
17.4.16 AES DMA interface . . . . . 491

17.4.17 AES error management . . . . . 492
17.5 AES interrupts . . . . . 493
17.6 AES processing latency . . . . . 493
17.7 AES registers . . . . . 494
17.7.1 AES control register (AES_CR) . . . . . 494
17.7.2 AES status register (AES_SR) . . . . . 497
17.7.3 AES data input register (AES_DINR) . . . . . 498
17.7.4 AES data output register (AES_DOUTR) . . . . . 498
17.7.5 AES key register 0 (AES_KEYR0) . . . . . 499
17.7.6 AES key register 1 (AES_KEYR1) . . . . . 500
17.7.7 AES key register 2 (AES_KEYR2) . . . . . 500
17.7.8 AES key register 3 (AES_KEYR3) . . . . . 500
17.7.9 AES initialization vector register 0 (AES_IVR0) . . . . . 501
17.7.10 AES initialization vector register 1 (AES_IVR1) . . . . . 501
17.7.11 AES initialization vector register 2 (AES_IVR2) . . . . . 501
17.7.12 AES initialization vector register 3 (AES_IVR3) . . . . . 502
17.7.13 AES key register 4 (AES_KEYR4) . . . . . 502
17.7.14 AES key register 5 (AES_KEYR5) . . . . . 502
17.7.15 AES key register 6 (AES_KEYR6) . . . . . 503
17.7.16 AES key register 7 (AES_KEYR7) . . . . . 503
17.7.17 AES suspend registers (AES_SUSPxR) . . . . . 503
17.7.18 AES register map . . . . . 504

**18 Public key accelerator (PKA) . . . . . 506**
18.1 Introduction . . . . . 506
18.2 PKA main features . . . . . 506
18.3 PKA functional description . . . . . 506
18.3.1 PKA block diagram . . . . . 506
18.3.2 PKA internal signals . . . . . 507
18.3.3 PKA reset and clocks . . . . . 507
18.3.4 PKA public key acceleration . . . . . 507
18.3.5 Typical applications for PKA . . . . . 509
18.3.6 PKA procedure to perform an operation . . . . . 511
18.3.7 PKA error management . . . . . 512
18.4 PKA operating modes . . . . . 512
18.4.1 Introduction . . . . . 512

18.4.2 Montgomery parameter computation . . . 513
18.4.3 Modular addition . . . 513
18.4.4 Modular subtraction . . . 514
18.4.5 Modular and Montgomery multiplication . . . 514
18.4.6 Modular exponentiation . . . 515
18.4.7 Modular inversion . . . 516
18.4.8 Modular reduction . . . 517
18.4.9 Arithmetic addition . . . 517
18.4.10 Arithmetic subtraction . . . 517
18.4.11 Arithmetic multiplication . . . 518
18.4.12 Arithmetic comparison . . . 518
18.4.13 RSA CRT exponentiation . . . 518
18.4.14 Point on elliptic curve Fp check . . . 519
18.4.15 ECC Fp scalar multiplication . . . 520
18.4.16 ECDSA sign . . . 521
18.4.17 ECDSA verification . . . 523
18.5 Example of configurations and processing times . . . 524
18.5.1 Supported elliptic curves . . . 524
18.5.2 Computation times . . . 526
18.6 PKA interrupts . . . 527
18.7 PKA registers . . . 528
18.7.1 PKA control register (PKA_CR) . . . 528
18.7.2 PKA status register (PKA_SR) . . . 529
18.7.3 PKA clear flag register (PKA_CLRFR) . . . 530
18.7.4 PKA RAM . . . 530
18.7.5 PKA register map . . . 531

**19 Advanced-control timer (TIM1) . . . 532**
19.1 TIM1 introduction . . . 532
19.2 TIM1 main features . . . 533
19.3 TIM1 functional description . . . 535
19.3.1 Time-base unit . . . 535
19.3.2 Counter modes . . . 537
19.3.3 Repetition counter . . . 548
19.3.4 External trigger input . . . 550
19.3.5 Clock selection . . . 551

19.3.6 Capture/compare channels . . . . . 555
19.3.7 Input capture mode . . . . . 557
19.3.8 PWM input mode . . . . . 558
19.3.9 Forced output mode . . . . . 559
19.3.10 Output compare mode . . . . . 560
19.3.11 PWM mode . . . . . 561
19.3.12 Asymmetric PWM mode . . . . . 564
19.3.13 Combined PWM mode . . . . . 565
19.3.14 Combined 3-phase PWM mode . . . . . 566
19.3.15 Complementary outputs and dead-time insertion . . . . . 567
19.3.16 Using the break function . . . . . 569
19.3.17 Bidirectional break inputs . . . . . 575
19.3.18 Clearing the OCxREF signal on an external event . . . . . 577
19.3.19 6-step PWM generation . . . . . 578
19.3.20 One-pulse mode . . . . . 579
19.3.21 Retriggerable one pulse mode . . . . . 580
19.3.22 Encoder interface mode . . . . . 581
19.3.23 UIF bit remapping . . . . . 583
19.3.24 Timer input XOR function . . . . . 584
19.3.25 Interfacing with Hall sensors . . . . . 584
19.3.26 Timer synchronization . . . . . 587
19.3.27 ADC synchronization . . . . . 591
19.3.28 DMA burst mode . . . . . 591
19.3.29 Debug mode . . . . . 592
19.4 TIM1 registers . . . . . 593
19.4.1 TIM1 control register 1 (TIM1_CR1) . . . . . 593
19.4.2 TIM1 control register 2 (TIM1_CR2) . . . . . 594
19.4.3 TIM1 slave mode control register (TIM1_SMCR) . . . . . 597
19.4.4 TIM1 DMA/interrupt enable register (TIM1_DIER) . . . . . 599
19.4.5 TIM1 status register (TIM1_SR) . . . . . 601
19.4.6 TIM1 event generation register (TIM1_EGR) . . . . . 603
19.4.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1) . . . . . 604
19.4.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) . . . . . 605
19.4.9 TIM1 capture/compare mode register 2 (TIM1_CCMR2) . . . . . 608

19.4.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) . . . . . 609
19.4.11 TIM1 capture/compare enable register (TIM1_CCER) . . . . . 611
19.4.12 TIM1 counter (TIM1_CNT) . . . . . 614
19.4.13 TIM1 prescaler (TIM1_PSC) . . . . . 614
19.4.14 TIM1 auto-reload register (TIM1_ARR) . . . . . 614
19.4.15 TIM1 repetition counter register (TIM1_RCR) . . . . . 615
19.4.16 TIM1 capture/compare register 1 (TIM1_CCR1) . . . . . 615
19.4.17 TIM1 capture/compare register 2 (TIM1_CCR2) . . . . . 616
19.4.18 TIM1 capture/compare register 3 (TIM1_CCR3) . . . . . 616
19.4.19 TIM1 capture/compare register 4 (TIM1_CCR4) . . . . . 617
19.4.20 TIM1 break and dead-time register (TIM1_BDTR) . . . . . 617
19.4.21 TIM1 DMA control register (TIM1_DCR) . . . . . 621
19.4.22 TIM1 DMA address for full transfer (TIM1_DMAR) . . . . . 622
19.4.23 TIM1 option register 1 (TIM1_OR1) . . . . . 623
19.4.24 TIM1 capture/compare mode register 3 (TIM1_CCMR3) . . . . . 623
19.4.25 TIM1 capture/compare register 5 (TIM1_CCR5) . . . . . 624
19.4.26 TIM1 capture/compare register 6 (TIM1_CCR6) . . . . . 625
19.4.27 TIM1 alternate function option register 1 (TIM1_AF1) . . . . . 626
19.4.28 TIM1 Alternate function register 2 (TIM1_AF2) . . . . . 627
19.4.29 TIM1 timer input selection register (TIM1_TISEL) . . . . . 627
19.4.30 TIM1 register map . . . . . 629

**20 General-purpose timer (TIM2) . . . . . 632**

20.1 TIM2 introduction . . . . . 632
20.2 TIM2 main features . . . . . 632
20.3 TIM2 functional description . . . . . 634
20.3.1 Time-base unit . . . . . 634
20.3.2 Counter modes . . . . . 636

20.3.3 Clock selection . . . . . 646
20.3.4 Capture/Compare channels . . . . . 650
20.3.5 Input capture mode . . . . . 652
20.3.6 PWM input mode . . . . . 653
20.3.7 Forced output mode . . . . . 654
20.3.8 Output compare mode . . . . . 654
20.3.9 PWM mode . . . . . 655
20.3.10 Asymmetric PWM mode . . . . . 659
20.3.11 Combined PWM mode . . . . . 659
20.3.12 Clearing the OCxREF signal on an external event . . . . . 660
20.3.13 One-pulse mode . . . . . 662
20.3.14 Retriggerable one pulse mode . . . . . 663
20.3.15 Encoder interface mode . . . . . 664
20.3.16 UIF bit remapping . . . . . 666
20.3.17 Timer input XOR function . . . . . 666
20.3.18 Timers and external trigger synchronization . . . . . 667
20.3.19 Timer synchronization . . . . . 670
20.3.20 DMA burst mode . . . . . 674
20.3.21 Debug mode . . . . . 675
20.4 TIM2 registers . . . . . 676
20.4.1 TIM2 control register 1 (TIM2_CR1) . . . . . 676
20.4.2 TIM2 control register 2 (TIM2_CR2) . . . . . 677
20.4.3 TIM2 slave mode control register (TIM2_SMCR) . . . . . 679
20.4.4 TIM2 DMA/Interrupt enable register (TIM2_DIER) . . . . . 682
20.4.5 TIM2 status register (TIM2_SR) . . . . . 683
20.4.6 TIM2 event generation register (TIM2_EGR) . . . . . 685
20.4.7 TIM2 capture/compare mode register 1 (TIM2_CCMR1) . . . . . 686
20.4.8 TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1) . . 688
20.4.9 TIM2 capture/compare mode register 2 (TIM2_CCMR2) . . . . . 690
20.4.10 TIM2 capture/compare mode register 2 [alternate] (TIM2_CCMR2) . . 691
20.4.11 TIM2 capture/compare enable register (TIM2_CCER) . . . . . 692
20.4.12 TIM2 counter (TIM2_CNT) . . . . . 693
20.4.13 TIM2 counter [alternate] (TIM2_CNT) . . . . . 694
20.4.14 TIM2 prescaler (TIM2_PSC) . . . . . 694
20.4.15 TIM2 auto-reload register (TIM2_ARR) . . . . . 695
20.4.16 TIM2 capture/compare register 1 (TIM2_CCR1) . . . . . 695

20.4.17 TIM2 capture/compare register 2 (TIM2_CCR2) . . . . . 695
20.4.18 TIM2 capture/compare register 3 (TIM2_CCR3) . . . . . 696
20.4.19 TIM2 capture/compare register 4 (TIM2_CCR4) . . . . . 696
20.4.20 TIM2 DMA control register (TIM2_DCR) . . . . . 697
20.4.21 TIM2 DMA address for full transfer (TIM2_DMAR) . . . . . 698
20.4.22 TIM2 option register 1 (TIM2_OR1) . . . . . 698
20.4.23 TIM2 alternate function option register 1 (TIM2_AF1) . . . . . 698
20.4.24 TIM2 timer input selection register (TIM2_TISEL) . . . . . 699
20.4.25 TIMx register map . . . . . 700

**21 General-purpose timers (TIM16/TIM17) . . . . . 703**

21.1 TIM16/TIM17 introduction . . . . . 703
21.2 TIM16/TIM17 main features . . . . . 703
21.3 TIM16/TIM17 functional description . . . . . 705
21.3.1 Time-base unit . . . . . 705
21.3.2 Counter modes . . . . . 707
21.3.3 Repetition counter . . . . . 711
21.3.4 Clock selection . . . . . 712
21.3.5 Capture/compare channels . . . . . 714
21.3.6 Input capture mode . . . . . 716
21.3.7 Forced output mode . . . . . 717
21.3.8 Output compare mode . . . . . 717
21.3.9 PWM mode . . . . . 719
21.3.10 Complementary outputs and dead-time insertion . . . . . 720
21.3.11 Using the break function . . . . . 722
21.3.12 Bidirectional break inputs . . . . . 725
21.3.13 6-step PWM generation . . . . . 726
21.3.14 One-pulse mode . . . . . 728
21.3.15 UIF bit remapping . . . . . 729
21.3.16 Slave mode – combined reset + trigger mode . . . . . 729
21.3.17 DMA burst mode . . . . . 729
21.3.18 Using timer output as trigger for other timers (TIM16/TIM17) . . . . . 730
21.3.19 Debug mode . . . . . 731
21.4 TIM16/TIM17 registers . . . . . 732
21.4.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) . . . . . 732
21.4.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) . . . . . 733
21.4.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) . . . . . 734

21.4.4 TIMx status register (TIMx_SR)(x = 16 to 17) . . . . . 735
21.4.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) . . . . . 736
21.4.6 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 16 to 17) . . . . . 737
21.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) . . . . . 738
21.4.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) . . 740
21.4.9 TIMx counter (TIMx_CNT)(x = 16 to 17) . . . . . 742
21.4.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) . . . . . 743
21.4.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17) . . . . . 743
21.4.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) . . . . . 744
21.4.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) . . . . . 744
21.4.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) . . . . . 745
21.4.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) . . . . . 747
21.4.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) . . . . . 748
21.4.17 TIM16 option register 1 (TIM16_OR1) . . . . . 749
21.4.18 TIM16 alternate function register 1 (TIM16_AF1) . . . . . 749
21.4.19 TIM16 input selection register (TIM16_TISEL) . . . . . 750
21.4.20 TIM17 option register 1 (TIM17_OR1) . . . . . 750
21.4.21 TIM17 alternate function register 1 (TIM17_AF1) . . . . . 750
21.4.22 TIM17 input selection register (TIM17_TISEL) . . . . . 751
21.4.23 TIM16/TIM17 register map . . . . . 752

**22 Low-power timer (LPTIM) . . . . . 754**
22.1 Introduction . . . . . 754
22.2 LPTIM main features . . . . . 754
22.3 LPTIM implementation . . . . . 755
22.4 LPTIM functional description . . . . . 755
22.4.1 LPTIM block diagram . . . . . 755
22.4.2 LPTIM pins and internal signals . . . . . 756
22.4.3 LPTIM trigger mapping . . . . . 756
22.4.4 LPTIM reset and clocks . . . . . 757
22.4.5 Glitch filter . . . . . 757
22.4.6 Prescaler . . . . . 758
22.4.7 Trigger multiplexer . . . . . 758
22.4.8 Operating mode . . . . . 759
22.4.9 Timeout function . . . . . 761

22.4.10 Waveform generation . . . 761
22.4.11 Register update . . . 762
22.4.12 Counter mode . . . 763
22.4.13 Timer enable . . . 763
22.4.14 Timer counter reset . . . 764
22.4.15 Encoder mode . . . 764
22.4.16 Debug mode . . . 766
22.5 LPTIM low-power modes . . . 766
22.6 LPTIM interrupts . . . 767
22.7 LPTIM registers . . . 767
22.7.1 LPTIM interrupt and status register (LPTIM_ISR) . . . 768
22.7.2 LPTIM interrupt clear register (LPTIM_ICR) . . . 769
22.7.3 LPTIM interrupt enable register (LPTIM_IER) . . . 769
22.7.4 LPTIM configuration register (LPTIM_CFGR) . . . 770
22.7.5 LPTIM control register (LPTIM_CR) . . . 773
22.7.6 LPTIM compare register (LPTIM_CMP) . . . 775
22.7.7 LPTIM autoreload register (LPTIM_ARR) . . . 775
22.7.8 LPTIM counter register (LPTIM_CNT) . . . 776
22.7.9 LPTIM register map . . . 777

**23 Infrared interface (IRTIM) . . . 778**

**24 Real-time clock (RTC) . . . 779**
24.1 Introduction . . . 779
24.2 RTC main features . . . 780
24.3 RTC functional description . . . 781
24.3.1 RTC block diagram . . . 781
24.3.2 Clock and prescalers . . . 782
24.3.3 Real-time clock and calendar . . . 782
24.3.4 Programmable alarms . . . 783
24.3.5 Periodic auto-wake-up . . . 783
24.3.6 RTC initialization and configuration . . . 784
24.3.7 Reading the calendar . . . 785
24.3.8 Resetting the RTC . . . 786
24.3.9 RTC synchronization . . . 787
24.3.10 RTC reference clock detection . . . 787
24.3.11 RTC smooth digital calibration . . . 788

24.3.12 Time-stamp function . . . . . 790
24.3.13 Tamper detection . . . . . 791
24.3.14 Calibration clock output . . . . . 793
24.3.15 Alarm output . . . . . 793
24.4 RTC low-power modes . . . . . 794
24.5 RTC interrupts . . . . . 794
24.6 RTC registers . . . . . 795
24.6.1 RTC time register (RTC_TR) . . . . . 795
24.6.2 RTC date register (RTC_DR) . . . . . 796
24.6.3 RTC control register (RTC_CR) . . . . . 797
24.6.4 RTC initialization and status register (RTC_ISR) . . . . . 800
24.6.5 RTC prescaler register (RTC_PRER) . . . . . 803
24.6.6 RTC wake-up timer register (RTC_WUTR) . . . . . 804
24.6.7 RTC alarm A register (RTC_ALRMAR) . . . . . 805
24.6.8 RTC alarm B register (RTC_ALRMBR) . . . . . 806
24.6.9 RTC write protection register (RTC_WPR) . . . . . 807
24.6.10 RTC sub second register (RTC_SSR) . . . . . 807
24.6.11 RTC shift control register (RTC_SHIFTR) . . . . . 808
24.6.12 RTC timestamp time register (RTC_TSTR) . . . . . 809
24.6.13 RTC timestamp date register (RTC_TSDR) . . . . . 810
24.6.14 RTC time-stamp sub second register (RTC_TSSSR) . . . . . 811
24.6.15 RTC calibration register (RTC_CALR) . . . . . 812
24.6.16 RTC tamper configuration register (RTC_TAMPCR) . . . . . 813
24.6.17 RTC alarm A sub second register (RTC_ALRMASSR) . . . . . 815
24.6.18 RTC alarm B sub second register (RTC_ALRMBSSR) . . . . . 816
24.6.19 RTC option register (RTC_OR) . . . . . 817
24.6.20 RTC backup registers (RTC_BKPxR) . . . . . 817
24.6.21 RTC register map . . . . . 818

**25 Independent watchdog (IWDG) . . . . . 820**
25.1 Introduction . . . . . 820
25.2 IWDG main features . . . . . 820
25.3 IWDG functional description . . . . . 820
25.3.1 IWDG block diagram . . . . . 820
25.3.2 Window option . . . . . 821
25.3.3 Hardware watchdog . . . . . 822

25.3.4 Low-power freeze . . . 822
25.3.5 Register access protection . . . 822
25.3.6 Debug mode . . . 822
25.4 IWDG registers . . . 823
25.4.1 IWDG key register (IWDG_KR) . . . 823
25.4.2 IWDG prescaler register (IWDG_PR) . . . 824
25.4.3 IWDG reload register (IWDG_RLR) . . . 825
25.4.4 IWDG status register (IWDG_SR) . . . 826
25.4.5 IWDG window register (IWDG_WINR) . . . 827
25.4.6 IWDG register map . . . 828

**26 System window watchdog (WWDG) . . . 829**
26.1 Introduction . . . 829
26.2 WWDG main features . . . 829
26.3 WWDG functional description . . . 829
26.3.1 WWDG block diagram . . . 830
26.3.2 Enabling the watchdog . . . 830
26.3.3 Controlling the down-counter . . . 830
26.3.4 How to program the watchdog timeout . . . 830
26.3.5 Debug mode . . . 832
26.4 WWDG interrupts . . . 832
26.5 WWDG registers . . . 832
26.5.1 WWDG control register (WWDG_CR) . . . 832
26.5.2 WWDG configuration register (WWDG_CFR) . . . 833
26.5.3 WWDG status register (WWDG_SR) . . . 834
26.5.4 WWDG register map . . . 834

**27 Inter-integrated circuit (I2C) interface . . . 835**
27.1 Introduction . . . 835
27.2 I2C main features . . . 835
27.3 I2C implementation . . . 836
27.4 I2C functional description . . . 836
27.4.1 I2C block diagram . . . 837
27.4.2 I2C pins and internal signals . . . 838
27.4.3 I2C clock requirements . . . 838
27.4.4 Mode selection . . . 838

27.4.5 I2C initialization . . . . . 839
27.4.6 Software reset . . . . . 843
27.4.7 Data transfer . . . . . 844
27.4.8 I2C slave mode . . . . . 846
27.4.9 I2C master mode . . . . . 855
27.4.10 I2C_TIMINGR register configuration examples . . . . . 866
27.4.11 SMBus specific features . . . . . 867
27.4.12 SMBus initialization . . . . . 870
27.4.13 SMBus: I2C_TIMEOUTR register configuration examples . . . . . 872
27.4.14 SMBus slave mode . . . . . 872
27.4.15 Wake-up from Stop mode on address match . . . . . 879
27.4.16 Error conditions . . . . . 880
27.4.17 DMA requests . . . . . 882
27.4.18 Debug mode . . . . . 882
27.5 I2C low-power modes . . . . . 883
27.6 I2C interrupts . . . . . 884
27.7 I2C registers . . . . . 885
27.7.1 I2C control register 1 (I2C_CR1) . . . . . 885
27.7.2 I2C control register 2 (I2C_CR2) . . . . . 887
27.7.3 I2C own address 1 register (I2C_OAR1) . . . . . 889
27.7.4 I2C own address 2 register (I2C_OAR2) . . . . . 890
27.7.5 I2C timing register (I2C_TIMINGR) . . . . . 891
27.7.6 I2C timeout register (I2C_TIMEOUTR) . . . . . 892
27.7.7 I2C interrupt and status register (I2C_ISR) . . . . . 893
27.7.8 I2C interrupt clear register (I2C_ICR) . . . . . 895
27.7.9 I2C PEC register (I2C_PECR) . . . . . 896
27.7.10 I2C receive data register (I2C_RXDR) . . . . . 897
27.7.11 I2C transmit data register (I2C_TXDR) . . . . . 897
27.7.12 I2C register map . . . . . 898

**28 Universal synchronous/asynchronous receiver transmitter (USART/UART) . . . . . 900**
28.1 USART introduction . . . . . 900
28.2 USART main features . . . . . 901
28.3 USART extended features . . . . . 902
28.4 USART implementation . . . . . 902

28.5 USART functional description . . . . . 903
28.5.1 USART block diagram . . . . . 903
28.5.2 USART signals . . . . . 904
28.5.3 USART character description . . . . . 905
28.5.4 USART FIFOs and thresholds . . . . . 907
28.5.5 USART transmitter . . . . . 907
28.5.6 USART receiver . . . . . 911
28.5.7 USART baud rate generation . . . . . 918
28.5.8 Tolerance of the USART receiver to clock deviation . . . . . 919
28.5.9 USART auto baud rate detection . . . . . 921
28.5.10 USART multiprocessor communication . . . . . 923
28.5.11 USART Modbus communication . . . . . 925
28.5.12 USART parity control . . . . . 926
28.5.13 USART LIN (local interconnection network) mode . . . . . 927
28.5.14 USART synchronous mode . . . . . 929
28.5.15 USART single-wire half-duplex communication . . . . . 933
28.5.16 USART receiver timeout . . . . . 933
28.5.17 USART smartcard mode . . . . . 934
28.5.18 USART IrDA SIR ENDEC block . . . . . 938
28.5.19 Continuous communication using USART and DMA . . . . . 941
28.5.20 RS232 hardware flow control and RS485 Driver Enable . . . . . 943
28.5.21 USART low-power management . . . . . 946
28.6 USART in low-power modes . . . . . 949
28.7 USART interrupts . . . . . 950
28.8 USART registers . . . . . 951
28.8.1 USART control register 1 (USART_CR1) . . . . . 951
28.8.2 USART control register 1 [alternate] (USART_CR1) . . . . . 955
28.8.3 USART control register 2 (USART_CR2) . . . . . 958
28.8.4 USART control register 3 (USART_CR3) . . . . . 962
28.8.5 USART baud rate register (USART_BRR) . . . . . 967
28.8.6 USART guard time and prescaler register (USART_GTPR) . . . . . 967
28.8.7 USART receiver timeout register (USART_RTOR) . . . . . 968
28.8.8 USART request register (USART_RQR) . . . . . 969
28.8.9 USART interrupt and status register (USART_ISR) . . . . . 970
28.8.10 USART interrupt and status register [alternate] (USART_ISR) . . . . . 976
28.8.11 USART interrupt flag clear register (USART_ICR) . . . . . 981
28.8.12 USART receive data register (USART_RDR) . . . . . 983

28.8.13 USART transmit data register (USART_TDR) . . . . . 983
28.8.14 USART prescaler register (USART_PRESC) . . . . . 984
28.8.15 USART register map . . . . . 985

**29 Serial peripheral interface (SPI) . . . . . 987**
29.1 Introduction . . . . . 987
29.2 SPI main features . . . . . 987
29.3 SPI implementation . . . . . 988
29.4 SPI functional description . . . . . 988
29.4.1 General description . . . . . 988
29.4.2 Communications between one master and one slave . . . . . 989
29.4.3 Standard multislave communication . . . . . 991
29.4.4 Multimaster communication . . . . . 992
29.4.5 Slave select (NSS) pin management . . . . . 993
29.4.6 Communication formats . . . . . 994
29.4.7 Configuration of SPI . . . . . 996
29.4.8 Procedure for enabling SPI . . . . . 997
29.4.9 Data transmission and reception procedures . . . . . 997
29.4.10 SPI status flags . . . . . 1007
29.4.11 SPI error flags . . . . . 1008
29.4.12 NSS pulse mode . . . . . 1009
29.4.13 TI mode . . . . . 1009
29.4.14 CRC calculation . . . . . 1010
29.5 SPI interrupts . . . . . 1012
29.6 SPI registers . . . . . 1013
29.6.1 SPI control register 1 (SPIx_CR1) . . . . . 1013
29.6.2 SPI control register 2 (SPIx_CR2) . . . . . 1015
29.6.3 SPI status register (SPIx_SR) . . . . . 1017
29.6.4 SPI data register (SPIx_DR) . . . . . 1018
29.6.5 SPI CRC polynomial register (SPIx_CRCPR) . . . . . 1019
29.6.6 SPI Rx CRC register (SPIx_RXCRCR) . . . . . 1019
29.6.7 SPI Tx CRC register (SPIx_TXCRCR) . . . . . 1019
29.6.8 SPI register map . . . . . 1021

**30 Inter-processor communication controller (IPCC) . . . . . 1022**
30.1 Introduction . . . . . 1022

30.2 IPCC main features . . . . . 1022
30.3 IPCC functional description . . . . . 1022
30.3.1 IPCC block diagram . . . . . 1023
30.3.2 IPCC Simplex channel mode . . . . . 1023
30.3.3 IPCC Half-duplex channel mode . . . . . 1026
30.3.4 IPCC interrupts . . . . . 1029
30.4 IPCC registers . . . . . 1030
30.4.1 IPCC processor 1 control register (IPCC_C1CR) . . . . . 1030
30.4.2 IPCC processor 1 mask register (IPCC_C1MR) . . . . . 1030
30.4.3 IPCC processor 1 status set clear register (IPCC_C1SCR) . . . . . 1031
30.4.4 IPCC processor 1 to processor 2 status register (IPCC_C1TOC2SR) . . . . . 1032
30.4.5 IPCC processor 2 control register (IPCC_C2CR) . . . . . 1032
30.4.6 IPCC processor 2 mask register (IPCC_C2MR) . . . . . 1033
30.4.7 IPCC processor 2 status set clear register (IPCC_C2SCR) . . . . . 1033
30.4.8 IPCC processor 2 to processor 1 status register (IPCC_C2TOC1SR) . . . . . 1034
30.4.9 IPCC register map . . . . . 1035

**31 Hardware semaphore (HSEM) . . . . . 1036**
31.1 Introduction . . . . . 1036
31.2 Main features . . . . . 1036
31.3 Functional description . . . . . 1037
31.3.1 HSEM block diagram . . . . . 1037
31.3.2 HSEM internal signals . . . . . 1037
31.3.3 HSEM lock procedures . . . . . 1037
31.3.4 HSEM write/read/read lock register address . . . . . 1039
31.3.5 HSEM unlock procedures . . . . . 1039
31.3.6 HSEM COREID semaphore clear . . . . . 1040
31.3.7 HSEM interrupts . . . . . 1040
31.3.8 AHB bus master ID verification . . . . . 1042
31.4 HSEM registers . . . . . 1043
31.4.1 HSEM register semaphore x (HSEM_Rx) . . . . . 1043
31.4.2 HSEM read lock register semaphore x (HSEM_RLRx) . . . . . 1044
31.4.3 HSEM interrupt enable register (HSEM_CnIER) . . . . . 1045
31.4.4 HSEM interrupt clear register (HSEM_CnICR) . . . . . 1045
31.4.5 HSEM interrupt status register (HSEM_CnISR) . . . . . 1045

31.4.6 HSEM interrupt status register (HSEM_CnMISR) . . . . . 1046
31.4.7 HSEM clear register (HSEM_CR) . . . . . 1046
31.4.8 HSEM clear semaphore key register (HSEM_KEYR) . . . . . 1047
31.4.9 HSEM register map . . . . . 1048

**32 Debug support (DBG) . . . . . 1050**
32.1 Introduction . . . . . 1050
32.2 Debug use cases . . . . . 1050
32.3 DBG functional description . . . . . 1052
32.3.1 DBG block diagram . . . . . 1052
32.3.2 DBG pins and internal signals . . . . . 1052
32.3.3 DBG power domains . . . . . 1053
32.3.4 DBG clocks . . . . . 1053
32.3.5 Debug and low power modes . . . . . 1053
32.3.6 DBG reset . . . . . 1053
32.4 Serial wire and JTAG debug port (SWJ-DP) . . . . . 1053
32.4.1 JTAG debug port . . . . . 1054
32.4.2 SW debug port . . . . . 1057
32.4.3 Debug port registers . . . . . 1058
32.4.4 DP debug port identification register (DP_DPIDR) . . . . . 1059
32.4.5 DP abort register (DP_ABORTR) . . . . . 1059
32.4.6 DP control and status register (DP_CTRL/STATR) . . . . . 1060
32.4.7 DP data link control register (DP_DLCR) . . . . . 1062
32.4.8 DP target identification register (DP_TARGETIDR) . . . . . 1062
32.4.9 DP data link protocol identification register (DP_DLPIDR) . . . . . 1063
32.4.10 DP resend register (DP_RESENDR) . . . . . 1063
32.4.11 DP access port select register (DP_SELECTR) . . . . . 1064
32.4.12 DP read buffer register (DP_BUFFR) . . . . . 1064
32.4.13 DP target selection register (DP_TARGETSELR) . . . . . 1065
32.4.14 Debug port register map and reset values . . . . . 1066
32.5 Access ports . . . . . 1067
32.5.1 AP control/status word register (AP_CSWR) . . . . . 1069
32.5.2 AP transfer address register (AP_TAR) . . . . . 1070
32.5.3 AP data read/write register (AP_DRWR) . . . . . 1070
32.5.4 AP banked data registers (AP_BD0-3R) . . . . . 1070
32.5.5 AP base address register (AP_BASER) . . . . . 1071
32.5.6 AP identification register (AP_IDR) . . . . . 1071

32.5.7 Access port register map and reset values . . . . . 1073
32.6 Cross trigger interface (CTI) and matrix (CTM) . . . . . 1074
32.7 Cross trigger interface registers . . . . . 1078
32.7.1 CTI control register (CTI_CONTROLR) . . . . . 1078
32.7.2 CTI trigger acknowledge register (CTI_INTACKR) . . . . . 1078
32.7.3 CTI application trigger set register (CTI_APPSETR) . . . . . 1078
32.7.4 CTI application trigger clear register (CTI_APPCLEAR) . . . . . 1079
32.7.5 CTI application pulse register (CTI_APPPULSER) . . . . . 1080
32.7.6 CTI trigger In x enable register (CTI_INENRx) . . . . . 1080
32.7.7 CTI trigger out x enable register (CTI_OUTENRx) . . . . . 1081
32.7.8 CTI trigger in status register (CTI_TRGISTSR) . . . . . 1081
32.7.9 CTI trigger out status register (CTI_TRGOSTSR) . . . . . 1082
32.7.10 CTI channel in status register (CTI_CHINSTSR) . . . . . 1082
32.7.11 CTI channel out status register (CTI_CHOUTSTSR) . . . . . 1082
32.7.12 CTI channel gate register (CTI_GATER) . . . . . 1083
32.7.13 CTI claim tag set register (CTI_CLAIMSETR) . . . . . 1083
32.7.14 CTI claim tag clear register (CTI_CLAIMCLR) . . . . . 1084
32.7.15 CTI lock access register (CTI_LAR) . . . . . 1084
32.7.16 CTI lock status register (CTI_LSR) . . . . . 1085
32.7.17 CTI authentication status register (CTI_AUTHSTATR) . . . . . 1085
32.7.18 CTI device configuration register (CTI_DEVIDR) . . . . . 1086
32.7.19 CTI device type identifier register (CTI_DEVTYPER) . . . . . 1086
32.7.20 CTI CoreSight peripheral identity register 4 (CTI_PIDR4) . . . . . 1087
32.7.21 CTI CoreSight peripheral identity register 0 (CTI_PIDR0) . . . . . 1087
32.7.22 CTI CoreSight peripheral identity register 1 (CTI_PIDR1) . . . . . 1087
32.7.23 CTI CoreSight peripheral identity register 2 (CTI_PIDR2) . . . . . 1088
32.7.24 CTI CoreSight peripheral identity register 3 (CTI_PIDR3) . . . . . 1088
32.7.25 CTI CoreSight component identity register 0 (CTI_CIDR0) . . . . . 1089
32.7.26 CTI CoreSight peripheral identity register 1 (CTI_CIDR1) . . . . . 1089
32.7.27 CTI CoreSight component identity register 2 (CTI_CIDR2) . . . . . 1090
32.7.28 CTI CoreSight component identity register 3 (CTI_CIDR3) . . . . . 1090
32.7.29 CTI register map and reset values . . . . . 1091
32.8 Microcontroller debug unit (DBGMCU) . . . . . 1094
32.8.1 DBGMCU identity code register (DBGMCU_IDCODE) . . . . . 1094
32.8.2 DBGMCU configuration register (DBGMCU_CR) . . . . . 1094
32.8.3 DBGMCU CPU1 APB1 peripheral freeze register 1 (DBGMCU_APB1FZR1) . . . . . 1095

32.8.4 DBGMCU CPU2 APB1 peripheral freeze register 1 (DBGMCU_C2APB1FZR1) . . . . . 1096
32.8.5 DBGMCU CPU1 APB1 peripheral freeze register 2 (DBGMCU_APB1FZR2) . . . . . 1097
32.8.6 DBGMCU CPU2 APB1 peripheral freeze register 2 (DBGMCU_C2APB1FZR2) . . . . . 1098
32.8.7 DBGMCU CPU1 APB2 peripheral freeze register (DBGMCU_APB2FZR) . . . . . 1098
32.8.8 DBGMCU CPU2 APB2 peripheral freeze register (DBGMCU_C2APB2FZR) . . . . . 1099
32.8.9 DBGMCU register map and reset values . . . . . 1100
32.9 CPU2 ROM tables . . . . . .1102
32.9.1 CPU2 ROM1 memory type register (C2ROM1_MEMTYPER) . . . . . 1104
32.9.2 CPU2 ROM1 CoreSight peripheral identity register 4 (C2ROM1_PIDR4) . . . . . 1104
32.9.3 CPU2 ROM1 CoreSight peripheral identity register 0 (C2ROM1_PIDR0) . . . . . 1104
32.9.4 CPU2 ROM1 CoreSight peripheral identity register 1 (C2ROM1_PIDR1) . . . . . 1105
32.9.5 CPU2 ROM1 CoreSight peripheral identity register 2 (C2ROM1_PIDR2) . . . . . 1105
32.9.6 CPU2 ROM1 CoreSight peripheral identity register 3 (C2ROM1_PIDR3) . . . . . 1106
32.9.7 CPU2 ROM1 CoreSight component identity register 0 (C2ROM1_CIDR0) . . . . . 1106
32.9.8 CPU2 ROM1 CoreSight peripheral identity register 1 (C2ROM1_CIDR1) . . . . . 1107
32.9.9 CPU2 ROM1 CoreSight component identity register 2 (C2ROM1_CIDR2) . . . . . 1107
32.9.10 CPU2 ROM1 CoreSight component identity register 3 (C2ROM1_CIDR3) . . . . . 1107
32.9.11 CPU2 processor ROM table registers and reset values . . . . . 1109
32.9.12 CPU2 ROM2 memory type register (C2ROM2_MEMTYPER) . . . . . 1110
32.9.13 CPU2 ROM2 CoreSight peripheral identity register 4 (C2ROM2_PIDR4) . . . . . 1110
32.9.14 CPU2 ROM2 CoreSight peripheral identity register 0 (C2ROM2_PIDR0) . . . . . 1110
32.9.15 CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2_PIDR1) . . . . . 1111
32.9.16 CPU2 ROM2 CoreSight peripheral identity register 2 (C2ROM2_PIDR2) . . . . . 1111

32.9.17 CPU2 ROM2 CoreSight peripheral identity register 3 (C2ROM2_PIDR3) . . . . . 1112
32.9.18 CPU2 ROM2 CoreSight component identity register 0 (C2ROM2_CIDR0) . . . . . 1112
32.9.19 CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2_CIDR1) . . . . . 1113
32.9.20 CPU2 ROM2 CoreSight component identity register 2 (C2ROM2_CIDR2) . . . . . 1113
32.9.21 CPU2 ROM2 CoreSight component identity register 3 (C2ROM2_CIDR3) . . . . . 1113
32.9.22 CPU2 ROM table register map and reset values . . . . . 1115
32.10 CPU2 data watchpoint and trace unit (DWT) . . . . . 1116
32.10.1 DWT control register (DWT_CTRLR) . . . . . 1116
32.10.2 DWT cycle count register (DWT_CYCCNTR) . . . . . 1118
32.10.3 DWT CPI count register (DWT_CPICNTR) . . . . . 1118
32.10.4 DWT exception count register (DWT_EXCCNTR) . . . . . 1119
32.10.5 DWT sleep count register (DWT_SLPCNTR) . . . . . 1119
32.10.6 DWT LSU count register (DWT_LSUCNTR) . . . . . 1120
32.10.7 DWT fold count register (DWT_FOLDCNTR) . . . . . 1120
32.10.8 DWT program counter sample register (DWT_PCSR) . . . . . 1120
32.10.9 DWT comparator register x (DWT_COMPxR) . . . . . 1121
32.10.10 DWT mask register x (DWT_MASKxR) . . . . . 1121
32.10.11 DWT function register x (DWT_FUNCTxR) . . . . . 1121
32.10.12 DWT CoreSight peripheral identity register 4 (DWT_PIDR4) . . . . . 1122
32.10.13 DWT CoreSight peripheral identity register 0 (DWT_PIDR0) . . . . . 1123
32.10.14 DWT CoreSight peripheral identity register 1 (DWT_PIDR1) . . . . . 1123
32.10.15 DWT CoreSight peripheral identity register 2 (DWT_PIDR2) . . . . . 1124
32.10.16 DWT CoreSight peripheral identity register 3 (DWT_PIDR3) . . . . . 1124
32.10.17 DWT CoreSight component identity register 0 (DWT_CIDR0) . . . . . 1125
32.10.18 DWT CoreSight peripheral identity register 1 (DWT_CIDR1) . . . . . 1125
32.10.19 DWT CoreSight component identity register 2 (DWT_CIDR2) . . . . . 1125
32.10.20 DWT CoreSight component identity register 3 (DWT_CIDR3) . . . . . 1126
32.10.21 CPU2 DWT registers . . . . . 1127
32.11 CPU2 breakpoint unit (PBU) . . . . . 1130
32.11.1 BPU control register (BPU_CTRLR) . . . . . 1130
32.11.2 BPU remap register (BPU_REMAPR) . . . . . 1130
32.11.3 BPU comparator registers (BPU_COMPxR) . . . . . 1131
32.11.4 BPU CoreSight peripheral identity register 4 (BPU_PIDR4) . . . . . 1131

32.11.5 BPU CoreSight peripheral identity register 0 (BPU_PIDR0) . . . . . 1132
32.11.6 BPU CoreSight peripheral identity register 1 (BPU_PIDR1) . . . . . 1132
32.11.7 BPU CoreSight peripheral identity register 2 (BPU_PIDR2) . . . . . 1132
32.11.8 BPU CoreSight peripheral identity register 3 (BPU_PIDR3) . . . . . 1133
32.11.9 BPU CoreSight component identity register 0 (BPU_CIDR0) . . . . . 1133
32.11.10 BPU CoreSight peripheral identity register 1 (BPU_CIDR1) . . . . . 1134
32.11.11 BPU CoreSight component identity register 2 (BPU_CIDR2) . . . . . 1134
32.11.12 BPU CoreSight component identity register 3 (BPU_CIDR3) . . . . . 1135
32.11.13 CPU2 BPU register map and reset values . . . . . 1136
32.12 CPU2 cross trigger interface (CTI) . . . . . .1137
32.13 CPU1 ROM table . . . . . .1137
32.13.1 CPU1 ROM memory type register (C1ROM_MEMTYPER) . . . . . 1138
32.13.2 CPU1 ROM CoreSight peripheral identity register 4 (C1ROM_PIDR4) . . . . . 1139
32.13.3 CPU1 ROM CoreSight peripheral identity register 0 (C1ROM_PIDR0) . . . . . 1139
32.13.4 CPU1 ROM CoreSight peripheral identity register 1 (C1ROM_PIDR1) . . . . . 1140
32.13.5 CPU1 ROM CoreSight peripheral identity register 2 (C1ROM_PIDR2) . . . . . 1140
32.13.6 CPU1 ROM CoreSight peripheral identity register 3 (C1ROM_PIDR3) . . . . . 1141
32.13.7 CPU1 ROM CoreSight component identity register 0 (C1ROM_CIDR0) . . . . . 1141
32.13.8 CPU1 ROM CoreSight peripheral identity register 1 (C1ROM_CIDR1) . . . . . 1141
32.13.9 CPU1 ROM CoreSight component identity register 2 (C1ROM_CIDR2) . . . . . 1142
32.13.10 CPU1 ROM CoreSight component identity register 3 (C1ROM_CIDR3) . . . . . 1142
32.13.11 CPU1 ROM table register map and reset values . . . . . 1144
32.14 CPU1 data watchpoint and trace unit (DWT) . . . . . .1145
32.14.1 DWT control register (DWT_CTRLR) . . . . . 1145
32.14.2 DWT cycle count register (DWT_CYCCNTR) . . . . . 1147
32.14.3 DWT CPI count register (DWT_CPICNTR) . . . . . 1147
32.14.4 DWT exception count register (DWT_EXCCNTR) . . . . . 1148
32.14.5 DWT sleep count register (DWT_SLPCNTR) . . . . . 1148
32.14.6 DWT LSU count register (DWT_LSUCNTR) . . . . . 1149
32.14.7 DWT fold count register (DWT_FOLDCNTR) . . . . . 1149

32.14.8 DWT program counter sample register (DWT_PCSR) . . . . . 1149
32.14.9 DWT comparator register x (DWT_COMPxR) . . . . . 1150
32.14.10 DWT mask register x (DWT_MASKxR) . . . . . 1150
32.14.11 DWT function register x (DWT_FUNCTxR) . . . . . 1150
32.14.12 DWT CoreSight peripheral identity register 4 (DWT_PIDR4) . . . . . 1151
32.14.13 DWT CoreSight peripheral identity register 0 (DWT_PIDR0) . . . . . 1152
32.14.14 DWT CoreSight peripheral identity register 1 (DWT_PIDR1) . . . . . 1152
32.14.15 DWT CoreSight peripheral identity register 2 (DWT_PIDR2) . . . . . 1153
32.14.16 DWT CoreSight peripheral identity register 3 (DWT_PIDR3) . . . . . 1153
32.14.17 DWT CoreSight component identity register 0 (DWT_CIDR0) . . . . . 1154
32.14.18 DWT CoreSight peripheral identity register 1 (DWT_CIDR1) . . . . . 1154
32.14.19 DWT CoreSight component identity register 2 (DWT_CIDR2) . . . . . 1154
32.14.20 DWT CoreSight component identity register 3 (DWT_CIDR3) . . . . . 1155
32.14.21 CPU1 DWT register map and reset values . . . . . 1156
32.15 CPU1 instrumentation trace macrocell (ITM) . . . . . .1158
32.15.1 ITM stimulus register x (ITM_STIMRx) . . . . . 1158
32.15.2 ITM trace enable register (ITM_TER) . . . . . 1158
32.15.3 ITM trace privilege register (ITM_TPR) . . . . . 1159
32.15.4 ITM trace control register (ITM_TCR) . . . . . 1159
32.15.5 ITM CoreSight peripheral identity register 4 (ITM_PIDR4) . . . . . 1160
32.15.6 ITM CoreSight peripheral identity register 0 (ITM_PIDR0) . . . . . 1161
32.15.7 ITM CoreSight peripheral identity register 1 (ITM_PIDR1) . . . . . 1161
32.15.8 ITM CoreSight peripheral identity register 2 (ITM_PIDR2) . . . . . 1162
32.15.9 ITM CoreSight peripheral identity register 3 (ITM_PIDR3) . . . . . 1162
32.15.10 ITM CoreSight component identity register 0 (ITM_CIDR0) . . . . . 1163
32.15.11 ITM CoreSight peripheral identity register 1 (ITM_CIDR1) . . . . . 1163
32.15.12 ITM CoreSight component identity register 2 (ITM_CIDR2) . . . . . 1163
32.15.13 ITM CoreSight component identity register 3 (ITM_CIDR3) . . . . . 1164
32.15.14 ITM register map and reset values . . . . . 1165
32.16 CPU1 breakpoint unit (FPB) . . . . . .1166
32.16.1 FPB control register (FPB_CTRLR) . . . . . 1166
32.16.2 FPB remap register (FPB_REMAPR) . . . . . 1166
32.16.3 FPB comparator registers (FPB_COMPxR) . . . . . 1167
32.16.4 FPB CoreSight peripheral identity register 4 (FPB_PIDR4) . . . . . 1167
32.16.5 FPB CoreSight peripheral identity register 0 (FPB_PIDR0) . . . . . 1168
32.16.6 FPB CoreSight peripheral identity register 1 (FPB_PIDR1) . . . . . 1168
32.16.7 FPB CoreSight peripheral identity register 2 (FPB_PIDR2) . . . . . 1169

32.16.8 FPB CoreSight peripheral identity register 3 (FPB_PIDR3) . . . . . 1169
32.16.9 FPB CoreSight component identity register 0 (FPB_CIDR0) . . . . . 1170
32.16.10 FPB CoreSight peripheral identity register 1 (FPB_CIDR1) . . . . . 1170
32.16.11 FPB CoreSight component identity register 2 (FPB_CIDR2) . . . . . 1170
32.16.12 FPB CoreSight component identity register 3 (FPB_CIDR3) . . . . . 1171
32.16.13 FPB register map and reset values . . . . . 1172
32.17 CPU1 trace port interface unit (TPIU) . . . . . .1173
32.17.1 TPIU supported port size register (TPIU_SSPSR) . . . . . 1173
32.17.2 TPIU current port size register (TPIU_CSPSR) . . . . . 1173
32.17.3 TPIU asynchronous clock presecaler register (TPIU_ACPR) . . . . . 1174
32.17.4 TPIU selected pin protocol register (TPIU_SPPR) . . . . . 1174
32.17.5 TPIU formatter and flush status register (TPIU_FFSR) . . . . . 1175
32.17.6 TPIU formatter and flush control register (TPIU_FFCR) . . . . . 1175
32.17.7 TPIU formatter synchronization counter register (TPIU_FSCR) . . . . 1176
32.17.8 TPIU claim tag set register (TPIU_CLAIMSETR) . . . . . 1176
32.17.9 TPIU claim tag clear register (TPIU_CLAIMCLR) . . . . . 1177
32.17.10 TPIU device configuration register (TPIU_DEVIDR) . . . . . 1177
32.17.11 TPIU device type identifier register (TPIU_DEVTYPER) . . . . . 1178
32.17.12 TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4) . . . . . 1178
32.17.13 TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0) . . . . . 1179
32.17.14 TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1) . . . . . 1179
32.17.15 TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2) . . . . . 1180
32.17.16 TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3) . . . . . 1180
32.17.17 TPIU CoreSight component identity register 0 (TPIU_CIDR0) . . . . . 1181
32.17.18 TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1) . . . . . 1181
32.17.19 TPIU CoreSight component identity register 2 (TPIU_CIDR2) . . . . . 1181
32.17.20 TPIU CoreSight component identity register 3 (TPIU_CIDR3) . . . . . 1182
32.17.21 CPU1 TPIU register map and reset values . . . . . 1183
32.18 CPU1 cross trigger interface (CTI) . . . . . .1185
32.19 References . . . . . .1185

**33 Device electronic signature . . . . . 1186**
33.1 Unique device ID register (96 bits) . . . . . .1186
33.2 Memory size data register . . . . . .1187
33.2.1 Flash size data register . . . . . 1187
33.3 Package data register . . . . . .1187

33.4 Part number codification register . . . . . .1188

**34 Important security notice . . . . . 1189**

**35 Revision history . . . . . 1190**

# List of tables

Table 1. STM32WB50CG/30CE memory map and peripheral register boundary addresses. . . . . 58
Table 2. Boot modes. . . . . 61
Table 3. Flash memory - Single bank organization . . . . . 65
Table 4. Number of wait states vs, flash memory clock (HCLK4) frequency . . . . . 67
Table 5. Page erase overview . . . . . 72
Table 6. Mass erase overview . . . . . 73
Table 7. Errors in page-based row programming . . . . . 78
Table 8. Option bytes format . . . . . 79
Table 9. Option bytes organization . . . . . 79
Table 10. Option loading control . . . . . 87
Table 11. UID64 organization . . . . . 88
Table 12. Flash memory read protection status . . . . . 89
Table 13. RDP regression from Level 1 to Level 0 and memory erase . . . . . 91
Table 14. Access status vs. protection level and execution modes . . . . . 92
Table 17. Flash memory interrupt requests . . . . . 97
Table 18. Flash interface register map and reset values . . . . . 116
Table 19. CRC internal input/output signals . . . . . 121
Table 20. CRC register map and reset values . . . . . 126
Table 21. Sub-system low power wake-up sources . . . . . 135
Table 22. Low-power mode summary . . . . . 137
Table 23. Functionalities depending on system operating mode . . . . . 138
Table 24. Low-power run . . . . . 141
Table 25. CPU CSTOP wake-up vs. system operating mode . . . . . 142
Table 26. Sleep mode. . . . . 143
Table 27. Low-power sleep. . . . . 144
Table 28. Stop0 mode . . . . . 146
Table 29. Stop1 mode . . . . . 147
Table 30. Stop2 mode . . . . . 149
Table 31. Standby mode. . . . . 150
Table 32. Shutdown mode . . . . . 152
Table 33. PWR register map and reset values . . . . . 169
Table 34. Maximum clock source frequency . . . . . 182
Table 35. Peripheral clock enable . . . . . 188
Table 36. Single core Low power debug configurations . . . . . 189
Table 37. RCC register map and reset values . . . . . 242
Table 38. Port bit configuration table . . . . . 250
Table 39. GPIO register map and reset values . . . . . 266
Table 40. SYSCFG register map and reset values. . . . . 284
Table 41. Peripherals interconnect matrix . . . . . 286
Table 42. DMA1 implementation . . . . . 292
Table 43. DMA internal input/output signals . . . . . 293
Table 44. Programmable data width and endian behavior (when PINC = MINC = 1) . . . . . 298
Table 45. DMA interrupt requests. . . . . 300
Table 46. DMA register map and reset values . . . . . 308
Table 47. DMAMUX instantiation . . . . . 312
Table 48. DMAMUX: assignment of multiplexer inputs to resources . . . . . 313
Table 49. DMAMUX: assignment of trigger inputs to resources. . . . . 313
Table 50. DMAMUX: assignment of synchronization inputs to resources . . . . . 314

Table 51. DMAMUX signals . . . 316
Table 52. DMAMUX interrupts . . . 320
Table 53. DMAMUX register map and reset values . . . 325
Table 54. CPU1 vector table. . . . 329
Table 55. CPU2 vector table. . . . 332
Table 56. Wakeup interrupt table . . . 334
Table 57. EXTI pin overview. . . . 337
Table 58. EVG pin overview . . . 337
Table 59. EXTI event input configurations and register control . . . 339
Table 60. Masking functionality . . . 342
Table 61. EXTI register map sections. . . . 343
Table 62. EXTI register map and reset values . . . 352
Table 63. ADC internal input/output signals . . . 357
Table 64. ADC input/output pins. . . . 357
Table 65. Configuring the trigger polarity for regular external triggers . . . 373
Table 66. Configuring the trigger polarity for injected external triggers . . . 373
Table 67. ADC1 - External triggers for regular channels . . . 374
Table 68. ADC1 - External trigger for injected channels. . . . 374
Table 69. TSAR timings depending on resolution . . . 386
Table 70. Offset computation versus data resolution . . . 389
Table 71. Analog watchdog channel selection . . . 399
Table 72. Analog watchdog 1 comparison . . . 400
Table 73. Analog watchdog 2 and 3 comparison . . . 400
Table 74. Maximum output results versus N and M (gray cells indicate truncation). . . . 404
Table 75. Effect of low-power modes on the ADC . . . 412
Table 76. ADC interrupts . . . 413
Table 77. ADC register map and reset values . . . 442
Table 78. ADC register map and reset values (master and slave ADC common registers) offset = 0x300 . . . 444
Table 79. RNG internal input/output signals. . . . 446
Table 80. RNG interrupt requests. . . . 453
Table 81. RNG register map and reset map. . . . 456
Table 82. AES internal input/output signals . . . 458
Table 83. CTR mode initialization vector definition. . . . 475
Table 84. GCM last block definition . . . 477
Table 85. Initialization of AES_IVRx registers in GCM mode . . . 478
Table 86. Initialization of AES_IVRx registers in CCM mode . . . 485
Table 87. Key endianness in AES_KEYRx registers (128- or 256-bit key length) . . . 490
Table 88. AES interrupt requests . . . 493
Table 89. Processing latency for ECB, CBC and CTR . . . 493
Table 90. Processing latency for GCM and CCM (in clock cycles). . . . 494
Table 91. AES register map and reset values . . . 504
Table 92. Internal input/output signals . . . 507
Table 93. PKA integer arithmetic functions list . . . 508
Table 94. PKA prime field (Fp) elliptic curve functions list . . . 508
Table 95. Montgomery parameter computation . . . 513
Table 96. Modular addition . . . 514
Table 97. Modular subtraction . . . 514
Table 98. Montgomery multiplication . . . 515
Table 99. Modular exponentiation (normal mode) . . . 516
Table 100. Modular exponentiation (fast mode) . . . 516
Table 101. Modular inversion . . . 516

Table 102. Modular reduction . . . 517
Table 103. Arithmetic addition . . . 517
Table 104. Arithmetic subtraction . . . 517
Table 105. Arithmetic multiplication . . . 518
Table 106. Arithmetic comparison . . . 518
Table 107. CRT exponentiation . . . 519
Table 108. Point on elliptic curve Fp check . . . 520
Table 109. ECC Fp scalar multiplication . . . 520
Table 110. ECC Fp scalar multiplication (Fast Mode) . . . 521
Table 111. ECDSA sign - Inputs . . . 522
Table 112. ECDSA sign - Outputs . . . 522
Table 113. Extended ECDSA sign (extra outputs) . . . 523
Table 114. ECDSA verification (inputs) . . . 523
Table 115. ECDSA verification (outputs) . . . 523
Table 116. Family of supported curves for ECC operations . . . 524
Table 117. Modular exponentiation computation times . . . 526
Table 118. ECC scalar multiplication computation times . . . 526
Table 119. ECDSA signature average computation times . . . 526
Table 120. ECDSA verification average computation times . . . 527
Table 121. Point on elliptic curve Fp check average computation times . . . 527
Table 122. Montgomery parameters average computation times . . . 527
Table 123. PKA interrupt requests . . . 527
Table 124. PKA register map and reset values . . . 531
Table 125. Behavior of timer outputs versus BRK/BRK2 inputs . . . 574
Table 126. Break protection disarming conditions . . . 576
Table 127. Counting direction versus encoder signals . . . 582
Table 128. TIM1 internal trigger connection . . . 599
Table 129. Output control bits for complementary OCx and OCxN channels with break feature . . . 613
Table 130. TIM1 register map and reset values . . . 629
Table 131. Counting direction versus encoder signals . . . 665
Table 132. TIM2 internal trigger connection . . . 682
Table 133. Output control bit for standard OCx channels . . . 693
Table 134. TIM2 register map and reset values . . . 700
Table 135. Break protection disarming conditions . . . 725
Table 136. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) . . . 742
Table 137. TIM16/TIM17 register map and reset values . . . 752
Table 138. LPTIM features . . . 755
Table 139. LPTIM input/output pins . . . 756
Table 140. LPTIM internal signals . . . 756
Table 141. LPTIM1 external trigger connection . . . 756
Table 142. LPTIM2 external trigger connection . . . 756
Table 143. Prescaler division ratios . . . 758
Table 144. Encoder counting scenarios . . . 765
Table 145. Effect of low-power modes on the LPTIM . . . 766
Table 146. Interrupt events . . . 767
Table 147. LPTIM register map and reset values . . . 777
Table 148. Effect of low-power modes on RTC . . . 794
Table 149. Interrupt control bits . . . 794
Table 150. RTC register map and reset values . . . 818
Table 151. IWDG register map and reset values . . . 828
Table 152. WWDG register map and reset values . . . 834

Table 153. STM32WB50CG/30CE I2C implementation . . . 836
Table 154. I2C input/output pins . . . 838
Table 155. I2C internal input/output signals . . . 838
Table 156. Comparison of analog vs. digital filters . . . 840
Table 157. I2C-SMBus specification data setup and hold times . . . 842
Table 158. I2C configuration . . . 846
Table 159. I2C-SMBus specification clock timings . . . 857
Table 160. Examples of timing settings for fI2CCLK = 8 MHz . . . 867
Table 161. Examples of timing settings for fI2CCLK = 16 MHz . . . 867
Table 162. SMBus timeout specifications . . . 869
Table 163. SMBus with PEC configuration . . . 871
Table 164. Examples of TIMEOUTA settings (max $t_{TIMEOUT}$ = 25 ms) . . . 872
Table 165. Examples of TIMEOUTB settings . . . 872
Table 166. Examples of TIMEOUTA settings (max $t_{IDLE}$ = 50 µs) . . . 872
Table 167. Effect of low-power modes on the I2C . . . 883
Table 168. I2C interrupt requests . . . 884
Table 169. I2C register map and reset values . . . 898
Table 170. USART features . . . 902
Table 171. Noise detection from sampled data . . . 917
Table 172. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 920
Table 173. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 921
Table 174. USART frame formats . . . 926
Table 175. Effect of low-power modes on the USART . . . 949
Table 176. USART interrupt requests . . . 950
Table 177. USART register map and reset values . . . 985
Table 178. SPI implementation . . . 988
Table 179. SPI interrupt requests . . . 1012
Table 180. SPI register map and reset values . . . 1021
Table 181. IPCC interface signals . . . 1023
Table 182. Bits used for the communication . . . 1024
Table 183. IPCC register map and reset values . . . 1035
Table 184. HSEM internal input/output signals . . . 1037
Table 185. Authorized AHB bus master IDs . . . 1042
Table 186. HSEM register map and reset values . . . 1048
Table 187. JTAG/Serial-wire debug port pins . . . 1052
Table 188. Single Wire Trace port pins . . . 1052
Table 189. JTAG-DP data registers . . . 1056
Table 190. Packet request . . . 1057
Table 191. ACK response . . . 1058
Table 192. Data transfer . . . 1058
Table 193. Debug port register map and reset values . . . 1066
Table 194. Access port register map and reset values . . . 1073
Table 195. CPU2 CTI inputs . . . 1074
Table 196. CPU2 CTI outputs . . . 1075
Table 197. CPU1 CTI inputs . . . 1075
Table 198. CPU1 CTI outputs . . . 1075
Table 199. CTI register map and reset values . . . 1091
Table 200. DBGMCU register map and reset values . . . 1100
Table 201. CPU2 processor ROM table . . . 1102
Table 202. CPU2 ROM table . . . 1102
Table 203. CPU2 processor ROM table register map and reset values . . . 1109
Table 204. CPU2 ROM table register map and reset values . . . 1115

Table 205. CPU2 DWT register map and reset values . . . . . 1127
Table 206. CPU2 BPU register map and reset values . . . . . 1136
Table 207. CPU1 ROM table . . . . . 1137
Table 208. CPU1 ROM table register map and reset values . . . . . 1144
Table 209. CPU1 DWT register map and reset values . . . . . 1156
Table 210. CPU1 ITM register map and reset values . . . . . 1165
Table 211. CPU1 FPB register map and reset values . . . . . 1172
Table 212. CPU1 TPIU register map and reset values . . . . . 1183
Table 213. Document revision history . . . . . 1190

# List of figures

Figure 1. System architecture . . . 54
Figure 2. Memory map . . . 57
Figure 3. Sequential 16-bit instructions execution . . . 69
Figure 4. Changing the Read protection (RDP) level . . . 92
Figure 5. Radio system block diagram . . . 119
Figure 6. CRC calculation unit block diagram . . . 121
Figure 7. Power supply overview . . . 128
Figure 8. Brown-out reset waveform . . . 131
Figure 9. PVD thresholds . . . 132
Figure 10. CPU2 boot options . . . 133
Figure 11. Low-power modes possible transitions . . . 136
Figure 12. Real-time radio activity flags . . . 153
Figure 13. Simplified diagram of the reset circuit . . . 172
Figure 14. Clock tree . . . 176
Figure 15. HSE clock sources . . . 177
Figure 16. LSE clock sources . . . 180
Figure 17. Frequency measurement with TIM16 in capture mode . . . 186
Figure 18. Frequency measurement with TIM17 in capture mode . . . 186
Figure 19. Three-volt or Five-volt tolerant GPIO structure (TT or FT) . . . 249
Figure 20. Input floating / pull up / pull down configurations . . . 254
Figure 21. Output configuration . . . 254
Figure 22. Alternate function configuration . . . 255
Figure 23. High impedance-analog configuration . . . 256
Figure 24. DMA block diagram . . . 292
Figure 25. DMAMUX block diagram . . . 315
Figure 26. Synchronization mode of the DMAMUX request line multiplexer channel . . . 318
Figure 27. Event generation of the DMA request line multiplexer channel . . . 318
Figure 28. Interrupt block diagram . . . 328
Figure 29. EXTI block diagram . . . 337
Figure 30. Configurable event trigger logic CPU wakeup . . . 340
Figure 31. Direct event trigger logic CPU wakeup . . . 341
Figure 32. ADC block diagram . . . 356
Figure 33. ADC clock scheme . . . 359
Figure 34. ADC1 connectivity . . . 360
Figure 35. ADC calibration . . . 363
Figure 36. Updating the ADC calibration factor . . . 364
Figure 37. Mixing single-ended and differential channels . . . 364
Figure 38. Enabling / disabling the ADC . . . 366
Figure 39. Analog to digital conversion time . . . 371
Figure 40. Stopping ongoing regular conversions . . . 372
Figure 41. Stopping ongoing regular and injected conversions . . . 372
Figure 42. Injected conversion latency . . . 376
Figure 43. Example of JSQR queue of context (sequence change) . . . 379
Figure 44. Example of JSQR queue of context (trigger change) . . . 379
Figure 45. Example of JSQR queue of context with overflow before conversion . . . 380
Figure 46. Example of JSQR queue of context with overflow during conversion . . . 380
Figure 47. Example of JSQR queue of context with empty queue (case JQM = 0) . . . 381
Figure 48. Example of JSQR queue of context with empty queue (case JQM = 1) . . . 382

Figure 49. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion . . . 382
Figure 50. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion and a new trigger occurs. . . . 383
Figure 51. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs outside an ongoing conversion . . . 383
Figure 52. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1) . . . 384
Figure 53. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0). . . . 384
Figure 54. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1). . . . 385
Figure 55. Single conversions of a sequence, software trigger . . . 387
Figure 56. Continuous conversion of a sequence, software trigger. . . . 387
Figure 57. Single conversions of a sequence, hardware trigger . . . 388
Figure 58. Continuous conversions of a sequence, hardware trigger . . . 388
Figure 59. Right alignment (offset disabled, unsigned value) . . . 390
Figure 60. Right alignment (offset enabled, signed value). . . . 391
Figure 61. Left alignment (offset disabled, unsigned value) . . . 391
Figure 62. Left alignment (offset enabled, signed value) . . . 392
Figure 63. Example of overrun (OVR) . . . 393
Figure 64. AUTODLY = 1, regular conversion in continuous mode, software trigger . . . 396
Figure 65. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . 396
Figure 66. AUTODLY = 1, regular HW conversions interrupted by injected conversions . . . (DISCEN = 1, JDISCEN = 1) . . . 397
Figure 67. AUTODLY = 1, regular continuous conversions interrupted by injected conversions . . . 398
Figure 68. AUTODLY = 1 in auto- injected mode (JAUTO = 1) . . . 398
Figure 69. Analog watchdog guarded area . . . 399
Figure 70. ADCy_AWDx_OUT signal generation (on all regular channels). . . . 401
Figure 71. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software) . . . 402
Figure 72. ADCy_AWDx_OUT signal generation (on a single regular channel) . . . 402
Figure 73. ADCy_AWDx_OUT signal generation (on all injected channels) . . . 402
Figure 74. 20-bit to 16-bit result truncation . . . 403
Figure 75. Numerical example with 5-bit shift and rounding . . . 403
Figure 76. Triggered regular oversampling mode (TROVS bit = 1) . . . 405
Figure 77. Regular oversampling modes (4x ratio) . . . 406
Figure 78. Regular and injected oversampling modes used simultaneously . . . 407
Figure 79. Triggered regular oversampling with injection . . . 407
Figure 80. Oversampling in auto-injected mode . . . 408
Figure 81. Temperature sensor channel block diagram . . . 409
Figure 82. VBAT channel block diagram . . . 410
Figure 83. VREFINT channel block diagram . . . 411
Figure 84. RNG block diagram . . . 446
Figure 85. Entropy source model. . . . 447
Figure 86. RNG initialization overview . . . 450
Figure 87. AES block diagram . . . 458
Figure 88. ECB encryption and decryption principle . . . 460
Figure 89. CBC encryption and decryption principle . . . 461
Figure 90. CTR encryption and decryption principle . . . 462
Figure 91. GCM encryption and authentication principle . . . 463
Figure 92. GMAC authentication principle . . . 463
Figure 93. CCM encryption and authentication principle . . . 464
Figure 94. Encryption key derivation for ECB/CBC decryption (Mode 2). . . . 467

Figure 95. Example of suspend mode management . . . 468
Figure 96. ECB encryption . . . 469
Figure 97. ECB decryption . . . 469
Figure 98. CBC encryption . . . 470
Figure 99. CBC decryption . . . 470
Figure 100. ECB/CBC encryption (Mode 1) . . . 471
Figure 101. ECB/CBC decryption (Mode 3) . . . 472
Figure 102. Message construction in CTR mode . . . 474
Figure 103. CTR encryption . . . 475
Figure 104. CTR decryption . . . 475
Figure 105. Message construction in GCM . . . 477
Figure 106. GCM authenticated encryption . . . 478
Figure 107. Message construction in GMAC mode . . . 482
Figure 108. GMAC authentication mode . . . 482
Figure 109. Message construction in CCM mode . . . 483
Figure 110. CCM mode authenticated encryption . . . 485
Figure 111. 128-bit block construction with respect to data swap . . . 489
Figure 112. DMA transfer of a 128-bit data block during input phase . . . 491
Figure 113. DMA transfer of a 128-bit data block during output phase . . . 492
Figure 114. PKA block diagram . . . 507
Figure 115. Advanced-control timer block diagram . . . 534
Figure 116. Counter timing diagram with prescaler division change from 1 to 2 . . . 536
Figure 117. Counter timing diagram with prescaler division change from 1 to 4 . . . 536
Figure 118. Counter timing diagram, internal clock divided by 1 . . . 538
Figure 119. Counter timing diagram, internal clock divided by 2 . . . 538
Figure 120. Counter timing diagram, internal clock divided by 4 . . . 539
Figure 121. Counter timing diagram, internal clock divided by N . . . 539
Figure 122. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . 540
Figure 123. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . 540
Figure 124. Counter timing diagram, internal clock divided by 1 . . . 542
Figure 125. Counter timing diagram, internal clock divided by 2 . . . 542
Figure 126. Counter timing diagram, internal clock divided by 4 . . . 543
Figure 127. Counter timing diagram, internal clock divided by N . . . 543
Figure 128. Counter timing diagram, update event when repetition counter is not used . . . 544
Figure 129. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 545
Figure 130. Counter timing diagram, internal clock divided by 2 . . . 546
Figure 131. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . 546
Figure 132. Counter timing diagram, internal clock divided by N . . . 547
Figure 133. Counter timing diagram, update event with ARPE=1 (counter underflow) . . . 547
Figure 134. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . 548
Figure 135. Update rate examples depending on mode and TIMx_RCR register settings . . . 549
Figure 136. External trigger input block . . . 550
Figure 137. TIM1 ETR input circuitry . . . 550
Figure 138. Control circuit in normal mode, internal clock divided by 1 . . . 551
Figure 139. TI2 external clock connection example . . . 552
Figure 140. Control circuit in external clock mode 1 . . . 553
Figure 141. External trigger input block . . . 553
Figure 142. Control circuit in external clock mode 2 . . . 554
Figure 143. Capture/compare channel (example: channel 1 input stage) . . . 555
Figure 144. Capture/compare channel 1 main circuit . . . 555
Figure 145. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3) . . . 556
Figure 146. Output stage of capture/compare channel (channel 4) . . . 556

Figure 147. Output stage of capture/compare channel (channel 5, idem ch. 6) . . . . . 557
Figure 148. PWM input mode timing . . . . . 559
Figure 149. Output compare mode, toggle on OC1 . . . . . 561
Figure 150. Edge-aligned PWM waveforms (ARR=8) . . . . . 562
Figure 151. Center-aligned PWM waveforms (ARR=8) . . . . . 563
Figure 152. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . . . 565
Figure 153. Combined PWM mode on channel 1 and 3 . . . . . 566
Figure 154. 3-phase combined PWM signals with multiple trigger pulses per period . . . . . 567
Figure 155. Complementary output with dead-time insertion . . . . . 568
Figure 156. Dead-time waveforms with delay greater than the negative pulse . . . . . 568
Figure 157. Dead-time waveforms with delay greater than the positive pulse. . . . . 569
Figure 158. Break and Break2 circuitry overview . . . . . 571
Figure 159. Various output behavior in response to a break event on BRK (OSSI = 1) . . . . . 573
Figure 160. PWM output state following BRK and BRK2 pins assertion (OSSI=1) . . . . . 574
Figure 161. PWM output state following BRK assertion (OSSI=0) . . . . . 575
Figure 162. Output redirection (BRK2 request not represented) . . . . . 576
Figure 163. Clearing TIMx OCxREF . . . . . 577
Figure 164. 6-step generation, COM example (OSSR=1) . . . . . 578
Figure 165. Example of one pulse mode. . . . . 579
Figure 166. Retriggerable one pulse mode . . . . . 581
Figure 167. Example of counter operation in encoder interface mode. . . . . 582
Figure 168. Example of encoder interface mode with TI1FP1 polarity inverted. . . . . 583
Figure 169. Measuring time interval between edges on 3 signals . . . . . 584
Figure 170. Example of Hall sensor interface . . . . . 586
Figure 171. Control circuit in reset mode . . . . . 587
Figure 172. Control circuit in Gated mode . . . . . 588
Figure 173. Control circuit in trigger mode . . . . . 589
Figure 174. Control circuit in external clock mode 2 + trigger mode . . . . . 590
Figure 175. General-purpose timer block diagram . . . . . 633
Figure 176. Counter timing diagram with prescaler division change from 1 to 2 . . . . . 635
Figure 177. Counter timing diagram with prescaler division change from 1 to 4 . . . . . 635
Figure 178. Counter timing diagram, internal clock divided by 1 . . . . . 636
Figure 179. Counter timing diagram, internal clock divided by 2 . . . . . 637
Figure 180. Counter timing diagram, internal clock divided by 4 . . . . . 637
Figure 181. Counter timing diagram, internal clock divided by N. . . . . 638
Figure 182. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded). . . . . 638
Figure 183. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded). . . . . 639
Figure 184. Counter timing diagram, internal clock divided by 1 . . . . . 640
Figure 185. Counter timing diagram, internal clock divided by 2 . . . . . 640
Figure 186. Counter timing diagram, internal clock divided by 4 . . . . . 641
Figure 187. Counter timing diagram, internal clock divided by N. . . . . 641
Figure 188. Counter timing diagram, Update event . . . . . 642
Figure 189. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 . . . . . 643
Figure 190. Counter timing diagram, internal clock divided by 2 . . . . . 644
Figure 191. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . . . 644
Figure 192. Counter timing diagram, internal clock divided by N. . . . . 645
Figure 193. Counter timing diagram, Update event with ARPE=1 (counter underflow). . . . . 645
Figure 194. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . . . 646
Figure 195. Control circuit in normal mode, internal clock divided by 1 . . . . . 647
Figure 196. TI2 external clock connection example. . . . . 647
Figure 197. Control circuit in external clock mode 1 . . . . . 648
Figure 198. External trigger input block . . . . . 649

Figure 199. Control circuit in external clock mode 2 . . . . . 650
Figure 200. Capture/Compare channel (example: channel 1 input stage) . . . . . 650
Figure 201. Capture/Compare channel 1 main circuit . . . . . 651
Figure 202. Output stage of Capture/Compare channel (channel 1) . . . . . 651
Figure 203. PWM input mode timing . . . . . 653
Figure 204. Output compare mode, toggle on OC1 . . . . . 655
Figure 205. Edge-aligned PWM waveforms (ARR=8) . . . . . 656
Figure 206. Center-aligned PWM waveforms (ARR=8) . . . . . 658
Figure 207. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . . . 659
Figure 208. Combined PWM mode on channels 1 and 3 . . . . . 660
Figure 209. Clearing TIMx OCxREF . . . . . 661
Figure 210. Example of one-pulse mode. . . . . 662
Figure 211. Retriggerable one-pulse mode. . . . . 664
Figure 212. Example of counter operation in encoder interface mode . . . . . 665
Figure 213. Example of encoder interface mode with TI1FP1 polarity inverted . . . . . 666
Figure 214. Control circuit in reset mode . . . . . 667
Figure 215. Control circuit in gated mode . . . . . 668
Figure 216. Control circuit in trigger mode . . . . . 669
Figure 217. Control circuit in external clock mode 2 + trigger mode . . . . . 670
Figure 218. Master/Slave timer example . . . . . 670
Figure 219. Master/slave connection example with 1 channel only timers . . . . . 671
Figure 220. Gating TIM2 with OC1REF of TIM1 . . . . . **672**
Figure 221. Gating TIM2 with Enable of TIM1 . . . . . 673
Figure 222. Triggering TIM2 with update of TIM1 . . . . . 673
Figure 223. Triggering TIM2 with Enable of TIM1 . . . . . 674
Figure 224. TIM16/TIM17 block diagram . . . . . 704
Figure 225. Counter timing diagram with prescaler division change from 1 to 2 . . . . . 706
Figure 226. Counter timing diagram with prescaler division change from 1 to 4 . . . . . 706
Figure 227. Counter timing diagram, internal clock divided by 1 . . . . . 708
Figure 228. Counter timing diagram, internal clock divided by 2 . . . . . 708
Figure 229. Counter timing diagram, internal clock divided by 4 . . . . . 709
Figure 230. Counter timing diagram, internal clock divided by N. . . . . 709
Figure 231. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded). . . . . 710
Figure 232. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded). . . . . 710
Figure 233. Update rate examples depending on mode and TIMx_RCR register settings . . . . . 712
Figure 234. Control circuit in normal mode, internal clock divided by 1 . . . . . 713
Figure 235. TI2 external clock connection example. . . . . 713
Figure 236. Control circuit in external clock mode 1 . . . . . 714
Figure 237. Capture/compare channel (example: channel 1 input stage) . . . . . 715
Figure 238. Capture/compare channel 1 main circuit . . . . . 715
Figure 239. Output stage of capture/compare channel (channel 1). . . . . 716
Figure 240. Output compare mode, toggle on OC1 . . . . . 719
Figure 241. Edge-aligned PWM waveforms (ARR=8) . . . . . 720
Figure 242. Complementary output with dead-time insertion. . . . . 721
Figure 243. Dead-time waveforms with delay greater than the negative pulse. . . . . 721
Figure 244. Dead-time waveforms with delay greater than the positive pulse. . . . . 722
Figure 245. Output behavior in response to a break . . . . . 724
Figure 246. Output redirection . . . . . 726
Figure 247. 6-step generation, COM example (OSSR=1) . . . . . 727
Figure 248. Example of one pulse mode . . . . . 728

Figure 249. Low-power timer block diagram . . . 755
Figure 250. Glitch filter timing diagram . . . 758
Figure 251. LPTIM output waveform, single counting mode configuration . . . 759
Figure 252. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 760
Figure 253. LPTIM output waveform, Continuous counting mode configuration . . . 760
Figure 254. Waveform generation . . . 762
Figure 255. Encoder mode counting sequence . . . 766
Figure 256. IRTIM internal hardware connections with TIM16 and TIM17 . . . 778
Figure 257. RTC block diagram . . . 781
Figure 258. Independent watchdog block diagram . . . 820
Figure 259. Watchdog block diagram . . . 830
Figure 260. Window watchdog timing diagram . . . 831
Figure 261. I2C block diagram . . . 837
Figure 262. I2C bus protocol . . . 839
Figure 263. Setup and hold timings . . . 841
Figure 264. I2C initialization flow . . . 843
Figure 265. Data reception . . . 844
Figure 266. Data transmission . . . 845
Figure 267. Slave initialization flow . . . 848
Figure 268. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0 . . . 850
Figure 269. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1 . . . 851
Figure 270. Transfer bus diagrams for I2C slave transmitter (mandatory events only) . . . 852
Figure 271. Transfer sequence flow for slave receiver with NOSTRETCH = 0 . . . 853
Figure 272. Transfer sequence flow for slave receiver with NOSTRETCH = 1 . . . 854
Figure 273. Transfer bus diagrams for I2C slave receiver (mandatory events only) . . . 854
Figure 274. Master clock generation . . . 856
Figure 275. Master initialization flow . . . 858
Figure 276. 10-bit address read access with HEAD10R = 0 . . . 858
Figure 277. 10-bit address read access with HEAD10R = 1 . . . 859
Figure 278. Transfer sequence flow for I2C master transmitter for N ≤ 255 bytes . . . 860
Figure 279. Transfer sequence flow for I2C master transmitter for N > 255 bytes . . . 861
Figure 280. Transfer bus diagrams for I2C master transmitter (mandatory events only) . . . 862
Figure 281. Transfer sequence flow for I2C master receiver for N ≤ 255 bytes . . . 864
Figure 282. Transfer sequence flow for I2C master receiver for N > 255 bytes . . . 865
Figure 283. Transfer bus diagrams for I2C master receiver (mandatory events only) . . . 866
Figure 284. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 870
Figure 285. Transfer sequence flow for SMBus slave transmitter N bytes + PEC . . . 873
Figure 286. Transfer bus diagrams for SMBus slave transmitter (SBC = 1) . . . 874
Figure 287. Transfer sequence flow for SMBus slave receiver N bytes + PEC . . . 875
Figure 288. Bus transfer diagrams for SMBus slave receiver (SBC = 1) . . . 876
Figure 289. Bus transfer diagrams for SMBus master transmitter . . . 877
Figure 290. Bus transfer diagrams for SMBus master receiver . . . 879
Figure 291. USART block diagram . . . 903
Figure 292. Word length programming . . . 906
Figure 293. Configurable stop bits . . . 908
Figure 294. TC/TXE behavior when transmitting . . . 911
Figure 295. Start bit detection when oversampling by 16 or 8 . . . 912
Figure 296. usart_ker_ck clock divider block diagram . . . 915

Figure 297. Data sampling when oversampling by 16 . . . 916
Figure 298. Data sampling when oversampling by 8 . . . 917
Figure 299. Mute mode using Idle line detection . . . 924
Figure 300. Mute mode using address mark detection . . . 925
Figure 301. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 928
Figure 302. Break detection in LIN mode vs. Framing error detection . . . 929
Figure 303. USART example of synchronous master transmission . . . 930
Figure 304. USART data clock timing diagram in synchronous master mode (M bits = 00) . . . 930
Figure 305. USART data clock timing diagram in synchronous master mode (M bits = 01) . . . 931
Figure 306. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . 932
Figure 307. ISO 7816-3 asynchronous protocol . . . 934
Figure 308. Parity error detection using the 1.5 stop bits . . . 936
Figure 309. IrDA SIR ENDEC block diagram . . . 940
Figure 310. IrDA data modulation (3/16) - normal mode . . . 940
Figure 311. Transmission using DMA . . . 942
Figure 312. Reception using DMA . . . 943
Figure 313. Hardware flow control between 2 USARTs . . . 943
Figure 314. RS232 RTS flow control . . . 944
Figure 315. RS232 CTS flow control . . . 945
Figure 316. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 948
Figure 317. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 948
Figure 318. SPI block diagram . . . 988
Figure 319. Full-duplex single master/ single slave application . . . 989
Figure 320. Half-duplex single master/ single slave application . . . 990
Figure 321. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) . . . 991
Figure 322. Master and three independent slaves . . . 992
Figure 323. Multimaster application . . . 993
Figure 324. Hardware/software slave select management . . . 994
Figure 325. Data clock timing diagram . . . 995
Figure 326. Data alignment when data length is not equal to 8-bit or 16-bit . . . 996
Figure 327. Packing data in FIFO for transmission and reception . . . 1000
Figure 328. Master full-duplex communication . . . 1003
Figure 329. Slave full-duplex communication . . . 1004
Figure 330. Master full-duplex communication with CRC . . . 1005
Figure 331. Master full-duplex communication in packed mode . . . 1006
Figure 332. NSSP pulse generation in Motorola SPI master mode . . . 1009
Figure 333. TI mode transfer . . . 1010
Figure 334. IPCC block diagram . . . 1023
Figure 335. IPCC Simplex channel mode transfer timing . . . 1024
Figure 336. IPCC Simplex - Send procedure state diagram . . . 1025
Figure 337. IPCC Simplex - Receive procedure state diagram . . . 1026
Figure 338. IPCC Half-duplex channel mode transfer timing . . . 1027
Figure 339. IPCC Half-duplex - Send procedure state diagram . . . 1027
Figure 340. IPCC Half-duplex - Receive procedure state diagram . . . 1028
Figure 341. HSEM block diagram . . . 1037
Figure 342. Procedure state diagram . . . 1038
Figure 343. Interrupt state diagram . . . 1041

Figure 344. Block diagram of debug support infrastructure . . . . . 1052
Figure 345. JTAG TAP state machine . . . . . 1055
Figure 346. Debug and access port connections. . . . . 1067
Figure 347. Embedded cross trigger . . . . . 1074
Figure 348. Mapping trigger inputs to outputs . . . . . 1076
Figure 349. Cross trigger configuration example . . . . . 1077
Figure 350. CPU2 CoreSight™ topology . . . . . 1103
Figure 351. CPU1 CoreSight™ topology . . . . . 1138
Figure 352. Trace port interface unit (TPIU) . . . . . 1173

# 1 Documentation conventions

## 1.1 General information

The STM32WB50CG/30CE devices embed an Arm®[a] CPU1 Cortex®-M4 core.

## 1.2 List of abbreviations for registers

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

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

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

## 1.3 Glossary

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

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **AHB**: advanced high-performance bus.

## 1.4 Availability of peripherals

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

# 2 System and memory overview

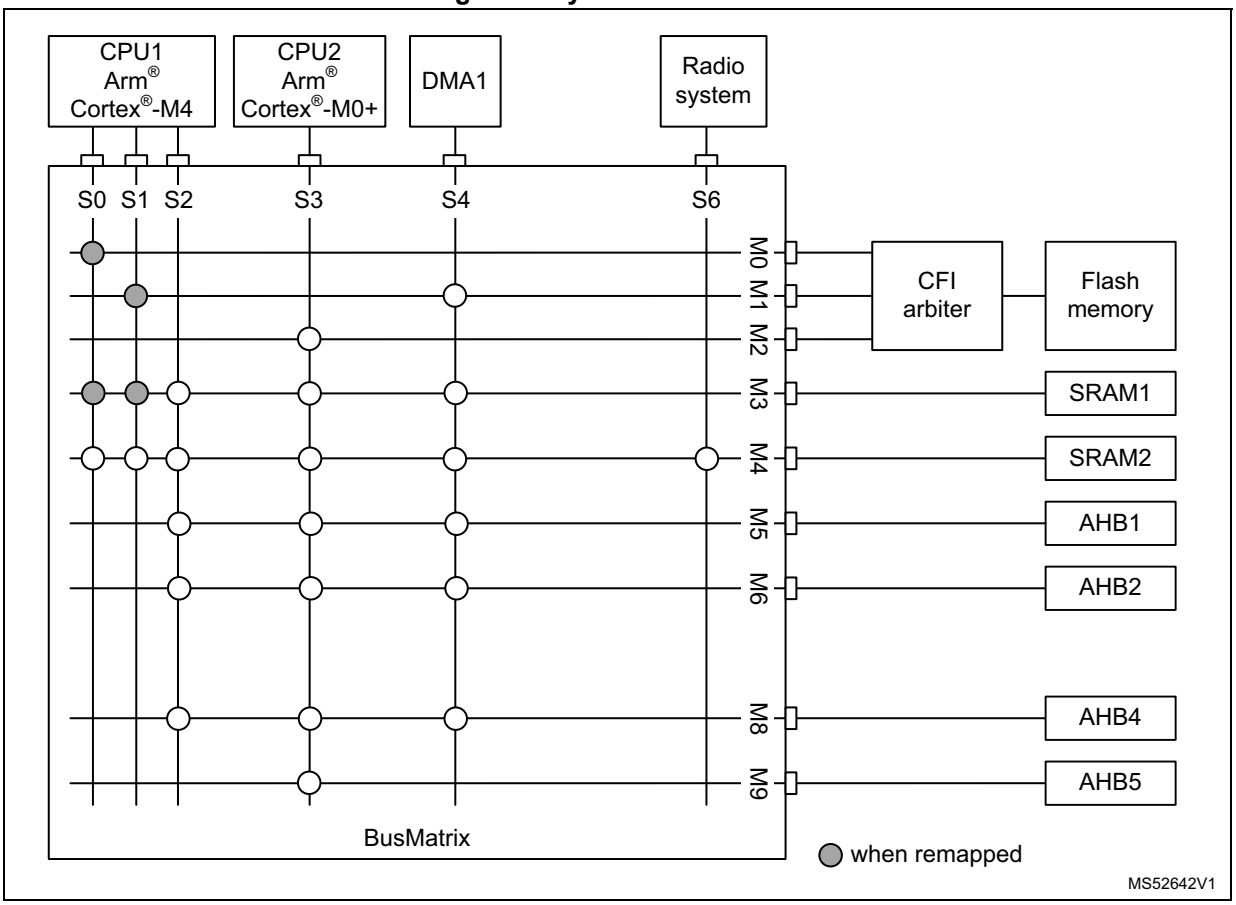
## 2.1 System architecture

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

- Seven masters:
  - CPU1 (CPU1 Cortex®-M4 with FPU) core I-bus
  - CPU1 (CPU1 Cortex®-M4 with FPU) core D-bus
  - CPU1 (CPU1 Cortex®-M4 with FPU) core S-bus
  - CPU2 (Cortex® -M0+) core S-bus
  - DMA1
  - Radio system
- Ten slaves:
  - Internal Flash memory on the CPU1 (CPU1 Cortex®-M4) ICode bus
  - Internal Flash memory on CPU1 (CPU1 Cortex®-M4) DCode bus
  - Internal Flash memory on CPU2 (Cortex® -M0+) S bus
  - Internal SRAM1 (64 KB for the STM32WB50CG, 32 KB for the STM32WB30CE)
  - Internal SRAM2a (32 KB) + SRAM2b (32 KB)
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - AHB4 shared peripheral
  - AHB5 including AHB to APB bridge and Radio peripherals (connected to APB3)

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*.

**Figure 1. System architecture**




### 2.1.1 S0: CPU1 (CPU1 Cortex®-M4) I-bus

This bus connects the instruction bus of the CPU1 core to the BusMatrix. This bus is used by the core to fetch instructions. The targets of this bus are the internal Flash memory, SRAM1, SRAM2a (backup) and SRAM2b (non backup).

### 2.1.2 S1: CPU1 (CPU1 Cortex®-M4) D-bus

This bus connects the data bus of the CPU1 core to the BusMatrix. This bus is used by the core for literal load and debug access. The targets of this bus are the internal Flash memory, SRAM1, SRAM2a (backup) and SRAM2b (non backup).

### 2.1.3 S2: CPU1 (CPU1 Cortex®-M4) S-bus

This bus connects the system bus of the CPU1 core to the BusMatrix. This bus is used by the core to access data located in a peripheral or SRAM area. The targets of this bus are SRAM1, SRAM2a (backup), SRAM2b (non backup), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals and the AHB4 peripherals.

### 2.1.4 S3: CPU2 (Cortex®-M0+) S-bus

This bus connects the system bus of the CPU2 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 the internal Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, and the AHB4 peripherals and the AHB5 peripherals including the APB3 peripherals.

### 2.1.5 S4: DMA-bus

This bus connects the AHB master interface of the DMA to the BusMatrix.The targets of this bus are the SRAM1, SRAM2a (backup), SRAM2b (non backup), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals and the AHB4 peripherals.

### 2.1.6 S6: Radio system-bus

This bus connects the AHB master interface of the Radio system to the BusMatrix.The targets of this bus are the SRAM2a (backup) and SRAM2b (non backup).

### 2.1.7 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a Round Robin algorithm. The BusMatrix is composed by six masters (CPU1: system bus, DCode bus, ICode bus, CPU2: system bus, DMA1-bus and Radio system-bus) and nine slaves (3 x Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup), AHB1 (including APB1 and APB2), AHB2, AHB4, and AHB5).

#### AHB/APB bridges

The two bridges AHB to APB1 and AHB to ABP2 provide full synchronous connections between the AHB and the two APB buses, allowing flexible selection of the peripheral frequency.

The bridges AHB to APB3 provide an a-synchronous connections between the AHB and the APB bus, allowing flexible selection of the frequency between the AHB and peripheral.

Refer to *Section 2.2: Memory organization* for the address mapping of the peripherals connected to this bridge.

After each device reset, all peripheral clocks are disabled (except for the SRAM1/2 and Flash memory interface). Before using a peripheral you have 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

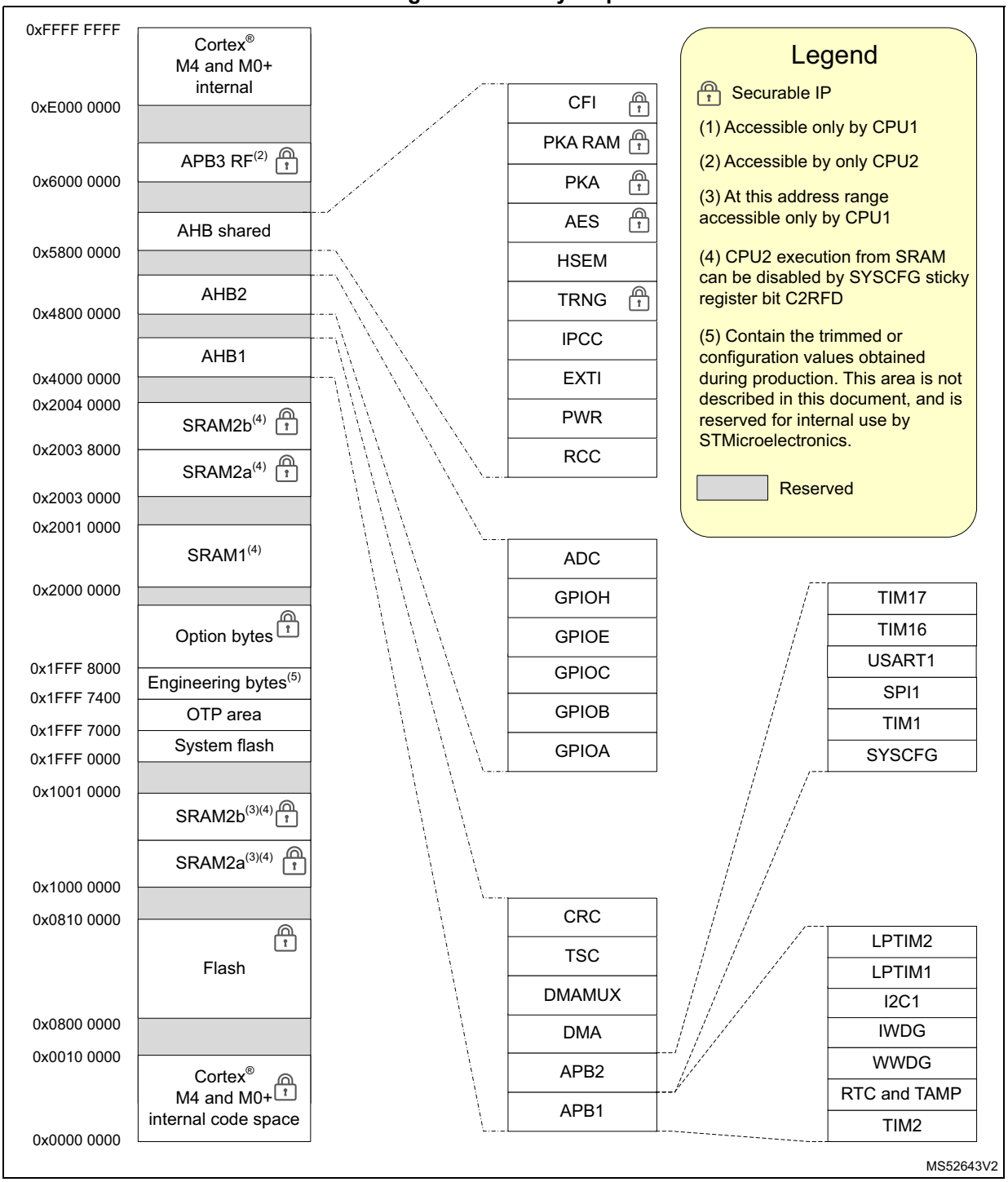
Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

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

The addressable memory space is divided into eight main blocks, of 512 Mbytes each.

### 2.2.2 Memory map and register boundary addresses

**Figure 2. Memory map**




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

**Table 1. STM32WB50CG/30CE memory map and peripheral register boundary addresses**

| Bus | Boundary address | Size (bytes) | Peripheral | Peripheral register map |
|---|---|---|---|---|
| - | 0x6000 2000 - 0xDFFF FFFF | - | Reserved | - |
| APB3 | 0x6000 1000 - 0x6000 1FFF | 4 K | 802.15.4 CTRL | - |
| | 0x6000 0800 - 0x6000 0FFF | 2 K | Reserved | - |
| | 0x6000 0400 - 0x6000 07FF | 1 K | Radio CTRL | - |
| | 0x6000 0000 - 0x6000 03FF | 1 K | BLE CTRL | - |
| AHB4 | 0x5800 4400 - 0x5FFF FFFF | - | Reserved | - |
| | 0x5800 4000 - 0x5800 43FF | 1 K | FLASH | *Section 3.10.20: FLASH register map on page 116* |
| | 0x5800 3400 - 0x5800 3FFF | - | Reserved | - |
| | 0x5800 2400 - 0x5800 33FF | 5 K | PKA RAM | *Section 18.7.5: PKA register map on page 531* |
| | 0x5800 2000 - 0x5800 23FF | | PKA | |
| | 0x5800 1C00 - 0x5800 1FFF | 1 K | Reserved | - |
| | 0x5800 1800 - 0x5800 1BFF | 1 K | AES2 | *Section 17.7.18: AES register map on page 504* |
| | 0x5800 1400 - 0x5800 17FF | 1 K | HSEM | *Section 31.4.9: HSEM register map on page 1048* |
| | 0x5800 1000 - 0x5800 13FF | 1 K | True RNG | *Section 16.7.4: RNG register map on page 456* |
| | 0x5800 0C00 - 0x5800 0FFF | 1 K | IPCC | *Section 30.4.9: IPCC register map on page 1035* |
| | 0x5800 0800 - 0x5800 0BFF | 1 K | EXTI | *Section 14.5.17: EXTI register map on page 352* |
| | 0x5800 0400 - 0x5800 07FF | 1 K | PWR | *Section 6.6.21: PWR register map and reset value table on page 169* |
| | 0x5800 0000 - 0x5800 03FF | 1 K | RCC | *Section 7.4.47: RCC register map on page 242* |
| AHB2 | 0x5004 0400 - 0x57FF FFFF | - | Reserved | - |
| | 0x5004 0000 - 0x5004 03FF | 1 K | ADC | *Section 15.8: ADC register map on page 442* |
| | 0x4800 1C00 - 0x4800 1FFF | 1 KB | GPIOH | *Section 8.4.12: GPIO register map on page 266* |
| | 0x4800 1400 - 0x4800 1BFF | 3 K | Reserved | - |
| | 0x4800 1000 - 0x4800 13FF | 1 K | GPIOE | *Section 8.4.12: GPIO register map on page 266* |
| | 0x4800 0C00 - 0x4800 0FFF | 1 K | Reserved | |
| | 0x4800 0800 - 0x4800 0BFF | 1 K | GPIOC | |
| | 0x4800 0400 - 0x4800 07FF | 1 K | GPIOB | |
| | 0x4800 0000 - 0x4800 03FF | 1 K | GPIOA | |

**Table 1. STM32WB50CG/30CE memory map and peripheral register boundary addresses**

| Bus | Boundary address | Size (bytes) | Peripheral | Peripheral register map |
|---|---|---|---|---|
| AHB1 | 0x4002 3400 - 0x47FF FFFF | - | Reserved | - |
| | 0x4002 3000 - 0x4002 33FF | 1 K | CRC | *Section 5.4.6: CRC register map on page 126* |
| | 0x4002 0C00 - 0x4002 2FFF | 9 K | Reserved | - |
| | 0x4002 0800 - 0x4002 0BFF | 1 K | DMAMUX | *Section 12.6.7: DMAMUX register map on page 325* |
| | 0x4002 0400 - 0x4002 07FF | 1 K | Reserved | - |
| | 0x4002 0000 - 0x4002 03FF | 1 K | DMA1 | *Section 11.6.7: DMA register map on page 308* |
| - | 0x4001 4C00 - 0x4001 FFFF | 45 K | Reserved | - |
| APB2 | 0x4001 4800 - 4001 4BFF | 1 K | TIM17 | *Section 21.4.23: TIM16/TIM17 register map on page 752* |
| | 0x4001 4400 - 4001 47FF | 1 K | TIM16 | *Section 21.4.23: TIM16/TIM17 register map on page 752* |
| | 0x4001 3C00 - 4001 43FF | 2 K | Reserved | - |
| | 0x4001 3800 - 4001 3BFF | 1 K | USART1 | *Section 28.8.15: USART register map on page 985* |
| | 0x4001 3400 - 4001 37FF | 1 K | Reserved | - |
| | 0x4001 3000 - 4001 33FF | 1 K | SPI1 | *Section 29.6.8: SPI register map on page 1021* |
| | 0x4001 2C00 - 4001 2FFF | 1 K | TIM1 | *Section 19.4.30: TIM1 register map on page 629* |
| | 0x4001 0400 - 4001 2BFF | 10 K | Reserved | - |
| | 0x4001 0200 - 4001 03FF | 1 K | Reserved | - |
| | 0x4001 0100 - 4001 01FF | 1 K | SYSCFG | *Section 9.2.17: SYSCFG register map on page 284* |
| | 0x4001 0030 - 4001 00FF | 1 K | Reserved | - |
| | 0x4001 0000 - 4001 002F | 1 K | SYSCFG | *Section 9.2.17: SYSCFG register map on page 284* |

**Table 1. STM32WB50CG/30CE memory map and peripheral register boundary addresses**

| Bus | Boundary address | Size (bytes) | Peripheral | Peripheral register map |
|---|---|---|---|---|
| APB1 | 0x4000 9800 - 4000 FFFF | 26 K | Reserved | - |
| | 0x4000 9400 - 04000 97FF | 1 K | LPTIM2 | *Section 22.7.9: LPTIM register map on page 777* |
| | 0x4000 8000 - 04000 93FF | 5 K | Reserved | - |
| | 0x4000 7C00 - 04000 7FFF | 1 K | LPTIM1 | *Section 22.7.9: LPTIM register map on page 777* |
| | 0x4000 5800 - 04000 7BFF | 9 K | Reserved | - |
| | 0x4000 5400 - 04000 57FF | 1 K | I2C1 | *Section 27.7.12: I2C register map on page 898* |
| | 0x4000 3400 - 04000 53FF | 8 K | Reserved | - |
| | 0x4000 3000 - 0x4000 33FF | 1 K | IWDG | *Section 25.4.6: IWDG register map on page 828* |
| | 0x4000 2C00 - 0x4000 2FFF | 1 K | WWDG | *Section 26.5.4: WWDG register map on page 834* |
| | 0x4000 2800 - 0x4000 2BFF | 1 K | RTC and TAMP | *Section 24.6.21: RTC register map on page 818* |
| | 0x4000 0400 - 0x4000 27FF | 9 K | Reserved | - |
| | 0x4000 0000 - 0x4000 03FF | 1 K | TIM2 | *Section 20.4.25: TIMx register map on page 700* |
| AHB4 | 0x2003 8000 - 0x2003 FFFF | 32 K | SRAM2b | - |
| | 0x2003 0000 - 0x2003 7FFF | 32 K | SRAM2a | - |
| AHB1 | 0x2001 0000 - 0x2002 FFFF | - | Reserved | - |
| | 0x2000 0000 - 0x2000 FFFF | 64 K | SRAM1(1) | - |
| AHB4 | 0x1FFF 8000 - 0x1FFF 807F | 128 | Flash memory options | *Section 3.10.20: FLASH register map on page 116* |
| | 0x1FFF 7000 - 0x1FFF 73FF | 1 K | Flash memory OTP | - |
| | 0x1FFF 0000 - 0x1FFF 6FFF | 28 K | Flash memory Boot loader | - |
| | 0x1000 0000 - 0x1000 FFFF | 64 K | SRAM2a/b CPU1 mirror | - |
| | 0x0800 0000 - 0x080 FFFF | 1 M(2) | User Flash memory | - |
| (3) | 0x0000 0000 - 0x000 FFFF | 1 M(2) | CPUn Boot area | - |

1. 64 KB for STM32WB50CG, 32 KB for STM32WB30CE.
2. 1 MB for STM32WB50CG, 512 KB for STM32WB30CE.
3. Bus depends upon selected CPUn Boot area.

### 2.2.3 Bit banding

The CPU1 map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

The AHB1, APB1, APB2 peripheral registers and the SRAM1, SRAM2a and SRAM2b are mapped to a bit-band region, hence single bit-band write and read operations are allowed. The operations are only available for CPU1 accesses, and not form other bus masters (e.g. DMA)

The peripheral bit-band alias is located from address 0x4200 0000 to 0x42FF FFFF

The SRAM bit-band alias is located from address 0x2200 0000 to 0x227F FFFF

A mapping formula shows how to reference each word in the alias region to a corresponding bit in the bit-band region. The mapping formula is:

bit_word_addr = bit_band_base + (byte_offset * 32) + (bit_number * 4), where:

- bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- bit_band_base is the starting address of the alias region
- byte_offset is the number of the byte in the bit_band region that contains the targeted bit
- bit_number is the bit position (0-7) of the targeted bit

#### Example

The following example shows how to map bit [2] of the byte located at SRAM1 address 0x2000 0300 to the alias region.

0x2200 6008 = 0x2200 0000 + 0x0300 * 32 + 2 * 4

Writing to address 0x2200 6008 has the same effect as a read-modify-write operation on bit [2] of the byte at SRAM1 address 0x2000 0300.

Reading address 0x2200 6008 returns the value 0x01 or 0x00 of bit [2] of the byte at SRAM1 address 0x2000 0300.

For more information on bit-band, refer to the Cortex®-M4 programming manual.

## 2.3 Boot configuration

Three different CPU1 boot modes can be selected through the BOOT0 pin and nBOOT1 bit in the User options, as shown in *Table 2*.

**Table 2. Boot modes**

| nBOOT1 FLASH_OPTR[23] | nBOOT0 FLASH_OPTR[27] | BOOT0 pin PH3 | nSWBOOT0 FLASH_OPTR[26] | Main flash empty(1) | Boot memory space alias |
|---|---|---|---|---|---|
| x | x | 0 | 1 | 0 | Main flash memory is selected as boot area |
| x | x | 0 | 1 | 1 | System memory is selected as boot area |

**Table 2. Boot modes (continued)**

| nBOOT1 FLASH_OPTR[23] | nBOOT0 FLASH_OPTR[27] | BOOT0 pin PH3 | nSWBOOT0 FLASH_OPTR[26] | Main flash empty(1) | Boot memory space alias |
|---|---|---|---|---|---|
| x | 1 | x | 0 | x | Main flash memory is selected as boot area |
| 0 | x | 1 | 1 | x | Embedded SRAM1 is selected as boot area |
| 0 | 0 | x | 0 | x | |
| 1 | x | 1 | 1 | x | System memory is selected as boot area |
| 1 | 0 | x | 0 | x | |

1. A flash empty check mechanism is implemented to force the boot from system flash if the first flash memory location is not programmed (0xFFFF FFFF) and if the boot selection was configured to boot from the main flash.

The values on both BOOT0 and BOOT1 are latched after a reset. It is up to the user to provide the correct value for the required boot mode.

The BOOT0 and BOOT1 are also re-sampled when exiting Standby mode. Consequently they must be kept in the required boot mode. After the startup delay, the CPU1 fetches the top-of-stack from address 0x0000 0000, then starts code execution from the boot memory at 0x0000 0004.

Depending on the selected boot mode, main flash, system flash, or SRAM1 memories are accessible as follows:

- Boot from main flash memory: the main flash memory is aliased in the CPU1 boot memory space at address 0x0000 0000, and is accessible even from its physical address 0x0800 0000. In other words, the flash memory content can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from system flash memory: the system flash memory is aliased in the CPU1 boot memory space at address 0x0000 0000, and is also still accessible from its physical address 0x1FFF 0000.
- Boot from SRAM: the memory is aliased in the CPU1 boot memory space at address 0x0000 0000, and is accessible even from its physical address 0x2000 0000.

### Empty check

An internal empty check flag (the EMPTY bit of the FLASH access control register (FLASH_ACR)) is available for easy programming of virgin devices by the boot loader. This flag is used when BOOT0 pin is defining main flash as the target boot area. When the flag is set, the device is considered as empty, and the system memory (boot loader) is selected instead of the main flash as a boot area, to allow user to program the memory. Therefore, some of the GPIOs are reconfigured from the high-Z state. Refer to AN2606 for more details concerning the bootloader and GPIO configuration in system memory boot mode. It is possible to disable this feature by configuring the option bytes to force boot from the main flash memory (nSWBOOT0 = 0, nBOOT0 = 1).

This empty check flag is updated only during the loading of option bytes: it is set when the content of the address 0x08000 0000 is read as 0xFFFF FFFF, otherwise it is cleared. A power reset or setting the OBL_LAUNCH bit in FLASH_CR register is needed to clear this flag after programming of a virgin device, to execute user code after System reset. The EMPTY bit can be written directly by software.

### CPU1 physical remap

Following CPU1 boot the application software can modify the memory map at address 0x0000 0000. This modification is performed by programming the *SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller.

The following memories can be remapped:

- Main flash memory
- System flash memory
- SRAM

### Embedded boot loader

The embedded boot loader is located in the system flash memory, programmed by ST during production. It is used to program the flash memory using one of the following device interfaces:

- USART1 on pins PA9 and PA10
- I2C1 on pins PB6 and PB7.
- SPI1 on pins PA4, PA5, PA6 and PA7

## 2.4 CPU2 boot

Following a device reset the CPU2 will only boot after CPU1 has set the C2BOOT bit in the *PWR control register 4 (PWR_CR4)*. The C2BOOT value is retained in Standby mode and the CPU2 will boot accordingly when exit from Standby.

The CPU2 will boot from it boot reset vector as defined by the flash user option C2OPT and SBRV.

The CPU2 may boot from anywhere in user flash or SRAM1/SRAM2a/SRAM2b.

### CPU2 safe boot

When, after a reset, the User options are not valid and the BOOT0 and BOOT1 select CPU1 to boot from main flash memory, the CPU2 boots from a safe boot vector in main flash memory at address 0x080F F000.

The safe boot can be used to restore the last known user options from an copied image.

## 2.5 CPU2 SRAM fetch disable

CPU2 execution from SRAM can be disabled by the C2RFD bit in SYSCFG register. Disabling CPU2 execution from SRAM improves robustness of the CPU2 software.

# 3 Embedded flash memory (FLASH)

## 3.1 Introduction

The flash memory interface manages CPU1 (CPU1 Cortex®-M4) AHB ICode and DCode accesses and the CPU2 (Cortex® -M0+) AHB to the flash memory. It implements the access arbitration between the two CPUs, the erase and program flash memory operations, the read and write protection, and the security mechanisms.

The flash memory interface accelerates code execution with a system of instruction prefetch and cache lines.

## 3.2 FLASH main features

- Up to 1 Mbyte of flash memory single bank architecture
- Memory organization: 1 bank
  - main memory: up to 1 Mbyte
  - page size: 4 Kbytes
- 72-bit wide data read (64 bits plus 8 ECC bits)
- 72-bit wide data write (64 bits plus 8 ECC bits)
- Page erase (4 Kbytes), and mass erase

Flash memory interface features:

- Flash memory read operations
- Flash memory program/erase operations
- Read protection activated by option (RDP)
- Two write protection areas selected by option (WRP)
- Two proprietary code read protection area selected by option (PCROP)
- CPU2 Security area
- Flash empty check
- Program and Erase suspension feature
- Prefetch on CPU1 ICODE and CPU2 S-bus
- CPU1 instruction cache: 32 cache lines of 4 x 64 bits on ICode (1 Kbyte RAM)
- CPU1 data cache: eight cache lines of 4 x 64 bits on DCode (256 bytes RAM)
- CPU2 instruction cache: four cache lines of 1 x 64 bits on S-bus (32 bytes RAM)
- Error code correction (ECC): eight bits for 64-bit
- Option byte loader

## 3.3 FLASH functional description

### 3.3.1 Flash memory organization

The memory is organized as 72-bit wide memory cells (64 bits, plus 8 ECC bits) that can be used for storing both code and data constants. It includes

- A main memory block containing up to 256 pages for STM32WB50CG, up to 128 pages for STM32WB30CE of 4 Kbytes, each page with eight rows of 512 bytes.
- An information block containing:
  - System memory from which the CPU1 boots in System memory boot mode. The area is reserved and contains the boot loader used to reprogram the flash memory through one of the following interfaces: USART1, I2C1 SPI1. It is programmed by STMicroelectronics when the device is manufactured, and protected against spurious write/erase operations. For further details, refer to the AN2606 available from *www.st.com*.
  - 1 Kbyte (128 double words) OTP (one-time programmable) for user data. The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double word (64 bits) cannot be written anymore, even with the value 0x0000 0000 0000 0000.
  - Option bytes for user configuration.

The memory is based on a main area and an information block as shown in *Table 3*.

**Table 3. Flash memory - Single bank organization**

| Area | Addresses | Size (bytes) | Name |
|---|---|---|---|
| Main memory(1) | 0x0800 0000 - 0x0800 0FFF | 4 K | Page 0 |
| | 0x0800 1000 - 0x0800 1FFF | 4 K | Page 1 |
| | 0x0800 2000 - 0x0800 2FFF | 4 K | Page 2 |
| | . . . | . . . | . . . |
| | 0x0807 E000 - 0x0807 EFFF | 4 K | Page 126 |
| | 0x0807 F000 - 0x0807 FFFF | 4 K | Page 127 |
| | 0x0808 0000 - 0x0808 0FFF | 4 K | Page 128 |
| | . . . | . . . | . . . |
| | 0x080F E000 - 0x080F EFFF | 4 K | Page 254 |
| | 0x080F F000 - 0x080F FFFF | 4 K | Page 255 |
| Information block | 0x1FFF 0000 - 0x1FFF 6FFF | 28 K | System memory |
| | 0x1FFF 7000 - 0x1FFF 73FF | 1 K | OTP area |
| | 0x1FFF 8000 - 0x1FFF 807F | 128 | Option bytes |

1. Pages 128 to 255 available on STM32WB50CG only.

### 3.3.2 Empty check

During the OBL phase, after loading all options, the flash memory interface checks whether the first location of the main memory is programmed. The result of this check in conjunction with the boot0 and boot1 information is used to determine where the system has to boot from. It prevents the system to boot from flash main memory area when, for instance, no user code has been programmed.

The flash main memory empty check status can be read from the EMPTY bit in *Flash memory access control register (FLASH_ACR)*. Software can modify the flash main memory empty status by writing to the EMPTY bit.

### 3.3.3 Error code correction (ECC)

Data in flash memory words are 72-bit wide: eight bits are added per each double word (64 bits). The ECC mechanism supports:

- One error detection and correction
- Two errors detection

When one error is detected and corrected, the flag ECCC (ECC correction) is set in *Flash memory ECC register (FLASH_ECCR)*. If ECCCIE is set, an interrupt is generated.

When two errors are detected, a flag ECCD (ECC detection) is set in *Flash memory ECC register (FLASH_ECCR)*. In this case, a NMI is generated.

When an ECC error is detected, the address of the failing double word is saved in ADDR_ECC[16:0] in the FLASH_ECCR register. ADDR_ECC[2:0] are always cleared. The bus-ID of the CPU accessing the address is saved in CPUID[2:0].

While ECCC or ECCD is set, FLASH_ECCR is not updated if a new ECC error occurs. FLASH_ECCR is updated only when ECC flags are cleared.

*Note: For a virgin data (0xFF FFFF FFFF FFFF FFFF), one error is detected and corrected, but two errors detection is not supported.*

*When an ECC error is reported, a new read at the failing address may not generate an ECC error if the data is still present in the current buffer, even if ECCC and ECCD are cleared. If this is not the desired behavior, the user must reset the cache.*

### 3.3.4 Read access latency

To correctly read data from flash memory, the number of wait states (LATENCY) must be correctly programmed in the *Flash memory access control register (FLASH_ACR)* according to the frequency of the flash memory (HCLK4) clock.

*Table 4* shows the correspondence between wait states and flash clock frequency.

**Table 4. Number of wait states vs, flash memory clock (HCLK4) frequency**

| Wait states (WS) (LATENCY) | HCLK4 (MHz) |
|---|---|
| 0 WS (1 HCLK cycles) | ≤ 18 |
| 1 WS (2 HCLK cycles) | ≤ 36 |
| 2 WS (3 HCLK cycles) | ≤ 54 |
| 3 WS (4 HCLK cycles) | ≤ 64 |

After POR on reset, the HCLK4 clock frequency is 4 MHz and 0 wait state (WS) is configured in the FLASH_ACR register.

When woken-up from Standby, the HCLK4 clock frequency is 16 MHz and 0 wait state (WS) is configured in the FLASH_ACR register.

When changing the flash memory clock frequency, the software sequences described below must be applied in order to tune the number of wait states needed to access the flash memory.

### Increasing the CPU frequency

1. Program the new number of wait states to the LATENCY bits in the *Flash memory access control register (FLASH_ACR)*.
2. Check that the new number of wait states is taken into account to access the flash memory by reading back the LATENCY from the *Flash memory access control register (FLASH_ACR)*, and wait until the programmed new number is read.
3. Modify the System cock source by writing the SW bits in the RCC_CFGR register.
4. If needed, modify the CPU clock prescaler by writing the SHDHPRE bits in RCC_EXTCFGR.
5. Optionally, check that the new System clock source or/and the new flash memory clock prescaler value is/are taken into account by reading the clock source status (SWS bits) in the RCC_CFGR register, or/and the AHB prescaler value (SHDHPREF bit), in the RCC_EXTCFGR register.

### Decreasing the CPU frequency

1. Modify the System clock source by writing the SW bits in the RCC_CFGR register.
2. If needed, modify the flash memory clock prescaler by writing the SHDHPRE bits in RCC_EXTCFGR.
3. Check that the new System clock source or/and the new flash memory clock prescaler value is/are taken into account by reading the clock source status (SWS bits) in the RCC_CFGR register, or/and the AHB prescaler value (SHDHPREF bit), in the RCC_EXTCFGR register, and wait until the programmed new system clock source or/and new flash memory clock prescaler value is/are read.
4. Program the new number of wait states to the LATENCY bits in *Flash memory access control register (FLASH_ACR)*.
5. Optionally, check that the new number of wait states is used to access the flash memory by reading back the LATENCY from the *Flash memory access control register (FLASH_ACR)*.

### 3.3.5 Adaptive real-time memory accelerator (ART Accelerator)

The proprietary Adaptive real-time (ART) memory accelerator is optimized for STM32 industry-standard Arm® Cortex®-M4 with FPU processors. It balances the inherent performance advantage of the Arm® Cortex®-M4 with FPU over flash memory technologies, which normally require the processor to wait for the flash memory at higher operating frequencies.

To release the processor full performance, the accelerator implements an instruction prefetch queue and branch cache that increases program execution speed from the 64-bit flash memory. Based on CoreMark® benchmark, the performance achieved thanks to the ART Accelerator™ is equivalent to 0 wait state program execution from flash memory at a CPU frequency up to 64 MHz.

#### Instruction prefetch

The CPU1 fetches the instruction over the ICode bus and the literal pool (constant/data) over the DCode bus. The prefetch block aims at increasing the efficiency of ICode bus accesses.

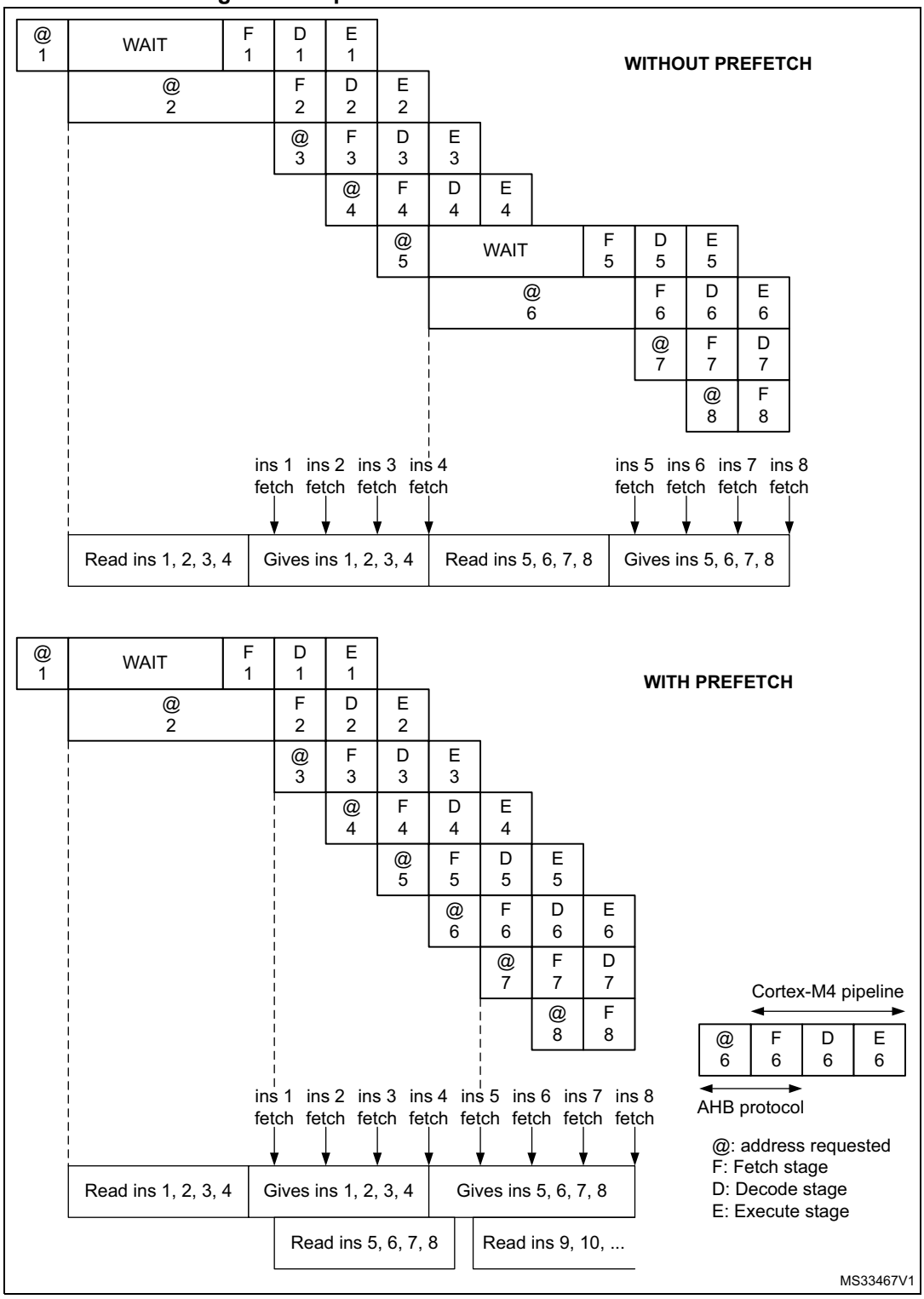
The CPU2 fetches the instruction and the literal pool (constant/data) over the S-bus. The prefetch block aims at increasing the efficiency of S-bus accesses.

Each flash memory read operation provides 64 bits from either two instructions of 32 bits or four instructions of 16 bits according to the program launched. This 64-bits current instruction line is saved in a current buffer. So, in case of sequential code, at least two CPU cycles are needed to execute the previous read instruction line. Prefetch on the CPU1 ICode bus or CPU2 S-bus can be used to read the next sequential instruction line from the flash memory while the current instruction line is being requested by the CPU.

Prefetch is enabled by setting the PRFTEN bit in the *Flash memory access control register (FLASH_ACR)* for the CPU1 or *Flash memory CPU2 access control register (FLASH_C2ACR)* for the CPU2. This feature is useful if at least one wait state is needed to access the flash memory.

*Figure 3* shows the execution of sequential 16-bit instructions with and without prefetch when three WS are needed to access the flash memory.

**Figure 3. Sequential 16-bit instructions execution**




When the code is not sequential (branch), the instruction may not be present in the currently used instruction line or in the prefetched instruction line. In this case (miss), the penalty in terms of number of cycles is at least equal to the number of wait states.

If a loop is present in the current buffer, no new access is performed.

### CPU1 Instruction cache memory (I-Cache)

To limit the CPU1 time lost due to jumps, it is possible to retain 32 lines of 4*64 bits (1 Kbyte) in an instruction cache memory.This feature can be enabled for the CPU1 by setting the instruction cache enable (ICEN) bit in the *Flash memory access control register (FLASH_ACR)*. Each time a miss occurs (requested data not present in the currently used instruction line, in the prefetched instruction line or in the instruction cache memory), the line read is copied into the instruction cache memory. If some data contained in the instruction cache memory are requested by the CPU, they are provided without inserting any delay. Once all the instruction cache memory lines have been filled, the LRU (least recently used) policy is used to determine the line to replace in the instruction memory cache. This feature is particularly useful in case of code containing loops.

The Instruction cache memory is enable after system reset.

### CPU1 Data cache memory (D-Cache)

CPU1 literal pools are fetched from flash memory through the DCode bus during the execution stage of the CPU pipeline. Each CPU1 DCode bus read access fetches 64 bits that are saved in a current buffer. The CPU pipeline is consequently stalled until the requested literal pool is provided. To limit the time lost due to literal pools, accesses through the AHB databus DCode have priority over accesses through the AHB instruction bus ICode.

If some literal pools are frequently used, the CPU1 data cache memory can be enabled by setting the data cache enable (DCEN) bit in the *Flash memory access control register (FLASH_ACR)*. This feature works like the instruction cache memory, but the retained data size is limited to 8 lines of 4*64 bits (256 bytes).

The Data cache memory is enabled after system reset.

*Note: The D-Cache is active only when data is requested by the CPU (not by DMAs). Data in option bytes block are not cacheable.*

### CPU2 cache memory (S-bus)

To limit the CPU2 time lost due to jumps, it is possible to retain four lines of 1*64 bits (32 bytes) in an instruction cache memory.This feature can be enabled for the CPU2 by setting the instruction cache enable (ICEN) bit in the *Flash memory CPU2 access control register (FLASH_C2ACR)*. Each time a miss occurs (requested data not present in the currently used instruction line, in the prefetched instruction line or in the instruction cache memory), the line read is copied into the instruction cache memory. If some data contained in the instruction cache memory are requested by the CPU, they are provided without inserting any delay. Once all the instruction cache memory lines have been filled, the LRU (least recently used) policy is used to determine the line to replace in the instruction memory cache. This feature is particularly useful in case of code containing loops.

The Instruction cache memory is enable after system reset.

CPU2 literal pools are fetched from flash memory through the S-bus during the execution stage of the CPU pipeline. Each CPU2 S-bus read access fetches 64 bits that are saved in a current buffer. The CPU pipeline is consequently stalled until the requested literal pool is provided.

No Data cache is available on Cortex®-M0+.

### 3.3.6 Flash memory program and erase operations

The embedded flash memory can be programmed using in-circuit or in-application programming.

The **in-circuit programming (ICP)** method is used to update the entire contents of the flash memory, using the JTAG, SWD protocol or the supported interfaces by the System boot loader, to load the user application for both the CPU1 and CPU2, into the microcontroller. ICP offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices.

In contrast to the ICP method, **in-application programming (IAP)** can use any communication interface supported by the microcontroller (e.g. I/Os, UART, $I^2C$, SPI) to download programming data into memory. IAP allows the user to re-program the flash memory while the application is running. Nevertheless, part of the application has to have been previously programmed in the flash memory using ICP.

The contents of the flash memory are not guaranteed if a device reset occurs during a flash memory operation.

During a program/erase operation to the flash memory, any attempt to read the flash memory stalls the bus. The read operation proceeds correctly once the program/erase operation is completed.

*Note: In a multi-CPU system it is good practice to use semaphores to manage flash memory program and erase operations, and prevent simultaneous operations by the CPUs.*

#### Secure system flash memory programming

The secure CPU2 application can only be programmed by **in-application programming (IAP)** running on the secure CPU2 . Only the secure CPU2 is able to download programming data into secure part of the memory. Secure IAP allows the user to re-program the flash memory while the application is running. Nevertheless, part of the application has to have been previously programmed in the flash memory using ICP.

The **in-circuit programming (ICP)** System boot loader is able to communicate with the secure CPU2 IAP, to download programming data into secure memory.

#### Unlocking the flash memory

After reset, write is not allowed in the *Flash memory control register (FLASH_CR)* or *Flash memory CPU2 control register (FLASH_C2CR)* to protect the flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence is used to unlock these registers:

1. Write KEY1 = 0x4567 0123 in the *Flash memory key register (FLASH_KEYR)*
2. Write KEY2 = 0xCDEF 89AB in the *Flash memory key register (FLASH_KEYR)*.

Any wrong sequence locks up the FLASH_CR registers until the next system reset. In the case of a wrong key sequence, a bus error is detected and a hard fault interrupt is generated.

The FLASH_CR registers can be locked again by software by setting the LOCK bit in one of these registers.

*Note: The FLASH_CR register cannot be written when the BSY bit in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR) is set. Any*

*attempt to write to these registers with the BSY bit set causes the AHB bus to stall until this bit is cleared.*

**Caution:** Due to the structure of the memory (64 bits of data associated to 8 bits of ECC), interrupted operations can result in corrupted data and/or inconsistent ECC bits. The concerned address can trigger an ECC interrupt or an NMI when accessed later on. Consequently, handle write/erase operations with care, and ensure that ECC events are managed by firmware.

### 3.3.7 Flash main memory erase sequences

The erase operation can be performed at page level (page erase), or on the whole memory (mass erase). Mass erase does not affect the Information block (system flash, OTP and option bytes).

#### Flash memory page erase

The CPU1 is able to page erase only the non-secure part of the user flash. The secure CPU2 can page erase both the secure and non-secure parts of the user flash memory.

A page erase starts only when allowed by the PESD bit in the *Flash memory status register (FLASH_SR)* and in the *Flash memory CPU2 status register (FLASH_C2SR)*.

When a page is protected by PCROP or WRP it is not erased.

**Table 5. Page erase overview**

<table>
<tr><th>Page</th><th>PCROP</th><th>WRP</th><th>PCROP_RDP</th><th>Comment</th><th>WRPERR</th><th>CPU1 bus error</th><th>CPU2 bus error</th></tr>
<tr><td rowspan="4">Non secure</td><td rowspan="2">No</td><td>No</td><td rowspan="8">x</td><td>Page is erased</td><td>No</td><td rowspan="4">No</td><td rowspan="4">No</td></tr>
<tr><td>Yes</td><td rowspan="3">Page erase aborted (no page erase started)</td><td rowspan="3">Yes</td></tr>
<tr><td rowspan="2">Yes</td><td>No</td></tr>
<tr><td>Yes</td></tr>
<tr><td rowspan="6">Secure</td><td rowspan="4">No</td><td rowspan="3">No</td><td>Requested by CPU2, secure page is erased</td><td rowspan="3">No</td><td>N/A</td><td>No</td></tr>
<tr><td>Requested by CPU1, secure page erase is aborted (no secure page erase started)</td><td rowspan="4">Yes</td><td>N/A</td></tr>
<tr><td>Yes</td><td rowspan="3">Page erase aborted (no page erase started)</td><td rowspan="3">Yes(1)</td><td rowspan="3">No</td></tr>
<tr><td rowspan="2">Yes</td><td>No</td></tr>
<tr><td>Yes</td></tr>
</table>

1. WRPERR is generated only when PER is requested by CPU2 (Cortex-M0+). There is no WRPERR when PER is requested by CPU1.

To erase a page (4 Kbytes), follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)*.
   Check that flash memory program and erase operations are allowed by checking the PESD bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2*

*status register (FLASH_C2SR)* (these checks are recommended even if status can change due to flash memory operation requests by the other CPU, to limit the risk of receiving a bus error when starting page erase).

2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Check that bit CFGBSY in the FLASH_xxSR register is cleared.
4. Set the PER bit and select the page to erase (PNB) in the *Flash memory control register (FLASH_CR)* or *Flash memory CPU2 control register (FLASH_C2CR)*.
5. Set the STRT bit in the FLASH_xxCR register.
6. Wait until bit CFGBSY is cleared.

*Note: The internal oscillator HSI16 (16 MHz) is enabled automatically when STRT bit is set, and disabled automatically when STRT bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.*

### Flash memory mass erase

A flash memory mass erase by the CPU1 is ignored and a bus error is generated.

When PCROP or WRP is enabled, any flash memory mass erase is aborted and no erase started.

**Table 6. Mass erase overview**

<table>
<tr><th>PCROP</th><th>WRP</th><th>PCROP_RDP</th><th>Comment</th><th>WRPERR</th><th>CPU1 bus error</th><th>CPU2 bus error</th></tr>
<tr><td>No</td><td>No</td><td rowspan="5">x</td><td>Requested by secure CPU2, flash memory is mass erased</td><td>No</td><td rowspan="4">N/A</td><td rowspan="4">No</td></tr>
<tr><td>No</td><td>Yes</td><td rowspan="3">Requested by secure CPU2, mass erase aborted (no erase started)</td><td rowspan="3">Yes</td></tr>
<tr><td>Yes</td><td>No</td></tr>
<tr><td>Yes</td><td>Yes</td></tr>
<tr><td>x</td><td>x</td><td>Requested by CPU1, mass erase aborted (no erase started)</td><td>No</td><td>Yes</td><td>N/A</td></tr>
</table>

To perform a mass erase, follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Check that bit CFGBSY in the FLASH_xxSR register is cleared.
4. Set the MER bit in the *Flash memory control register (FLASH_CR)* or *Flash memory CPU2 control register (FLASH_C2CR)*.
5. Set the STRT bit in the FLASH_xxCR register.
6. Wait until bit CFGBSY is cleared.

*Note: The internal oscillator HSI16 (16 MHz) is enabled automatically when STRT bit is set, and disabled automatically when STRT bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.*

### 3.3.8 Flash main memory programming sequences

The flash memory is programmed 72 bits (a double word, 64 bits plus ECC, 8 bits) at a time.

Programming in a previously programmed double word is only allowed when programming an all 0 value. It is not allowed to program any other value in a previously programmed double word, any attempt sets PROGERR flag in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)*, except when programming an already programmed double word with an all 0 value.

It is only possible to program a double word (2 x 32-bit data).

- Any attempt to write byte (8 bits) or half-word (16 bits) sets SIZERR flag in the FLASH_xxSR register.
- Any attempt to write a double word that is not aligned with a double word address sets PGAERR flag in the FLASH_xxSR register.

Only the secure CPU2 is able to download programming data into the secure part of the memory. A flash memory programming by the CPU1 in the secure flash memory area is ignored and a bus error is generated.

#### Standard programming

The flash memory programming sequence in standard mode is as follows:

1. Check that no flash main memory operation is ongoing by checking the BSY bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)*.
   Check that flash memory program and erase operations are allowed by checking the PESD bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)* (these checks are recommended even if status may change due to flash memory operation requests by the other CPU, to limit the risk of receiving a bus error when starting programming).
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Check that bit CFGBSY in the FLASH_xxSR register is cleared.
4. Set the PG bit in the *Flash memory control register (FLASH_CR)* or *Flash memory CPU2 control register (FLASH_C2CR)*.
5. Perform the data write operation at the desired memory address, inside main memory block or OTP area. Only double word (64 bits) can be programmed.
   a) Write a first word in an address aligned with double word
   b) Write the second word.
6. Wait until bit CFGBSY is cleared.
7. Check that EOP flag is set in the FLASH_xxSR register (meaning that the programming operation has succeeded), and clear it by software.
8. Clear the PG bit in the FLASH_xxSR register if there are no more programming requests anymore.

*Note: When the flash memory interface has received a good sequence (a double word), programming is automatically launched and BSY bit is set. The internal oscillator HSI16 (16 MHz) is enabled automatically when PG bit is set, and disabled automatically when PG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.*

*If the user needs to program only one word, double word must be completed with the erase value 0xFFFF FFFF, to launch automatically the programming.*

*ECC is calculated from the double word to program.*

### Fast programming

This mode allows to program a row or 64 double words (512 bytes). It reduces the page programming time by eliminating the need for verifying the flash memory locations before they are programmed, and avoids rising and falling times of high voltage for each double word. During fast programming, the clock frequency (HCLK4) must be at least 8 MHz.

Only the main memory can be programmed in fast programming mode.

Fast row programming must be performed by executing software from SRAM and disabling interrupts when not relocating the CPU interrupt vector table. A read access from the CPU requesting row programming causes a bus error. A read from any other source (the other CPU or DMA) is stalled until the row programming finishes (standard double word programming does not cause a bus error to the requesting CPU, but stalls any read until standard programming finishes).

The flash main memory programming sequence in standard mode is described below:

1. Perform a mass erase. If not, PGSERR is set.
2. Check that no flash main memory operation is ongoing by checking the BSY bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)*.
   Check that flash memory program and erase operation is allowed by checking the PESD bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)* (these checks are recommended even if status may change due to flash memory operation requests by the other CPU, to limit the risk of receiving a bus error when starting programming).
3. Check and clear all error programming flags due to previous programming.
4. Check that bit CFGBSY in the FLASH_xxSR register is cleared.
5. Set the FSTPG bit in *Flash memory control register (FLASH_CR)* or *Flash memory CPU2 control register (FLASH_C2CR)*.
6. Write the 64 double words to program a row (512 bytes).
7. Wait until bit CFGBSY is cleared.
8. Check that EOP flag is set in the FLASH_xxSR register (meaning that the programming operation has succeed), and clear it by software.
9. Clear the FSTPG bit in the FLASH_xxSR register if there are no more programming requests anymore.

*Note: When attempting to write in fast programming mode while a read operation is on going, the programming is aborted without any system notification (no error flag is set).*

*When the flash memory interface has received the first double word, programming is automatically launched. The BSY bit is set when the high voltage is applied for the first double word, and it is cleared when the last double word has been programmed or in case of error. The internal oscillator HSI16 (16 MHz) is enabled automatically when FSTPG bit is*

*set, and disabled automatically when FSTPG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.*

*The 64 double words must be written successively. The high voltage is kept on the flash memory for all the programming. Maximum time between two double words write requests is the time programming (around 20 µs). If a second double word arrives after this time programming, fast programming is interrupted and MISSERR is set.*

*High voltage must not exceed 8 ms for a full row between two erases. This is guaranteed by the sequence of 64 double words successively written with a clock system greater or equal to 8 MHz. An internal time-out counter counts 7 ms when fast programming is set and stops the programming when time-out is over. In this case the FASTERR bit is set.*

*If an error occurs, high voltage is stopped and next double word to programmed is not programmed. Anyway, all previous double words have been properly programmed.*

### Programming errors signaled by flags

Several kind of errors can be detected. In case of error, the flash memory operation (programming or erasing) is aborted.

- **PROGERR**: Programming error

  In standard programming: PROGERR is set if the word to write with values different from all 0 has not been previously erased, except if the value to program is all 0. If any other error (SIZERR, PAGERR, PGSERR, WRPERR) occurs the PROGRERR may not be set even if there is a word re-programming without erase error.
- **SIZERR**: Size programming error

  In standard programming or in fast programming: only double word can be programmed, and only 32-bit data can be written. SIZERR is set if a byte or an half-word is written.
- **PGAERR**: Alignment programming error

  PGAERR is set if one of the following conditions occurs:
  - Alignment programming errors are individually checked per CPU. No checks are available for simultaneous multi-CPU programming. HSEM or other software mechanisms shall be used to prevent simultaneous multi CPU programming.
  - In standard programming: the first word to be programmed is not aligned with a double word address, or the second word does not belong to the same double word address.
  - In fast programming: the data to program does not belong to the same row than the previous programmed double words, or the address to program is not greater than the previous one.
- **PGSERR**: Programming sequence error

  PGSERR is set if one of the following conditions occurs:
  - In the standard programming sequence or the fast programming sequence: data is written when PG and FSTPG are cleared.
  - In the standard programming sequence or the fast programming sequence: MER and PER are not cleared when PG or FSTPG is set.
  - In the fast programming sequence: the Mass erase is not performed before setting the FSTPG bit.
  - In the mass erase sequence: PG, FSTPG, and PER are not cleared when MER is set.

  - In the page erase sequence: PG, FSTPG, and MER are not cleared when PER is set.
  - PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR, or PGSERR is set due to a previous programming error.
  - In the fast programming sequence: if the row to be programmed is on a page different from the one previously erased. If a mass erase has been done it is allowed to program rows on higher order pages, but not on lower order.
- **WRPERR**: Write protection error

  WRPERR is set if one of the following conditions occurs:
  - Attempt to program or erase in a write protected area (WRP) or in a PCROP area.
  - Attempt to perform a mass erase when one page or more is protected by WRP or PCROP.
  - The debug features are connected or the boot is executed from SRAM or from system flash when the read protection (RDP) is set to Level 1.
  - Attempt to modify the option bytes when the read protection (RDP) is set to Level 2, except when requested by the secure CPU2.
- **MISSERR**: Fast programming data miss error

  In fast programming: all the data must be written successively. MISSERR is set if the previous data programmation is finished and the next data to program is not written yet.
- **FASTERR**: Fast programming error

  In fast programming: FASTERR is set if one of the following conditions occurs:
  - When FSTPG bit is set for more than 7 µs, which generates a time-out detection.
  - When the row fast programming has been interrupted by a MISSERR, PGAERR, WRPERR or SIZERR.

If an error occurs during a program or erase operation, one of the following error flags is set in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)*:

- PROGERR, SIZERR, PGAERR, PGSERR, MISSERR (program error flags)
- WRPERR (protection error flag)

In this case, if the error interrupt enable bit ERRIE is set in the *Flash memory control register (FLASH_CR)* or in the *Flash memory CPU2 control register (FLASH_C2CR)*, an interrupt is generated and the operation error flag OPERR is set in the FLASH_xxSR register.

*Note: If several successive errors are detected (for example, in case of DMA transfer to the flash memory), the error flags cannot be cleared until the end of the successive write request.*

### Programming errors causing a bus error

Some error conditions, listed below, do not generate an error flag but a bus error instead.

- AHB write to any page when RDP level 1 and boot is performed from system flash memory or SRAM1
- AHB write when flash memory is powered down
- Read or write from flash memory through the debugger
- Reading from flash memory when fast row programming is ongoing, from the source which requested the fast row programming.
- Requesting a new programming request, when the previous one has not finished.
- FLASH_CR register write between the two accesses of a double word programming.
- FLASH_CR register write when PESDx is active (set).
- Writing a wrong key in FLASH_KEYR or FLASH_OPTKEYR register
- Any subsequent write to FLASH_KEYR or FLASH_OPTKEYR after unlocking the respective feature

### PGSERR and PGAERR in a page-based row programming

When performing a fast programming *Table 7* describes how PGSERR and PGAERR are handled.

**Table 7. Errors in page-based row programming**

| Last page / row | Current page / row | MER active | PER active |
|---|---|---|---|
| page [x] / row [y] | page [x] / row [y-n] | PGAERR | PGAERR |
| page [x] / row [y] | page [x-n] / row [any] | PGAERR and PGSERR | PGAERR and PGSERR |
| page [x] / row [y] | page [x+n] / row [any] | No error | PGAERR |

When after a system reset neither MER nor PER is performed, any programming attempt causes PGAERR and PGSERR errors.

### Programming and caches

If a flash memory write access impacts data in the data cache, the flash memory write access modifies the data in the memory and the data in the cache.

If an erase operation in flash memory also concerns data in the data or instruction cache, the user has to ensure that these data are rewritten before they are accessed during code execution. Upon an erase operation the cache content is invalidated.

*Note: The I/D cache must be flushed only when it is disabled (I/DCEN = 0).*

## 3.4 FLASH option bytes

### 3.4.1 Option bytes description

The option bytes are configured by the end user depending on the application requirements. As a configuration example, the watchdog may be selected in hardware or software mode (refer to *Section 3.4.2: Option bytes programming*).

A double word is split up in option bytes as indicated in *Table 8*.

**Table 8. Option bytes format**

| 63-56 | 55-48 | 47-40 | 39-32 | 31-24 | 23-16 | 15 -8 | 7-0 |
|---|---|---|---|---|---|---|---|
| Complemented option byte 3 | Complemented option byte 2 | Complemented option byte 1 | Complemented option byte 0 | Option byte 3 | Option byte 2 | Option byte 1 | Option byte 0 |

The organization of these bytes in the information block is shown in *Table 9*. The option bytes can be read from the memory locations listed in *Table 9* or from the Option byte registers:

- *Flash memory option register (FLASH_OPTR)*
- *Flash memory PCROP zone A start address register (FLASH_PCROP1ASR)*
- *Flash memory PCROP zone A end address register (FLASH_PCROP1AER)*
- *Flash memory PCROP zone B start address register (FLASH_PCROP1BSR)*
- *Flash memory PCROP zone B end address register (FLASH_PCROP1BER)*
- *Flash memory WRP area A address register (FLASH_WRP1AR)*
- *Flash memory WRP area B address register (FLASH_WRP1BR)*
- *Flash memory IPCC mailbox data buffer address register (FLASH_IPCCBR)*
- *Secure flash memory start address register (FLASH_SFR)*
- *Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)*

**Table 9. Option bytes organization**

| Address[1] | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1FFF8000 | AGC_TRIM | | | Unused | nBOOT0 | nSWBOOT0 | SRAM2_RST | SRAM2_PE | nBOOT1 | Unused | | | WWDG_SW | IWDG_STBY | IWDG_STOP | IWDG_SW | Unused | nRST_SHDW | nRST_STBY | nRST_STOP | BOR_LEV | | | ESE | RDP | | | | | | | |
| 1FFF8008 | Unused | | | | | | | | | | | | | | | | | | | | | | | PCROP1A_STRT | | | | | | | | |
| 1FFF8010 | PCROP_RDP | Unused | | | | | | | | | | | | | | | | | | | | | | PCROP1A_END | | | | | | | | |
| 1FFF8018 | Unused | | | | | | | | WRP1A_END | | | | | | | | Unused | | | | | | | | WRP1A_STRT | | | | | | | |
| 1FFF8020 | Unused | | | | | | | | WRP1B_END | | | | | | | | Unused | | | | | | | | WRP1B_STRT | | | | | | | |

**Table 9. Option bytes organization (continued)**

| Address[1] | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1FFF8028 | Unused | | | | | | | | | | | | | | | | | | | | | | | PCROP1B_STRT | | | | | | | | |
| 1FFF8030 | Unused | | | | | | | | | | | | | | | | | | | | | | | PCROP1B_END | | | | | | | | |
| 1FFF8038 to 1FFF8060 | Unused | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1FFF8068 | Unused | | | | | | | | | | | | | | | | | | IPCCDBA | | | | | | | | | | | | | |
| 1FFF8070 | Unused | | | | | | | | | | | | | | | | | | | DDS | Unused | | | FSD | SFSA | | | | | | | |
| 1FFF8078 | C2OPT | NBRSD | SNBRSA | | | | | Unused | BRSD | SBRSA | | | | | SBRV | | | | | | | | | | | | | | | | | |
| 1FFF8FF8 | OPVAL[2] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

1. The upper 32-bits of the double-word address contain the inverted data from the lower 32 bits.
2. The value 0x0BC0DE1D is written during production, when option bytes are configured.

### User and read protection option bytes

Flash memory address: 0x1FFF 8000

Reset value: 0x3FFF F1AA (ST production value)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AGC_TRIM[2:0] | | | Res. | n BOOT0 | nSW BOOT0 | SRAM2 _RST | SRAM2 _PE | n BOOT1 | Res. | Res. | Res. | WWDG _SW | IWDG _STDBY | IWDG _STOP | IWDG _SW |
| 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. | nRST _SHDW | nRST _STDBY | nRST _STOP | BOR_LEV[2:0] | | | ESE | RDP[7:0] | | | | | | | |
| | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:29 **AGC_TRIM[2:0]**: Automatic gain control trimming.
Default value 0b001.

Bit 28 Reserved, must be kept at reset value.

Bit 27 **nBOOT0:** nBOOT0 option bit
0: nBOOT0 = 0
1: nBOOT0 = 1

Bit 26 **nSWBOOT0:** Software BOOT0
0: BOOT0 taken from the option bit nBOOT0
1: BOOT0 taken from PH3/BOOT0 pin

Bit 25 **SRAM2_RST:** SRAM2 and PKA RAM erase when system reset
0: SRAM2 and PKA RAM erased when a system reset occurs
1: SRAM2 and non-secure PKA RAM not erased when a system reset occurs

Bit 24 **SRAM2_PE:** SRAM2 parity check enable
0: SRAM2 parity check enabled
1: SRAM2 parity check disabled

Bit 23 **nBOOT1:** Boot configuration

Together with the BOOT0 pin or option bit nBOOT0 (depending on nSWBOOT0 option bit configuration), this bit selects boot mode from the flash main memory, SRAM1 or the System memory. Refer to *Section 2.3: Boot configuration*.

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 **WWDG_SW:** Window watchdog selection

0: Hardware window watchdog
1: Software window watchdog

Bit 18 **IWDG_STDBY:** Independent watchdog counter freeze in Standby mode

0: Independent watchdog counter is frozen in Standby mode
1: Independent watchdog counter is running in Standby mode

Bit 17 **IWDG_STOP:** Independent watchdog counter freeze in Stop mode

0: Independent watchdog counter is frozen in Stop mode
1: Independent watchdog counter is running in Stop mode

Bit 16 **IWDG_SW:** Independent watchdog selection

0: Hardware independent watchdog
1: Software independent watchdog

Bit 15 Reserved, must be kept at reset value.

Bit 14 **nRST_SHDW**

0: Reset generated when entering the Shutdown mode
1: No reset generated when entering the Shutdown mode

Bit 13 **nRST_STDBY**

0: Reset generated when entering the Standby mode
1: No reset generate when entering the Standby mode

Bit 12 **nRST_STOP**

0: Reset generated when entering the Stop mode
1: No reset generated when entering the Stop mode

Bits 11:9 **BOR_LEV[2:0]:** BOR reset Level

These bits contain the VDD supply level threshold that activates/releases the reset.

000: BOR Level 0. Reset level threshold is ~ 1.7 V
001: BOR Level 1. Reset level threshold is ~ 2.0 V
010: BOR Level 2. Reset level threshold is ~ 2.2 V
011: BOR Level 3. Reset level threshold is ~ 2.5 V
100: BOR Level 4. Reset level threshold is ~ 2.8 V

Bit 8 **ESE:** System security enabled flag.

0: Security disabled
1: Security enabled

Bits 7:0 **RDP[7:0]:** Read protection level

0xAA: Level 0, read protection not active
0xCC: Level 2, chip read protection active
Others: Level 1, memories read protection active

### PCROP1A start address option bytes

Flash memory address: 0x1FFF 8008

Reset value: 0xFFFF FFFF, (ST production value)

| 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. | PCROP1A_STRT[8:0] | | | | | | | | |
| | | | | | | | r | r | r | r | r | r | r | r | r |

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

Bits 8:0 **PCROP1A_STRT[8:0]:** PCROP1A area start offset

PCROP1A_STRT contains the first 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB30CE devices.

### PCROP1A end address option bytes

Flash memory address: 0x1FFF 8010

Reset value: 0x0000 0000 (ST production value)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PCROP_RDP | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | PCROP1A_END[8:0] | | | | | | | | |
| | | | | | | | r | r | r | r | r | r | r | r | r |

Bit 31 **PCROP_RDP:** PCROP area preserved when RDP level is decreased

This bit is set only. It is reset after a full mass erase due to a change of RDP from Level 1 to Level 0.

0: PCROP area is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:9 Reserved, must be kept at reset value.

Bits 8:0 **PCROP1A_END:** PCROP1A area end offset

PCROP1A_END contains the last 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB30CE devices.

### WRP Area A address option bytes

Flash memory address: 0x1FFF 8018

Reset value: 0x0000 00FF (ST production value)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1A_END[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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1A_STRT[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 23:16 **WRP1A_END[7:0]:** WRP area “A” end offset
WRPA1_END contains the last 4 Kbytes page of the WRP area “A”. Note that bit 23 is reserved on STM32WB30CE devices.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **WRP1A_STRT[7:0]:** WRP area “A” start offset
WRPA1_STRT contains the first 4 Kbytes page of the WRP area “A”. Note that bit 7 is reserved on STM32WB30CE devices.

### WRP Area B address option bytes

Flash memory address: 0x1FFF 8020

Reset value: 0x0000 00FF (ST production value)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1B_END[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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1B_STRT[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 23:16 **WRP1B_END[7:0]:** WRP area “B” end offset
WRPB1_END contains the last 4 Kbytes page of the WRP area “B”. Note that bit 23 is reserved on STM32WB30CE devices.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **WRP1B_STRT[7:0]:** WRP area “B” start offset
WRPB1_STRT contains the first 4 Kbytes page of the WRP area “B”. Note that bit 7 is reserved on STM32WB30CE devices.

### PCROP1B start address option bytes

Flash memory address: 0x1FFF 8028

Reset value: 0xFFFF FFFF (ST production value)

| 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. | PCROP1B_STRT[8:0] | | | | | | | | |
| | | | | | | | r | r | r | r | r | r | r | r | r |

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

Bits 8:0 **PCROP1B_STRT[8:0]:** PCROP1B area start offset
PCROP1B_STRT contains the first 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB30CE devices.

### PCROP1B end address option bytes

Flash memory address: 0x1FFF 8030

Reset value: 0x0000 0000 (ST production value)

| 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. | PCROP1B_END[8:0] | | | | | | | | |
| | | | | | | | r | r | r | r | r | r | r | r | r |

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

Bits 8:0 **PCROP1B_END[8:0]:** PCROP1B area end offset

PCROP1B_END contains the last 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB30CE devices.

### IPCC mailbox data buffer address option bytes

Flash memory address: 0x1FFF 8068

Reset value: 0x0000 0000 (ST production value)

| 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. | IPCCDBA[13:0] | | | | | | | | | | | | | |
| | | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

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

Bits 13:0 **IPCCDBA[13:0]:** IPCC mailbox data buffer base address offset

IPCCDBA contains the first double-word offset of the IPCC mailbox data buffer area in SRAM2.

### Secure flash memory start address option bytes

Flash memory address: 0x1FFF 8070

Reset value: 0xFFFF FEXX (ST production value)

| 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. | DDS | Res. | Res. | Res. | FSD | SFSA[7:0] | | | | | | | |
| | | | r | | | | r | r | r | r | r | r | r | r | r |

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

Bit 12 **DDS:** Disable CPU2 debug access
0: CPU2 debug access enabled.
1: CPU2 debug access disabled.

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

Bit 8 **FSD:** Flash security disable
FSD = 1: System and flash non-secure
FSD = 0: System and flash secure. SFSA[7:0] contains the start address of the first 4 Kbytes page of the secure flash area.

Bits 7:0 **SFSA[7:0]:** Secure flash start address
System and flash secure. SFSA[7:0] contains the start address of the first 4 Kbytes page of the secure flash area.

### Secure SRAM2 start address and CPU2 reset vector option bytes

Flash memory address: 0x1FFF 8078

Reset value: 0xXXXX XXXX (ST production value)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C2OPT | NBRSD | SNBRSA[4:0] | | | | | Res. | BRSD | SBRSA[4:0] | | | | | SBRV[17:16] | |
| 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 |
| SBRV[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31 **C2OPT:** CPU2 boot reset vector memory selection.
0: SBRV offset addresses SRAM1 or SRAM2, from start address 0x2000 0000 (SBRV value must be kept within the SRAM area).
1: SBRV offset addresses flash memory, from start address 0x0800 0000.

Bit 30 **NBRSD:** Non-backup SRAM2b security disable
NBRSD = 1: SRAM2b is non-secure
NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.

Bits 29:25 **SNBRSA:** Secure non-backup SRAM2b start address
NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **BRSD:** Backup SRAM2a security disable
BRSD = 1: SRAM2a is non-secure
BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.

Bits 22:18 **SBRSA:** Secure backup SRAM2a start address
BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.

Bits 17:0 **SBRV[17:0]:** CPU2 boot reset vector
Contains the word aligned CPU2 boot reset start address offset within the selected memory area by C2OPT.

### 3.4.2 Option bytes programming

After reset, the options related bits in the *Flash memory control register (FLASH_CR)* and *Flash memory CPU2 control register (FLASH_C2CR)* are write-protected. To run any operation on the option bytes page, the option lock bit OPTLOCK in the *Flash memory control register (FLASH_CR)* must be cleared. The following sequence is used to unlock this register:

1. Unlock the FLASH_CR with the LOCK clearing sequence (refer to *Unlocking the flash memory*)
2. Write OPTKEY1= 0x08192A3B in the *Flash memory option key register (FLASH_OPTKEYR)*
3. Write OPTKEY2 = 0x4C5D6E7F in the *Flash memory option key register (FLASH_OPTKEYR)*

Any wrong sequence locks up the flash memory option registers until the next system reset. In the case of a wrong key sequence, a bus error is detected and a hard fault interrupt is generated.

The user options can be protected against unwanted erase/program operations by setting the OPTLOCK bit by software.

*Note: If LOCK is set by software, OPTLOCK is automatically set as well.*

*Note: In a multi-CPU system it is good practice to use semaphores to manage option programming, and prevent simultaneous option programming by the CPUs.*

#### Modifying user options

The option bytes are programmed differently from a main memory user address.

To modify the value of user options follow the procedure below:

1. Clear OPTLOCK option lock bit with the clearing sequence described above
2. Write the desired options value in the options registers.
3. Check that no flash memory operation is on going by checking the BSY bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)*.
   Check that flash program and erase operation is allowed by checking the PESD bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)* (these checks are recommended even if status may change due to flash operation requests by the other CPU, to limit the risk of receiving a bus error when modifying user options).
4. Set the Options start bit OPTSTRT in the *Flash memory control register (FLASH_CR)*.
5. Wait for the BSY bit to be cleared.

*Note: Any modification of the value of one option is automatically performed by erasing user option bytes pages first, and then programming all the option bytes with the values contained in the flash memory option registers.*

#### Secure user options

The secure option bytes *Secure flash memory start address register (FLASH_SFR)* and *Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)* can only be written by the secure CPU2.

### Option byte loading

After the BSY bit is cleared, all new options are updated into the flash memory, but not applied to the system. A read from the option registers returns the last loaded option byte values, the new options have effect on the system only after they are loaded.

Option bytes loading is performed in two cases:

- when OBL_LAUNCH bit is set in the *Flash memory control register (FLASH_CR)*
- after a power reset (BOR reset or exit from Standby/Shutdown modes)

Option byte loader performs a read of the options block and stores the data into internal option registers. These internal registers configure the system and can be read by software. Setting OBL_LAUNCH generates a reset so the option byte loading is performed under system reset.

Each option bit has also its complement in the same double word. During option loading, a verification of the option bit and its complement allows to check the loading has correctly taken place.

During option byte loading, the options are read by double word. ECC on option words is not taken into account during OBL, but only during direct SW read of option area.

If the word and its complement are matching, the option word/byte is copied into the option register.

If the comparison between the word and its complement fails, a status bit OPTVERR is set. Mismatch values are forced into the option registers:

- for USR OPT option, the value of mismatch is all options at ‘1’, except for BOR_LEV that is “000” (lowest threshold)
- for WRP option, the value of mismatch is the default value “No protection”
- for RDP option, the value of mismatch is the default value “Level 1”
- for PCROP, the value of mismatch is “all memory protected”
- for FSD and SFSA option, the value of mismatch is “all memories (Flash, SRAM2a and SRAM2b) secured”
- for BRSD, SBRSA, NBRSD and SNBRASA options, the value of mismatch is “SRAM2 memory part secured”
- for DDS option, the value of mismatch is “CPU2 debug disabled”
- for C2OPT and SBRV options, the value of mismatch is “CPU2 boot from start address of last Flash page (safe boot)”
- for OPTVAL option, the value of mismatch is “not valid”.

If the OPTVAL option indicates “not valid” all memories (Flash, SRAM2a and SRAM2b) are secure.

**Table 10. Option loading control**

| OPTVERR | OPTNV | Description |
|---|---|---|
| 0 | 0 | Options correctly loaded and OPTVAL is “valid”.<br>– Security applied according to options. |
| 0 | 1 | Does not occur |

**Table 10. Option loading control (continued)**

| OPTVERR | OPTNV | Description |
|---|---|---|
| 1 | 0 | OPTVAL option correctly loaded as “valid”, but some or all other option and engineering bits corrupted, mismatch values loaded.<br>– When secure option is loaded correctly, security is applied according to the loaded secure option values.<br>– When secure option is corrupted, security is applied on the full memory as indicated by the loaded mismatch value. |
| 1 | 1 | Some or all option and engineering bits corrupted, mismatch values loaded. OPTVAL correctly loaded as “not valid”. Security applied on full memories irrespective of the loaded secure option values. |

On system reset rising, internal option registers are copied into option registers that can be read and written by software:

- FLASH_OPTR
- FLASH_PCROPxySR (x = 1 and y = A or B)
- FLASH_PCROPxyER (x = 1 and y = A or B)
- FLASH_WRPxyR (x = 1 and y = A or B)
- FLASH_IPCCDBA
- FLASH_SFR
- FLASH_SRRVA

These registers are also used to modify options. If these registers are not modified by user, they reflect the options states of the system. See *Modifying user options* for more details.

## 3.5 FLASH UID64

A 64-bit unique device identification (UID64) is stored in the flash memory, and can be accessed by the CPUs.

**Table 11. UID64 organization**

| Address | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x1FFF 7580 | Unique device number | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1FFF 7584 | ST company ID | | | | | | | | | | | | | | | | | | | | | | | | Device ID | | | | | | | |

The UID64 is programed at device production, and provides a unique code for each device

- the 24-bit ST company ID returns 0x00 80 E1
- the 8-bit device ID returns 0x26 for STM32WB50CG revision Y and STM32WB30CE revision A, 0x27 for STM32WB50CG/STM32WB30CE revision X, as indicated in DBGMCU_IDCODE
- the 32-bit unique device number is a sequential number, different for each individual device.

## 3.6 Flash memory protection

The flash main memory can be protected against external accesses with the Read protection (RDP). The pages can also be protected against unwanted write (WRP) due to loss of program counter context. The write-protection WRP granularity is 4 Kbytes. Apart from the RDP and WRP, flash memory can also be protected against read and write from third parties (PCROP). The PCROP granularity is 2 Kbytes.

Part of the flash main memory can be secured. It grants exclusive access to this part of the memory to the CPU2.

### 3.6.1 Read protection (RDP)

The read protection is activated by setting the RDP option byte and then, by applying a system reset to reload the new RDP option byte. The read protection protects the flash main memory, the option bytes, the backup registers (RTC_BKPxR in the RTC) and the SRAM2.

*Note: If the read protection is set while the debugger is still connected (or was connected since the last power on) through JTAG/SWD, apply a POR (power-on reset) instead of a system reset. If the read protection is programmed through software, do not set the OBL_LAUNCH bit (FLASH_CR register), but perform a POR to reload the option byte. This can be done with a transition to Standby (or Shutdown) mode, followed by a wake-up.*

There are three levels of read protection from no protection (Level 0) to maximum protection or no debug (Level 2).

The flash memory is protected when the RDP option byte and its complement contain the pair of values shown in *Table 12*.

**Table 12. Flash memory read protection status**

| RDP byte value | RDP complement value | Read protection level |
|---|---|---|
| 0xAA | 0x55 | Level 0 |
| Any value except 0xAA or 0xCC | Any value (not necessarily complementary), except 0x55 and 0x33 | Level 1 (default) |
| 0xCC | 0x33 | Level 2 |

The System memory area is read accessible whatever the protection level. It is never accessible for program/erase operation.

#### Level 0: No protection

Read, program and erase operations into the flash main memory area are possible. The option bytes, the SRAM2 and the backup registers are also accessible by all operations.

### Level 1: Read protection

This is the default protection level when RDP option byte is erased. It is defined as well when RDP value is at any value different from 0xAA and 0xCC, or even if the complement is not correct.

- **User mode:** Code executing in user mode (**Boot flash**) can access flash main memory, option bytes, SRAM2 and backup registers with all operations.
- **Debug, boot RAM and boot loader modes:** In debug mode or when code is running from boot RAM or boot loader, the flash main memory, the backup registers (RTC_BKPxR in the RTC) and the SRAM2 are totally inaccessible. In these modes, a read or write access to the flash memory generates a bus error and a hard fault interrupt.

**Caution:** In case the Level 1 is configured and no PCROP areas are defined, it is mandatory to set PCROP_RDP bit to 1 (full mass erase when the RDP level is decreased from Level 1 to Level 0). In case the Level 1 is configured and a PCROP area is defined, if user code needs to be protected by RDP but not by PCROP, it must not be placed in a page containing a PCROP area.

### Level 2: No debug

In this level, the protection Level 1 is guaranteed. In addition, the CPU1 and CPU2 debug port, the boot from RAM (boot RAM mode) and the boot from System memory (boot loader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the flash main memory. On the contrary, only read and secure write operations can be performed on the option bytes. Option bytes, can only be programmed and erased by a secure CPU2.

The Level 2 cannot be removed from the non-secure application side: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag WRPERR is set in the FLASH_xxxSR register and an interrupt can be generated.

*Note: The debug feature is also disabled under reset.*

*Note: STMicroelectronics is not able to perform analysis on defective parts on which the Level 2 protection has been set.*

### Changing the Read protection level

It is easy to move from Level 0 to Level 1 by changing the value of the RDP byte to any value (except 0xCC). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 either directly from Level 0 or from Level 1. Once in Level 2 it is no more possible to modify the Read protection level.

When the RDP is reprogrammed to the value 0xAA to move from Level 1 to Level 0, a mass erase of the flash main memory is performed if PCROP_RDP is set in the *Flash memory PCROP zone A end address register (FLASH_PCROP1AER)*. The backup registers (RTC_BKPxR in the RTC), the SRAM2 and the PKA SRAM are also erased. The user options (except PCROP protection) are set to their previous values copied from FLASH_OPTR, FLASH_WRPxyR (x = 1 and y = A or B). PCROP is disabled. The OTP area is not affected by mass erase and remains unchanged.

If the bit PCROP_RDP is cleared in the FLASH_PCROP1AER, the full mass erase is replaced by a partial mass erase that is successive page erases, except for the pages protected by PCROP. This is done in order to keep the PCROP code. Only when the flash

memory is erased, options are re-programmed with their previous values. This is also true for FLASH_PCROPxySR and FLASH_PCROPxyER registers (x = 1 and y = A or B).

A requested mass erase performs a partial mass erase (a sequence of page erases), except for the pages protected by CPU2 security (SFSA). This is done to keep the CPU2 secure code.

**Table 13. RDP regression from Level 1 to Level 0 and memory erase**

<table>
<tr><th>SFSA</th><th>PCROP</th><th>PCROP_RDP</th><th>Comment</th></tr>
<tr><td rowspan="4">Partial</td><td>None</td><td>x</td><td rowspan="2">Flash memory multiple page erase of all non-secure pages, SRAM2, PKA RAM, and backup registers erase (secure flash pages conserved).</td></tr>
<tr><td rowspan="2">Partial</td><td>1</td></tr>
<tr><td rowspan="2">0</td><td>Flash memory multiple page erase of all non-PCROP pages and non-secure pages, SRAM2, PKA RAM, and backup registers erase (PCROP flash memory pages and secure flash memory pages conserved).</td></tr>
<tr><td>Complete non-secure</td><td>Flash memory, SRAM2, and PKA RAM and backup registers are conserved.</td></tr>
<tr><td>Complete flash memory</td><td>x</td><td>x</td><td>Flash memory, SRAM2, and PKA RAM and backup registers are conserved.</td></tr>
</table>

*Note: Partial mass erase is performed only when Level 1 is active and Level 0 requested. When the protection level is increased (0→1, 1→2, 0→2) there is no mass erase.*

*To validate the protection level change, the option bytes must be reloaded through the OBL_LAUNCH bit in flash memory control register.*

**Figure 4. Changing the Read protection (RDP) level**

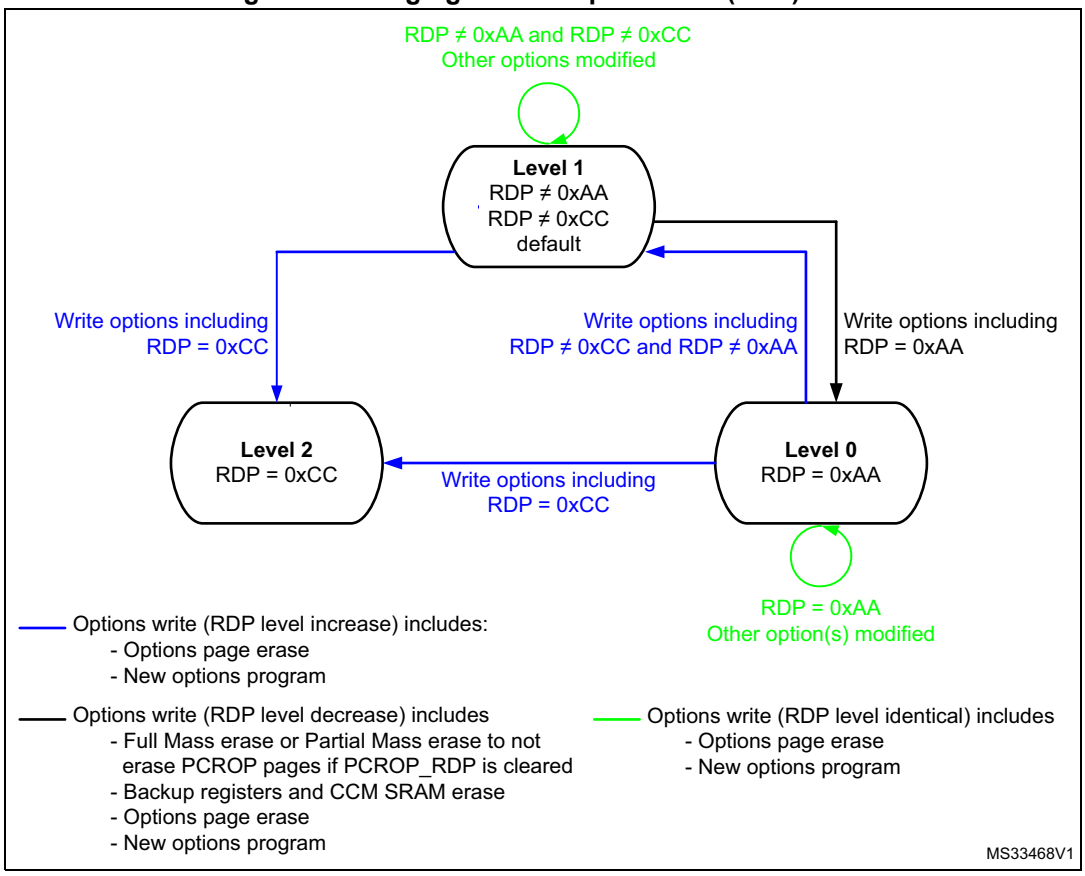



**Table 14. Access status vs. protection level and execution modes**

| Area | Protection level | User execution (BootFromFlash memory) | | | Debug / BootFromRam / BootFromLoader | | |
|---|---|---|---|---|---|---|---|
| | | Read | Write | Erase | Read | Write | Erase |
| Flash main memory | 1 | Yes | Yes | Yes | No | No | No(4) |
| | 2 | Yes | Yes | Yes | N/A(1) | N/A(1) | N/A(1) |
| System memory(2) | 1 | Yes | No | No | Yes | No | No |
| | 2 | Yes | No | No | N/A(1) | N/A(1) | N/A(1) |
| Option bytes | 1 | Yes | Yes(3) | Yes | Yes | Yes(3) | Yes |
| | 2 | Yes | CPU1 and CPU2 none secure - No<br>CPU2 secure - Yes | CPU1 and CPU2 none secure - No<br>CPU2 secure - Yes | N/A(1) | N/A(1) | N/A(1) |
| Backup registers | 1 | Yes | Yes | N/A | No | No | No(4) |
| | 2 | Yes | Yes | N/A | N/A(1) | N/A(1) | N/A(1) |
| SRAM2 | 1 | Yes | Yes | N/A | No | No | No(5) |
| | 2 | Yes | Yes | N/A | N/A(1) | N/A(1) | N/A(1) |

1. When the protection Level 2 is active, the Debug port, the boot from RAM and the boot from system memory are disabled.
2. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.

3. The flash non secure main memory is erased when the RDP option byte is programmed with all level of protections disabled (0xAA). The flash secure main memory is also erased when the SFSA option byte is programmed to be non-secure and at the same time the RDP option byte is programmed with all level protections disabled (0xAA).
4. The backup registers are erased when RDP changes from Level 1 to Level 0.
5. The SRAM2 is erased when RDP changes from Level 1 to Level 0.

### 3.6.2 Proprietary code readout protection (PCROP)

Two parts of the flash memory can be protected against read and write from third parties.

The protected area is execute-only: it can only be reached by the STM32 CPUs, with an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. The PCROP areas have a 2 Kbytes granularity. An additional option bit (PCROP_RDP) makes it possible to select if the PCROP area is erased or not when the RDP protection is changed from Level 1 to Level 0 (refer to *Changing the Read protection level*).

Each PCROP area is defined by a start page offset and an end page offset into the flash memory. These offsets are defined in the PCROP address registers *Flash memory PCROP zone A start address register (FLASH_PCROP1ASR)*, *Flash memory PCROP zone A end address register (FLASH_PCROP1AER)*, *Flash memory PCROP zone B start address register (FLASH_PCROP1BSR)* and *Flash memory PCROP zone B end address register (FLASH_PCROP1BER)*.

A PCROP area is defined from the address *Flash memory Base address + [PCROPxy_STRT x 0x800]* (included) to the address: *Flash memory Base address + [(PCROPxy_END+1) x 0x800]* (excluded). The minimum PCROP area size is two PCROP pages (4 Kbytes) *PCROPxy_END = PCROPxy_STRT + 1*.

When *PCROPxy_END = PCROPxy_STRT* the full flash memory is PCROP protected.

For example, to protect by PCROP from the address 0x0806 2F80 (included) to the address 0x0807 0004 (included):

- if boot in flash memory is selected, one of the FLASH_PCROPxySR and FLASH_PCROPxyER registers (x = 1 and y = A or B) must be programmed with:
  - PCROPxy_STRT = 0xC5 (PCROP area first address 0x0806 2800)
  - PCROPxy_END = 0xE0 (PCROP area last address 0x0807 07FF)

Any data read access performed through a PCROP protected area triggers the RDERR flag error.

Any PCROP protected address is also write protected and any write access to one of these addresses triggers WRPERR.

Any PCROP area is also erase protected. Consequently, any erase to a page in this zone is impossible (including the page containing the start address and the end address of this zone). Moreover, a software mass erase cannot be performed if one zone is PCROP protected.

In the previous example, due to erase by page, all pages from page 0xC5 to 0xE0 are protected in case of page erase, all addresses from 0x0806 2800 to 0x0807 07FF cannot be erased.

Deactivation of PCROP can only occur when the RDP is changing from Level 1 to Level 0. If the user options modification tries to clear PCROP or to decrease the PCROP areas, the

options programming is launched but PCROP areas stays unchanged. On the contrary, it is possible to increase the PCROP areas.

When option bit PCROP_RDP is cleared, and when the RDP is changing from Level 1 to Level 0, Full mass erase is replaced by Partial mass erase to preserve the PCROP area (refer to *Changing the Read protection level*). In this case, PCROPxy_STRT and PCROPxy_END (x = 1 and y = A or B) are not erased.

**Table 15: PCROP protection**

| PCROP registers values (x = 1 and y = A or B) | PCROP protection area |
|---|---|
| PCROPxy_STRT = PCROPxy_END | The full flash memory is PCROP protected |
| PCROPxy_STRT > PCROPxy_END | No PCROPxy, unprotected |
| PCROPxy_STRT < PCROPxy_END | Pages from PCROPxy_STRT to PCROPxy_END are protected |

*Note: It is recommended to align PCROP areas with page granularity when using PCROP_RDP, or to leave free the rest of the page where PCROP zones start or end.*

### 3.6.3 Write protection (WRP)

The user area in flash memory can be protected against unwanted write operations. Two write-protected (WRP) areas can be defined, with page (4 Kbytes) granularity. Each area is defined by a start page offset and an end page offset related to the physical flash memory base address. These offsets are defined in the WRP address registers *Flash memory WRP area A address register (FLASH_WRP1AR)* and *Flash memory WRP area B address register (FLASH_WRP1BR)*.

The WRP “y” area (y = A, B) is defined from the address *Flash memory Base address + [WRP1y_STRT x 0x1000]* (included) to the address *Flash memory Base address + [(WRP1y_END+1) x 0x1000]* (excluded). The minimum WRP area size is one WRP page (4 Kbytes), *WRP1y_END = WRP1y_STRT*.

For example, to protect by WRP from the address 0x0806 2000 (included) to the address 0x0807 1FFF (included):

- If boot in flash memory is selected, FLASH_WRP1AR register must be programmed with:
  - WRP1A_STRT = 0x62.
  - WRP1A_END = 0x71.

  WRP1B_STRT and WRP1B_END in FLASH_WRP1BR can be used instead (area “B” in flash memory).

When WRP is active, it cannot be erased or programmed. Consequently, a software mass erase cannot be performed if one area is write-protected.

If an erase/program operation to a write-protected part of the flash memory is attempted, the write protection error flag (WRPERR) is set in the FLASH_SR register. This flag is also set for any write access to:

- OTP area
- part of the flash memory that can never be written like the ICP
- PCROP area.

*Note: When the flash memory read protection level is selected (RDP level = 1), it is not possible to program or erase the memory if the CPU debug features are connected (JTAG or single wire) or boot code is being executed from RAM or system flash memory, even if WRP is not activated. Any attempt generates an hard fault (BusFault).*

**Table 16: WRP protection**

| WRP registers values (x = 1 and y = A or B) | WRP protection area |
|---|---|
| WRPxy_STRT = WRPxy_END | Page WRPxy is protected |
| WRPxy_STRT > WRPxy_END | No WRP, unprotected |
| WRPxy_STRT < WRPxy_END | Pages from WRPxy_STRT to WRPxy_END are protected |

*Note: To validate the WRP options, the option bytes must be reloaded through the OBL_LAUNCH bit in flash memory control register.*

### 3.6.4 CPU2 security (ESE)

All or a part of the flash memory and the SRAM2a and SRAM2b can be made secure, exclusively accessible by the CPU2, protected against execution, read and write from third parties. Only the CPU2 can execute, read and write in these areas. It can only be reached by the CPU2, while all other accesses (CPU1 and DMA) are strictly prohibited.

#### Changing the CPU2 security mode

CPU2 security start address can be modified by the secure CPU2 by loading a new user option SFSA.

#### CPU2 secure flash memory area

The CPU2 secure flash memory area has sector (4 Kbytes) granularity and is defined by the secure flash memory start page offset user option (SFSA) into the flash memory. This offset is controlled from the SFSA field in the *Secure flash memory start address register (FLASH_SFR)*.

The CPU2 secure flash memory area is defined as follows: *Flash memory Base address + [SFSA x 0x1000] (included)* to the last flash memory address: When CPU2 security is enabled, the minimum CPU2 secure area size is one sector (4 Kbytes).

For example, a CPU2 secure area from the address 0x080E 7000 (included) to the address 0x080F FFFF (included):

- FLASH_SFR registers must be programmed with:
  - SFSA= 0x0E7.

A flag (ESE) is available from the *Flash memory option register (FLASH_OPTR)* informing that CPU2 security is enabled.

Any CPU1 access to a CPU2 security area triggers RDERR or WRPERR flag error.

#### CPU2 secure SRAM2 areas

The CPU2 secure SRAM2a and SRAM2b areas have a 1 Kbyte granularity and are defined by the secure backup RAM (SRAM2a) start address user options (BRSD and SBRSA) and the secure non-backup RAM (SRAM2b) start address user options (NBRSD and SNBRSA) into the flash memory. These offset are controlled by the SBRSA and SNBRSA fields in the

*Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)*.

The CPU2 secure SRAM2a area is defined as *Backup SRAM2a Base address + [SBRSA x 0x0400]* (included) to the last SRAM2a address.

As an example, for a CPU2 secure SRAM2a area from the address 0x2003 5000 (included) to the address 0x2003 7FFF (included) FLASH_SRRVR registers must be programmed with SBRSA = 0x14.

Any CPU1 read access returns no data, and a write access to a CPU2 security SRAM2a area is discarded and triggers a bus error.

When BRSD is set to 1 the SRAM2a is non-secure.

The CPU2 secure non-backup SRAM2b area is defined as *Non-backup SRAM2b base address + [SNBRSA x 0x0400]* (included) to the last SRAM2b address.

As an example, for a CPU2 secure SRAM2b area from the address 0x2003 EC00 (included) to the address 0x2003 FFFF (included) FLASH_SRRVR registers must be programmed with SNBRSA = 0x1B.

Any CPU1 read access returns no data, and a write access to a CPU2 security SRAM2b area is discarded and triggers a bus error.

When NBRSD is set to 1 the SRAM2b is non-secure.

### CPU2 debug access

Debug access to the CPU2 is disabled, as indicated by the DDS filed in the *Secure flash memory start address register (FLASH_SFR)*. The debugger has no access to the CPU2 and the secure peripherals and memory areas.

## 3.7 FLASH program/erase suspension

Flash memory program and erase operations can be suspended by setting the PES bit in the *Flash memory access control register (FLASH_ACR)* or *Flash memory CPU2 access control register (FLASH_C2ACR)*. This feature is useful when executing time critical sections by a CPU. It makes it possible to suspend any new program or erase operation from being started, preventing CPU instruction and data fetches from being blocked.

When at least one PES bit is set:

- Any ongoing program or erase operation is completed.
  - The maximum latency for a flash memory program erase suspension is the maximum time for one program or erase operation to complete (see the device datasheet for more information on the program and erase timing).
- All new requested program and erase operations are not started, but suspended.
  - PESD bits in the *Flash memory status register (FLASH_SR)* and *Flash memory CPU2 status register (FLASH_C2SR)* are set as soon as any PES is set, no matter if a program/erase is currently suspended. This allows a CPU to test PESD prior to requesting a program or an erase operation.

When all PES bits are reset to 0:

- A suspended program or erase operation is started.
  - The PESD bit in both the *Flash memory status register (FLASH_SR)* and in the *Flash memory CPU2 status register (FLASH_C2SR)* are cleared.

## 3.8 FLASH interrupts

**Table 17. Flash memory interrupt requests**

| Interrupt event | Event flag | Event flag/interrupt clearing method | Interrupt enable control bit |
|---|---|---|---|
| End of operation | EOP(1) | Write EOP = 1 | EOPIE |
| Operation error | OPERR(2) | Write OPERR = 1 | ERRIE |
| Read protection error | RDERR | Write RDERR = 1 | RDERRIE |
| Write protection error | WRPERR | Write WRPERR = 1 | N/A |
| Size error | SIZERR | Write SIZERR = 1 | N/A |
| Programming sequential error | PROGERR | Write PROGERR = 1 | N/A |
| Programming alignment error | PGAERR | Write PGAERR = 1 | N/A |
| Programming sequence error | PGSERR | Write PGSERR = 1 | N/A |
| Data miss during fast programming error | MISSERR | Write MISSERR = 1 | N/A |
| Fast programming error | FASTERR | Write FASTERR = 1 | N/A |
| ECC error correction | ECCC | Write ECCC = 1 | ECCCIE |
| ECC double error (NMI) | ECCD | Write ECCD = 1 | N/A |

1. EOP is set only if EOPIE is set.
2. OPERR is set only if ERRIE is set.

*Note: The flash interface provides only a single interrupt line to both CPUs. It is good practice to manage flash operations between the CPUs using a semaphore. To avoid receiving unwanted flash operation interrupts, the CPU must mask the flash interrupt in the NVIC or SYSCFG pre-mask.*

## 3.9 Register access protection

The user options registers can be protected by security.

The FLASH secure registers (FSD, SFSA, BRSD, SBRSA, NBRSD, SNBRSA, SBRV, C2OPT, and DDS) are secure and can be written only by the secure CPU2 and read by any CPU, secure and non-secure. When the CPU1 tries to write, the write is discarded.

## 3.10 FLASH registers

### 3.10.1 Flash memory access control register (FLASH_ACR)

Address offset: 0x000

Reset value: 0x0000 0600

| 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. | EMPTY |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PES | Res. | Res. | DCRST | ICRST | DCEN | ICEN | PRFTEN | Res. | Res. | Res. | Res. | Res. | LATENCY[2:0] | | |
| rw | | | rw | rw | rw | rw | rw | | | | | | rw | rw | rw |

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

Bit 16 **EMPTY**: Flash memory user area empty.

When read indicates whether the first location of the user flash memory is erased or has a programmed value.

0: Read: User flash memory programmed

1: Read: User flash memory empty

When written this bit is overwritten with the written value.

Bit 15 **PES**: CPU1 program / erase suspend request

0: Flash memory program and erase operations granted.

1: Any new flash memory program and erase operation is suspended until this bit and the same bit in *Flash memory CPU2 access control register (FLASH_C2ACR)* are cleared. The PESD bit in both the *Flash memory status register (FLASH_SR)* and *Flash memory CPU2 status register (FLASH_C2SR)* is set when at least one PES bit in FLASH_ACR or FLASH_C2ACR is set.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **DCRST**: CPU1 Data cache reset

0: CPU1 Data cache is not reset

1: CPU1 Data cache is reset

This bit can be written only when the data cache is disabled.

Bit 11 **ICRST**: CPU1 Instruction cache reset

0: CPU1 Instruction cache is not reset

1: CPU1 Instruction cache is reset

This bit can be written only when the instruction cache is disabled.

Bit 10 **DCEN**: CPU1 Data cache enable

0: CPU1 Data cache is disabled

1: CPU1 Data cache is enabled

Bit 9 **ICEN**: CPU1 Instruction cache enable

0: CPU1 Instruction cache is disabled

1: CPU1 Instruction cache is enabled

Bit 8 **PRFTEN:** CPU1 Prefetch enable

0: CPU1 Prefetch disabled

1: CPU1 Prefetch enabled

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **LATENCY[2:0]**: Latency

These bits represent the ratio of the flash memory HCLK clock period to the flash memory access time.

000: Zero wait states
001: One wait state
010: Two wait states
011: Three wait states
Others: Reserved

### 3.10.2 Flash memory key register (FLASH_KEYR)

Address offset: 0x008

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| 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 |
| KEY[15:0] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **KEY[31:0]**: Flash memory key

The following values must be written consecutively to unlock the *Flash memory control register (FLASH_CR)* and *Flash memory CPU2 control register (FLASH_C2CR)*, thus enabling programming/erasing operations:

KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

### 3.10.3 Flash memory option key register (FLASH_OPTKEYR)

Address offset: 0x00C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OPTKEY[31:16] | | | | | | | | | | | | | | | |
| 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 |
| OPTKEY[15:0] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **OPTKEY[31:0]**: Option byte key

The following values must be written consecutively to unlock the flash memory option registers, enabling option byte programming/erasing operations:

KEY1: 0x0819 2A3B
KEY2: 0x4C5D 6E7F

### 3.10.4 Flash memory status register (FLASH_SR)

Address offset: 0x010

Reset value: 0x000X 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. | PESD | CFGBSY | Res. | BSY |
| | | | | | | | | | | | | r | r | | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OPTV ERR | RD ERR | OPTNV | Res. | Res. | Res. | FAST ERR | MISS ERR | PGS ERR | SIZ ERR | PGA ERR | WRP ERR | PROG ERR | Res. | OP ERR | EOP |
| rc_w1 | rc_w1 | r | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | | rc_w1 | rc_w1 |

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

Bit 19 **PESD**: Programming/erase operation suspended

This bit is set and reset by hardware.
Set when at least one PES bit in either *Flash memory access control register (FLASH_ACR)* or *Flash memory CPU2 access control register (FLASH_C2ACR)* is set.
Cleared when both PES bits in FLASH_ACR and FLASH_C2ACR are cleared.
When set, new program or erase operations are not started.

Bit 18 **CFGBSY**: Programming or erase configuration busy

This flag is set and reset by hardware:
- set when first word is sent for program, or when setting STRT for erase
- reset when operation completes or is interrupted by an error.

When set to 1, any other operation launch through *Flash memory control register (FLASH_CR)* is impossible and must be postponed (a programming or erase operation is ongoing). When reset to 0, programming and erase settings in *Flash memory control register (FLASH_CR)* can be modified.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **BSY**: Busy

Indicates that a flash memory operation requested by *Flash memory control register (FLASH_CR)* is in progress. This bit is set at the beginning of a flash memory operation, and reset when the operation finishes or when an error occurs.

Bit 15 **OPTVERR**: Option and Engineering bits loading validity error

Set by hardware when the options and engineering bits read may not be the one configured by the user or production. If options and engineering bits have not been properly loaded, OPTVERR is set again after each system reset. Option bytes that fail loading are forced to a safe value, see *Section 3.4.2: Option bytes programming*.
Cleared by writing 1.

Bit 14 **RDERR**: PCROP read error

Set by hardware when an address to be read through the D-bus belongs to a read protected area of the flash memory (PCROP protection). An interrupt is generated if RDERRIE is set in FLASH_CR.
Cleared by writing 1.

Bit 13 **OPTNV**: User option OPTVAL indication

This bit is set and reset by hardware.
0: The OBL user option OPTVAL indicates “valid”.
1: The OBL user option OPTVAL indicates “not valid”.

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **FASTERR**: Fast programming error

Set by hardware when a fast programming sequence (activated by FSTPG) is interrupted due to an error (alignment, size, write protection or data miss). The corresponding status bit (PGAERR, SIZERR, WRPERR or MISSERR) is set at the same time.
Cleared by writing 1.

Bit 8 **MISSERR**: Fast programming data miss error

In fast programming mode, 64 double words (512 bytes) must be sent to flash memory successively, and the new data must be sent to the logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.
Cleared by writing 1.

Bit 7 **PGSERR**: Programming sequence error

Set by hardware when a write access to the flash memory is performed by the code while PG or FSTPG have not been set previously. Set also by hardware when PROGERR, SIZERR, PGAERR, WRPERR, MISSERR or FASTERR is set due to a previous programming error.
Cleared by writing 1.

Bit 6 **SIZERR**: Size error

Set by hardware when the size of the access is a byte or half-word during a program or a fast program sequence. Only double word programming is allowed (consequently: word access).
Cleared by writing 1.

Bit 5 **PGAERR**: Programming alignment error

Set by hardware when the data to program cannot be contained in the same double word (64-bit) flash memory in case of standard programming, or if there is a change of page during fast programming.
Cleared by writing 1.

Bit 4 **WRPERR**: Write protection error

Set by hardware when an address to be erased/programmed belongs to a write-protected part (by WRP, PCROP or RDP Level 1) of the flash memory.
Cleared by writing 1.

Bit 3 **PROGERR**: Programming error

Set by hardware when a double-word address to be programmed contains a value different from 0xFFFF FFFF FFFF FFFF before programming, except if the data to write is 0x0000 0000 0000 0000.
Cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Operation error

Set by hardware when a flash memory operation (program / erase) completes unsuccessfully.
This bit is set only if error interrupts are enabled (ERRIE = 1).
Cleared by writing ‘1’.

Bit 0 **EOP**: End of operation

Set by hardware when one or more flash memory operation (programming / erase) has been completed successfully.
This bit is set only if the end of operation interrupts are enabled (EOPIE = 1).
Cleared by writing 1.

### 3.10.5 Flash memory control register (FLASH_CR)

Address offset: 0x014

Reset value: 0xC000 0000

Access: no wait state when no flash memory operation is on going, word, half-word and byte access.

This register must not be modified when CFGBSY in *Flash memory status register (FLASH_SR)* is set. This would result in:

- a bus error, if CFGBSY is set due to a double-word programming sequence and the write to FLASH_CR is performed after the first word is sent for programming and before the second.
- a bus stall, if CFGBSY is set for any other reason
  - When the PESD bit in *Flash memory status register (FLASH_SR)* is cleared, the register write access is stalled until the CFGBSY bit is cleared. (by the other CPU)
  - When the PESD bit in *Flash memory status register (FLASH_SR)* is set and a program or an erase operation is ongoing, the register write access causes a bus error.
  - When the PESD bit in *Flash memory status register (FLASH_SR)* is set, but there is no ongoing programming or erase operation, the register write access is completed. The requested program or erase operation is suspended, the BSY/CFGBSY asserted and remains 1, until suspend is deactivated. Consequently, PESD bit goes back to 0 and the suspended operation completes.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCK | OPT LOCK | Res. | Res. | OBL_ LAUNCH | RD ERRIE | ERRIE | EOPIE | Res. | Res. | Res. | Res. | Res. | FSTPG | OPT STRT | STRT |
| rs | rs | | | rc_w1 | rw | rw | rw | | | | | | rw | rs | rs |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | PNB[7:0] | | | | | | | | MER | PER | PG |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **LOCK:** FLASH_CR lock

This bit is set only. When set, the FLASH_CR register is locked. It is cleared by hardware after detecting the unlock sequence.

In case of an unsuccessful unlock operation, this bit remains set until the next system reset.

Bit 30 **OPTLOCK:** Options lock

This bit is set only. When set, all bits concerning in FLASH_CR register and so option page are locked. This bit is cleared by hardware after detecting the unlock sequence. The LOCK bit must be cleared before doing the unlock sequence for OPTLOCK bit.

In case of an unsuccessful unlock operation, this bit remains set until the next reset.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **OBL_LAUNCH**: Forces the option byte loading

When set to 1, this bit forces the option byte reloading. This bit is cleared only when the option byte loading is complete. It cannot be written if OPTLOCK is set.

0: Option byte loading complete

1: Option byte loading requested

Bit 26 **RDERRIE**: PCROP read error interrupt enable

This bit enables the interrupt generation when the RDERR bit in the FLASH_SR is set to 1.

0: PCROP read error interrupt disabled

1: PCROP read error interrupt enabled

Bit 25 **ERRIE**: Error interrupt enable
This bit enables the interrupt generation when the OPERR bit in the FLASH_SR is set to 1.
0: OPERR error interrupt disabled
1: OPERR error interrupt enabled

Bit 24 **EOPIE**: End of operation interrupt enable
This bit enables the interrupt generation when the EOP bit in the FLASH_SR is set to 1.
0: EOP Interrupt disabled
1: EOP Interrupt enabled

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **FSTPG**: Fast programming
0: Fast programming disabled
1: Fast programming enabled

Bit 17 **OPTSTRT**: Options modification start
This bit triggers an options operation when set.
This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH_SR.

Bit 16 **STRT**: Start
This bit triggers an erase operation when set. If MER and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition must be forbidden.
This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH_SR.
Starting operations by the CPU1, involving secure flash memory pages are rejected and a bus error is generated.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:3 **PNB[7:0]**: Page number selection
These bits select the page to erase:
0x00: page 0
0x01: page 1
...
0x7F: page 127
... (STM32WB50CG only)
0xFF: page 255 (STM32WB50CG only)

Bit 2 **MER**: Mass erase
This bit triggers the mass erase (all user pages) when set.

Bit 1 **PER**: Page erase
0: page erase disabled
1: page erase enabled

Bit 0 **PG**: Programming
0: Flash memory programming disabled
1: Flash memory programming enabled

### 3.10.6 Flash memory ECC register (FLASH_ECCR)

Address offset: 0x018

Reset value: 0x0000 0000

Access: no wait state when no flash memory operation is on going, word, half-word and byte access.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCD | ECCC | Res. | CPUID[2:0] | | | Res. | ECCCIE | Res. | Res. | Res. | SYSF _ECC | Res. | Res. | Res. | ADDR _ECC[16] |
| rc_w1 | rc_w1 | | r | r | r | | rw | | | | r | | | | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADDR_ECC[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31 **ECCD**: ECC detection

Set by hardware when two ECC errors have been detected. When this bit is set, an NMI is generated.
Cleared by writing 1.

Bit 30 **ECCC**: ECC correction

Set by hardware when one ECC error has been detected and corrected. An interrupt is generated if ECCIE is set.
Cleared by writing 1.

Bit 29 Reserved, must be kept at reset value.

Bits 28:26 **CPUID[2:0]**: CPU identification

Set by hardware, indicates the Bus-ID of the CPU access causing the ECC failure.
0x0: value for CPUID for CPU1 bus-ID
0x1: value for CPUID for CPU2 bus-ID

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ECCCIE**: ECC correction interrupt enable

0: ECCC interrupt disabled
1: ECCC interrupt enabled

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **SYSF_ECC**: System flash memory ECC fail

This bit indicates that the ECC error correction or double ECC error detection is located in the system flash memory.

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

Bits 16:0 **ADDR_ECC[16:0]**: ECC fail double-word address

These bits indicate that double word address is concerned by the ECC error correction or causes the double ECC error detection.

### 3.10.7 Flash memory option register (FLASH_OPTR)

Address offset: 0x020

Reset value: 0bxxx1 xxxx x111 xxxx 1xxx xxx1 xxxx xxxx (the option bits are loaded with values from flash memory at reset release)

Access: no wait state when no flash memory operation is ongoing, word, half-word and byte access.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AGC_TRIM[2:0] | | | Res. | n BOOT0 | nSW BOOT0 | SRAM2 _RST | SRAM2 _PE | n BOOT1 | Res. | Res. | Res. | WWDG _SW | IWDG _STDBY | IWDG _STOP | IWDG _SW |
| 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. | nRST _SHDW | nRST _STDBY | nRST _STOP | BOR_LEV[2:0] | | | ESE | RDP[7:0] | | | | | | | |
| | rw | rw | rw | rw | rw | rw | r | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:29 **AGC_TRIM[2:0]**: Radio automatic gain control trimming
Default value 0b001

Bit 28 Reserved, must be kept at reset value.

Bit 27 **nBOOT0:** nBOOT0 option bit
If nSWBOOT0 bit configuration select BOOT0 is taken from bit nBOOT0, together with bit nBOOT1, selects boot from the user flash memory, SRAM1, or system flash memory. Refer to *Section 2.3: Boot configuration*.
0: nBOOT0 = 0
1: nBOOT0 = 1

Bit 26 **nSWBOOT0:** Software BOOT0 selection
0: BOOT0 taken from the option bit nBOOT0
1: BOOT0 taken from PH3/BOOT0 pin

Bit 25 **SRAM2_RST:** SRAM2 and PKA RAM Erase when system reset
0: SRAM2 and PKA RAM erased when a system reset occurs
1: SRAM2 and non-secure PKA RAM not erased when a system reset occurs

Bit 24 **SRAM2_PE:** SRAM2 parity check enable
0: SRAM2 parity check enabled
1: SRAM2 parity check disabled

Bit 23 **nBOOT1:** Boot configuration
Together with the BOOT0 pin or option bit nBOOT0 (depending on nSWBOOT0 option bit configuration), this bit selects boot mode from the user flash memory, SRAM1 or the System memory. Refer to *Section 2.3: Boot configuration*.

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 **WWDG_SW:** Window watchdog selection
0: Hardware window watchdog
1: Software window watchdog

Bit 18 **IWDG_STDBY:** Independent watchdog counter freeze in Standby mode
0: Independent watchdog counter is frozen in Standby mode
1: Independent watchdog counter is running in Standby mode

Bit 17 **IWDG_STOP:** Independent watchdog counter freeze in Stop mode
0: Independent watchdog counter is frozen in Stop mode
1: Independent watchdog counter is running in Stop mode

Bit 16 **IWDG_SW:** Independent watchdog selection
0: Hardware independent watchdog
1: Software independent watchdog

Bit 15 Reserved, must be kept at reset value.

Bit 14 **nRST_SHDW:**

0: Reset generated when entering the Shutdown mode
1: No reset generated when entering the Shutdown mode

Bit 13 **nRST_STDBY:**

0: Reset generated when entering the Standby mode
1: No reset generated when entering the Standby mode

Bit 12 **nRST_STOP:**

0: Reset generated when entering the Stop mode
1: No reset generated when entering the Stop mode

Bits 11:9 **BOR_LEV[2:0]:** BOR reset level

These bits contain the $V_{DD}$ supply level threshold that activates/releases the reset.

000: BOR Level 0. Reset level threshold is ~ 1.7 V
001: BOR Level 1. Reset level threshold is ~ 2.0 V
010: BOR Level 2. Reset level threshold is ~ 2.2 V
011: BOR Level 3. Reset level threshold is ~ 2.5 V
100: BOR Level 4. Reset level threshold is ~ 2.8 V

Bit 8 **ESE:** System security enabled flag.

Indicates whether the system security is enabled user flash memory FSD = 0.

0: Security disabled
1: Security enabled

Bits 7:0 **RDP[7:0]:** Read protection level

0xAA: Level 0, read protection not active
0xCC: Level 2, chip read protection active
Others: Level 1, memories read protection active

*Note: Take care about PCROP_RDP configuration in Level 1. Refer to Level 1: Read protection for more details.*

### 3.10.8 Flash memory PCROP zone A start address register (FLASH_PCROP1ASR)

Address offset: 0x024

Reset value: ‘b1111 1111 1111 1111 1111 111X XXXX XXXX

Access: no wait state when no flash memory operation is on going, word, half-word access

| 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. | PCROP1A_STRT[8:0] | | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 8:0 **PCROP1A_STRT[8:0]:** PCROP1A area start offset

PCROP1A_STRT contains the first 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB30CE.

### 3.10.9 Flash memory PCROP zone A end address register (FLASH_PCROP1AER)

Address offset: 0x028

Reset value: ‘bX111 1111 1111 1111 1111 1111 XXXX XXXX

Access: no wait state when no flash memory operation is on going, word, half-word access. PCROP_RDP bit can be accessed with byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PCROP_RDP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| rs | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | PCROP1A_END[8:0] | | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **PCROP_RDP:** PCROP area preserved when RDP level decreased

This bit is set only. It is reset after a full mass erase due to a change of RDP from Level 1 to Level 0.

0: PCROP area is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:9 Reserved, must be kept at reset value.

Bits 8:0 **PCROP1A_END[8:0]:** PCROP1A area end offset

PCROP1A_END contains the last 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB30CE devices.

### 3.10.10 Flash memory WRP area A address register (FLASH_WRP1AR)

Address offset: 0x02C

Reset value: 0xFFXX FFXX

Access: no wait state when no flash memory operation is on going, word, half-word and byte access.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1A_END[7:0] | | | | | | | |
| | | | | | | | | 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. | Res. | Res. | Res. | Res. | WRP1A_STRT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 23:16 **WRP1A_END[7:0]:** WRP first area “A” end offset

Contains the last 4 Kbytes page of the WRP first area. Note that bit 23 is reserved on STM32WB30CE devices.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **WRP1A_STRT[7:0]:** WRP first area “A” start offset
Contains the first 4 Kbytes page of the WRP first area. Note that bit 7 is reserved on STM32WB30CE devices.

### 3.10.11 Flash memory WRP area B address register (FLASH_WRP1BR)

Address offset: 0x030

Reset value: 0xFFXX FFXX

Access: no wait state when no flash memory operation is on going, word, half-word and byte access.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1B_END[7:0] | | | | | | | |
| | | | | | | | | 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. | Res. | Res. | Res. | Res. | WRP1B_STRT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 23:16 **WRP1B_END[7:0]:** WRP second area “B” end offset
WRPB1_END contains the last 4 Kbytes page of the WRP second area. Note that bit 23 is reserved on STM32WB30CE devices.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **WRP1B_STRT[7:0]:** WRP second area “B” start offset
WRPB1_END contains the first 4 Kbytes page of the WRP second area. Note that bit 7 is reserved on STM32WB30CE devices.

### 3.10.12 Flash memory PCROP zone B start address register (FLASH_PCROP1BSR)

Address offset: 0x034

Reset value: ‘b1111 1111 1111 1111 1111 111X XXXX XXXX

Access: no wait state when no flash memory operation is on going, word, half-word access.

| 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. | PCROP1B_STRT[8:0] | | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 8:0 **PCROP1B_STRT[8:0]:** PCROP1B area start offset
Contains the first 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB30CE devices.

### 3.10.13 Flash memory PCROP zone B end address register (FLASH_PCROP1BER)

Address offset: 0x038

Reset value: ‘b1111 1111 1111 1111 1111 111X XXXX XXXX

Access: no wait state when no flash memory operation is on going, word, half-word access.

| 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. | PCROP1B_END[8:0] | | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 8:0 **PCROP1B_END[8:0]:** PCROP1B area end offset

Contains the last 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB30CE devices.

### 3.10.14 Flash memory IPCC mailbox data buffer address register (FLASH_IPCCBR)

Address offset: 0x03C

Reset value: ‘b1111 1111 1111 1111 11XX XXXX XXXX XXXX

| 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. | IPCCDBA[13:0] | | | | | | | | | | | | | |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 13:0 **IPCCDBA[13:0]:** IPCC mailbox data buffer base address offset

Contains the first double-word offset of the IPCC mailbox data buffer area in SRAM2.

### 3.10.15 Flash memory CPU2 access control register (FLASH_C2ACR)

Address offset: 0x05C

Reset value: 0x0000 0600

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PES | Res. | Res. | Res. | ICRST | Res. | ICEN | PRFTEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| rw | | | | rw | | rw | rw | | | | | | | | |

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

Bit 15 **PES**: CPU2 program / erase suspend request

0: Flash memory program and erase operations granted.

1: Any new flash memory program and erase operation is suspended until this bit and the same bit in *Flash memory access control register (FLASH_ACR)* are cleared. The PESD bit in both the *Flash memory status register (FLASH_SR)* and *Flash memory CPU2 status register (FLASH_C2SR)* is set when at least one PES bit in FLASH_ACR or FLASH_C2ACR is set.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ICRST**: CPU2 Instruction cache reset

0: CPU2 Instruction cache is not reset

1: CPU2 Instruction cache is reset

This bit can be written only when the instruction cache is disabled.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **ICEN**: CPU2 Instruction cache enable

0: CPU2 Instruction cache is disabled

1: CPU2 Instruction cache is enabled

Bit 8 **PRFTEN:** CPU2 Prefetch enable.

0: CPU2 prefetch is disabled

1: CPU2 prefetch is enabled

Bits 7:0 Reserved, must be kept at reset value.

### 3.10.16 Flash memory CPU2 status register (FLASH_C2SR)

Address offset: 0x060

Reset value: 0x000X 0000

Access: no wait state, word, half-word and byte access.

| 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. | PESD | CFGBSY | Res. | BSY |
| | | | | | | | | | | | | r | r | | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | RD ERR | Res. | Res. | Res. | Res. | FAST ERR | MISS ERR | PGS ERR | SIZ ERR | PGA ERR | WRP ERR | PROG ERR | Res. | OP ERR | EOP |
| | rc_w1 | | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | | rc_w1 | rc_w1 |

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

Bit 19 **PESD**: Programming / erase operation suspended.

This bit is set and reset by hardware.

Set when at least one PES bit in either *Flash memory access control register (FLASH_ACR)* or *Flash memory CPU2 access control register (FLASH_C2ACR)* is set.

Cleared when both PES bits in FLASH_ACR and FLASH_C2ACR are cleared.

When set new program or erase operations are not started.

Bit 18 **CFGBSY**: Programming or erase configuration busy

This flag is set and reset by hardware:

- set when the first word is sent for program, or when setting STRT for erase
- reset when operation completes or is interrupted by an error.

When set to 1, any other operation through *Flash memory CPU2 control register (FLASH_C2CR)* is impossible and must be postponed (a programming or erase operation is ongoing). When reset to 0, programming and erase settings in *Flash memory CPU2 control register (FLASH_C2CR)* can be modified.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **BSY**: Busy

This bit indicates that a flash memory operation requested by *Flash memory CPU2 control register (FLASH_C2CR)* is in progress. This is set on the beginning of a flash memory operation and reset when the operation finishes or when an error occurs.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RDERR**: PCROP read error

Set by hardware when an address to be read through the D-bus belongs to a read protected area of the flash memory (PCROP protection). An interrupt is generated if RDERRIE is set in FLASH_CR.

Cleared by writing 1.

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **FASTERR**: Fast programming error

Set by hardware when a fast programming sequence (activated by FSTPG) is interrupted due to an error (alignment, size, write protection or data miss). The corresponding status bit (PGAERR, SIZERR, WRPERR or MISSERR) is set at the same time.

Cleared by writing 1.

Bit 8 **MISSERR**: Fast programming data miss error

In fast programming mode, 64 double words (512 bytes) must be sent to the flash memory successively, and the new data must be sent to the flash memory logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.

Cleared by writing 1.

Bit 7 **PGSERR**: Programming sequence error

Set by hardware when a write access to the flash memory is performed by the code while PG or FSTPG have not been set previously. Set also by hardware when PROGERR, SIZERR, PGAERR, WRPERR, MISSERR or FASTERR is set due to a previous programming error.

Cleared by writing 1.

Bit 6 **SIZERR**: Size error

Set by hardware when the size of the access is a byte or half-word during a program or a fast program sequence. Only double word programming is allowed (consequently: word access).

Cleared by writing 1.

Bit 5 **PGAERR**: Programming alignment error

Set by hardware when the data to program cannot be contained in the same double word (64-bit) flash memory in case of standard programming, or if there is a change of page during fast programming.

Cleared by writing 1.

Bit 4 **WRPERR**: Write protection error

Set by hardware when an address to be erased/programmed belongs to a write-protected part (by WRP, PCROP or RDP Level 1) of the flash memory.
Cleared by writing 1.

Bit 3 **PROGERR**: Programming error

Set by hardware when a double-word address to be programmed contains a value different from 0xFFFF FFFF before programming, except if the data to write is 0x0000 0000 0000 0000.
Cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Operation error

Set by hardware when a flash memory operation (program / erase) completes unsuccessfully.
This bit is set only if error interrupts are enabled (ERRIE = 1).
Cleared by writing ‘1’.

Bit 0 **EOP**: End of operation

Set by hardware when one or more flash memory operation (programming / erase) completes successfully.
This bit is set only if the end of operation interrupts are enabled (EOPIE = 1).
Cleared by writing 1.

### 3.10.17 Flash memory CPU2 control register (FLASH_C2CR)

Address offset: 0x064

Reset value: 0xC000 0000

Access: no wait state when no flash memory operation is on going, word, half-word and byte access

This register cannot be modified when CFGBSY in *Flash memory CPU2 status register (FLASH_C2SR)* is set.

- When the PESD bit in *Flash memory CPU2 status register (FLASH_C2SR)* is cleared, the register write access is stalled until the CFGBSY bit is cleared (by the other CPU).
- When the PESD bit in *Flash memory CPU2 status register (FLASH_C2SR)* is set, the register write access causes a bus error.
- When the PESD bit in *Flash memory CPU2 status register (FLASH_C2SR)* is set, but there is no ongoing programming or erase operation, the register write access is completed. The requested program or erase operation is suspended, the BSY/CFGBSY asserted and remains 1 until suspend has been deactivated. Consequently, PESD bit goes back to 0 and the suspended operation completes.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | RD ERRIE | ERR IE | EOP IE | Res. | Res. | Res. | Res. | Res. | FSTPG | Res. | STRT |
| | | | | | rw | rw | rw | | | | | | rw | | rs |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | PNB[7:0] | | | | | | | | MER | PER | PG |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RDERRIE**: PCROP read error interrupt enable
This bit enables the interrupt generation when the RDERR bit in the FLASH_SR is set to 1.
0: PCROP read error interrupt disabled
1: PCROP read error interrupt enabled

Bit 25 **ERRIE**: Error interrupt enable
This bit enables the interrupt generation when the OPERR bit in the FLASH_SR is set to 1.
0: OPERR error interrupt disabled
1: OPERR error interrupt enabled

Bit 24 **EOPIE**: End of operation interrupt enable
This bit enables the interrupt generation when the EOP bit in the FLASH_SR is set to 1.
0: EOP Interrupt disabled
1: EOP Interrupt enabled

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **FSTPG**: Fast programming
0: Fast programming disabled
1: Fast programming enabled

Bit 17 Reserved, must be kept at reset value.

Bit 16 **STRT**: Start
This bit triggers an erase operation when set. If MER and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition must be forbidden.
This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH_SR.
Starting operations by the CPU1, involving secure flash memory pages is rejected and a bus error generated.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:3 **PNB[7:0]**: Page number selection
These bits select the page to erase:
0x00: page 0
0x01: page 1
...

... (STM32WB50CG only)
0xFF: page 255 (STM32WB50CG only)

Bit 2 **MER**: Mass erase
This bit triggers the mass erase (all user pages) when set.

Bit 1 **PER**: Page erase
0: Page erase disabled
1: Page erase enabled

Bit 0 **PG**: Programming
0: Flash programming disabled
1: Flash programming enabled

### 3.10.18 Secure flash memory start address register (FLASH_SFR)

Address offset: 0x080

Reset value: 0b1111 1111 1111 1111 1111 1110 xxxx xxxx

This register provides write access security and can only be written by the CPU2. A write access from the CPU1 is ignored and a bus error generated. On any read access the register value is returned.

Written values are only taken into account after OBL.

| 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. | DDS | Res. | Res. | Res. | FSD | SFSA[7:0] | | | | | | | |
| | | | rw | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bit 12 **DDS:** Disable CPU2 debug access
0: CPU2 debug access enabled
1: CPU2 debug access disabled

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

Bit 8 **FSD:** Flash memory security disabled.
1: System and flash memory non-secure
0: System and flash memory secure (the secure area of the flash memory is given by SFSA)

Bits 7:0 **SFSA[7:0]:** Secure flash memory start address
SFSA[7:0] contain the start address of the first 4 Kbytes page of the secure flash memory area.

### 3.10.19 Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)

Address offset: 0x084

Reset value: 0bxxxx xxx1 xxxx xxxx xxxx xxxx xxxx xxxx

This register provides write access security and can only be written by the CPU2. A write access from the CPU1 is ignored and a bus error generated. On any read access the register value is returned.

Written values are only taken into account after OBL.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C2OPT | NBRSD | SNBRSA[4:0] | | | | | Res. | BRSD | SBRSA[4:0] | | | | | SBRV[17:16] | |
| 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 |
| SBRV[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **C2OPT:** CPU2 boot reset vector memory selection.

0: SBRV offset addresses SRAM1 or SRAM2, from start address 0x2000 0000. SBRV value must be kept within the SRAM area.

1: SBRV offset addresses flash memory, from start address 0x0800 0000.

Bit 30 **NBRSD:** Non-backup SRAM2b security disable.

NBRSD = 1: SRAM2b is non-secure

NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.

Bits 29:25 **SNBRSA[4:0]:** Secure non-backup SRAM2b start address

NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **BRSD:** Backup SRAM2a security disable.

BRSD = 1: SRAM2a is non-secure

BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.

Bits 22:18 **SBRSA[4:0]:** Secure backup SRAM2a start address

BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.

Bits 17:0 **SBRV[17:0]:** CPU2 boot reset vector

Contains the word aligned CPU2 boot reset start address offset within the selected memory area by C2OPT.

### 3.10.20 FLASH register map

**Table 18. Flash interface 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 | FLASH_ACR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EMPTY | PES | Res. | Res. | DCRST | ICRST | DCEN | ICEN | PRFTEN | Res. | Res. | Res. | Res. | Res. | LATENCY [2:0] | | |
| | Reset value | | | | | | | | | | | | | | | | 0 | 0 | | | 0 | 0 | 1 | 1 | 0 | | | | | | 0 | 0 | 0 |
| 0x004 | 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. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x008 | FLASH_KEYR | KEYR[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 | FLASH_OPT KEYR | OPTKEYR[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 | FLASH_SR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PESD | CFGBSY | Res. | BSY | OPTVERR | RDERR | OPTNV | Res. | Res. | Res. | FASTERR | MISERR | PGSERR | SIZERR | PGAERR | WRPERR | PROGERR | Res. | OPERR | EOP |
| | Reset value | | | | | | | | | | | | | 0 | 0 | | 0 | X | 0 | X | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 |
| 0x014 | FLASH_CR | LOCK | OPTLOCK | Res. | Res. | OBL_LAUNCH | RDERRIE | ERRIE | EOPIE | Res. | Res. | Res. | Res. | Res. | FSTPG | OPTSTRT | STRT | Res. | Res. | Res. | Res. | Res. | PNB[7:0] | | | | | | | | MER | PER | PG |
| | Reset value | 1 | 1 | | | 0 | 0 | 0 | 0 | | | | | | 0 | 0 | 0 | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x018 | FLASH_ECCR | ECCD | ECCC | Res. | CPUID [2:0] | | | Res. | ECCIE | Res. | Res. | Res. | SYSF_ECC | Res. | Res. | Res. | ADDR_ECC[16:0] | | | | | | | | | | | | | | | | |
| | 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 |
| 0x020 | FLASH_OPTR | AGC_TRIM [2:0] | | | Res. | nBOOT0 | nSWBOOT0 | SRAM2_RST | SRAM2_PE | nBOOT1 | Res. | Res. | Res. | WWDG_SW | IWDG_STBY | IWDG_STOP | IWDG_SW | Res. | nRST_SHDW | nRST_STDB | nRST_STOP | BOR_LEV [2:0] | | | ESE | RDP[7:0] | | | | | | | |
| | Reset value | X | X | X | | X | X | X | X | X | | | | X | X | X | X | | X | X | X | X | X | X | 1 | X | X | X | X | X | X | X | X |
| 0x024 | FLASH_ PCROP1ASR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PCROP1A_STRT[8:0] | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | X | X | X | X | X | X | X | X | X |
| 0x028 | FLASH_ PCROP1AER | PCROP_RDP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PCROP1A_END[8:0] | | | | | | | | |
| | Reset value | x | | | | | | | | | | | | | | | | | | | | | | | X | X | X | X | X | X | X | X | X |
| 0x02C | FLASH_ WRP1AR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1A_END[7:0] | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1A_STRT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | X | X | X | X | X | X | X | X | | | | | | | | | X | X | X | X | X | X | X | X |

**Table 18. Flash interface 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x030 | FLASH_ WRP1BR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1B_END[7:0] | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WRP1B_STRT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | X | X | X | X | X | X | X | X | | | | | | | | | X | X | X | X | X | X | X | X |
| 0x034 | FLASH_ PCROP1BSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PCROP1B_STRT[8:0] | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | X | X | X | X | X | X | X | X | X |
| 0x038 | FLASH_ PCROP1BER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PCROP1B_END[8:0] | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | X | X | X | X | X | X | X | X | X |
| 0x03C | FLASH_ IPCCBR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | IPCCDBA[13:0] | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 0x05C | FLASH_C2ACR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PES | Res. | Res. | Res. | ICRST | Res. | ICEN | PRFTEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | 0 | | | | 0 | | 1 | 0 | | | | | | | | |
| 0x060 | FLASH_C2SR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PESD | CFGBSY | Res. | BSY | Res. | RDERR | Res. | Res. | Res. | Res. | FASTERR | MISERR | PGSERR | SIZERR | PGAERR | WRPERR | PROGERR | Res. | OPERR | EOP |
| | Reset value | | | | | | | | | | | | | 0 | 0 | | 0 | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 |
| 0x064 | FLASH_C2CR | Res. | Res. | Res. | Res. | Res. | RDERRIE | ERRIE | EOPIE | Res. | Res. | Res. | Res. | Res. | FSTPG | Res. | STRT | Res. | Res. | Res. | Res. | Res. | PNB[7:0] | | | | | | | | MER | PER | PG |
| | Reset value | | | | | | 0 | 0 | 0 | | | | | | 0 | | 0 | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x080 | FLASH_SFR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DDS | Res. | Res. | Res. | FSD | SFSA[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | 1 | | | | 0 | X | X | X | X | X | X | X | X |
| 0x084 | FLASH_SRRVR | C2OPT | NBRSD | SNBRSA[4:0] | | | | | Res. | BRSD | SBRSA[4:0] | | | | | SBRV[17:0] | | | | | | | | | | | | | | | | | |
| | Reset value | X | X | X | X | X | X | X | | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 4 Radio system

## 4.1 Introduction

The system is ultra low power compliant with the Bluetooth® core specification BLE5.4 and IEEE 802.15.4 standard.

The radio system consists of a 2.4 GHz RF front end and a Bluetooth® Low Energy (BLE) and IEEE 802.15.4 physical layer controller. The system is controlled from the CPU2 that contains the radio lower protocol software layers. Interface to the application running on the CPU2 is provided via mailbox message system.

## 4.2 Main features

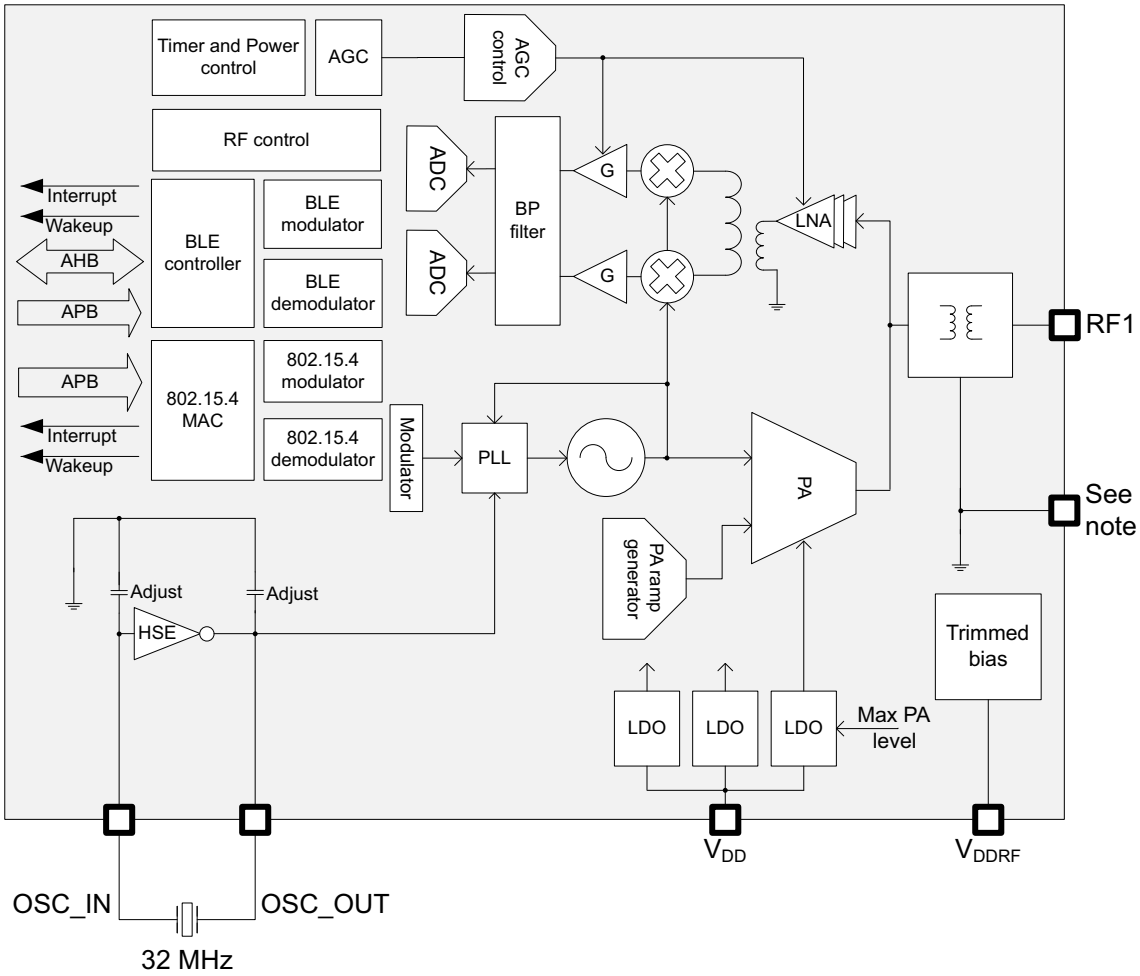
- 2.4 GHz RF transceiver supporting:
  - Bluetooth® BLE5.4 standard support
  - IEEE 802.15.4-2011 standard support
- Programmable output power
- RSSI
- Integrated balun
- Bluetooth® BLE5.4 features:
  - GAP: central, peripheral, observer and broadcaster roles
  - Simultaneous multiple role support
  - Master / slave support
  - ATT / GATT: client and server

## 4.3 Radio system functional description

### 4.3.1 General description

The block diagram of the Radio system is shown in *Figure 5*.

**Figure 5. Radio system block diagram**




Note: on QFN packages $V_{SS}$ through exposed pad, and $V_{SSRF}$ pin must be connected to ground plane

# 5 Cyclic redundancy check calculation unit (CRC)

## 5.1 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.

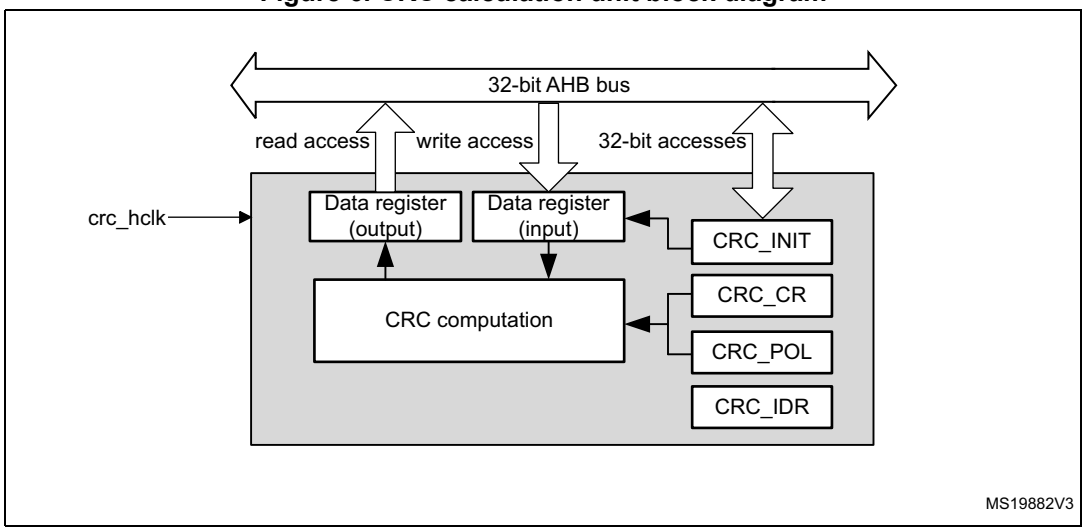
## 5.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$
- Alternatively, uses 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
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

## 5.3 CRC functional description

### 5.3.1 CRC block diagram

**Figure 6. CRC calculation unit block diagram**




### 5.3.2 CRC internal signals

**Table 19. CRC internal input/output signals**

| Signal name | Signal type | Description |
|---|---|---|
| crc_hclk | Digital input | AHB clock |

### 5.3.3 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 accesses are allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written 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, 0x1A2B3C4D input data are 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, 0x11223344 output data are converted to 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.

*Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.*

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.

## 5.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

### 5.4.1 CRC 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.

### 5.4.2 CRC 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDR[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 |
| IDR[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

### 5.4.3 CRC 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 at reset value.

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

This bitfield controls 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 at reset value.

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

### 5.4.4 CRC initial 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[31:0]***:* Programmable initial CRC value

This register is used to write the CRC initial value.

### 5.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POL[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 |
| POL[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | 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.

### 5.4.6 CRC register map

**Table 20. CRC register map and reset values**

| Offset | Register name | 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 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 6 Power control (PWR)

## 6.1 Power supplies

The devices require a $V_{DD}$ operating voltage supply between 2.00 V and 3.6 V. Several independent supplies ($V_{DDA}$, $V_{DDRF}$) can be provided for specific peripherals:

- $V_{DD}$ = 2.00 V to 3.6 V

  $V_{DD}$ is the external power supply for the I/Os, the system analog blocks such as reset, power management, internal clocks and low power regulator. It is provided externally through VDD pins.
- $V_{DDA}$ = 2.00 V to 3.6 V

  $V_{DDA}$ is the external analog power supply for A/D converters. The $V_{DDA}$ voltage level is independent from the $V_{DD}$ voltage and must preferably be connected to $V_{DD}$ when these peripherals are not used.
- $V_{DDRF}$ = 2.00 V to 3.6 V

  $V_{DDRF}$ is the external power supply for the Radio. It is provided externally through the VDDRF pin, and must be connected to the same supply as $V_{DD}$.
- $V_{BAT}$ = 1.55 V to 3.6 V

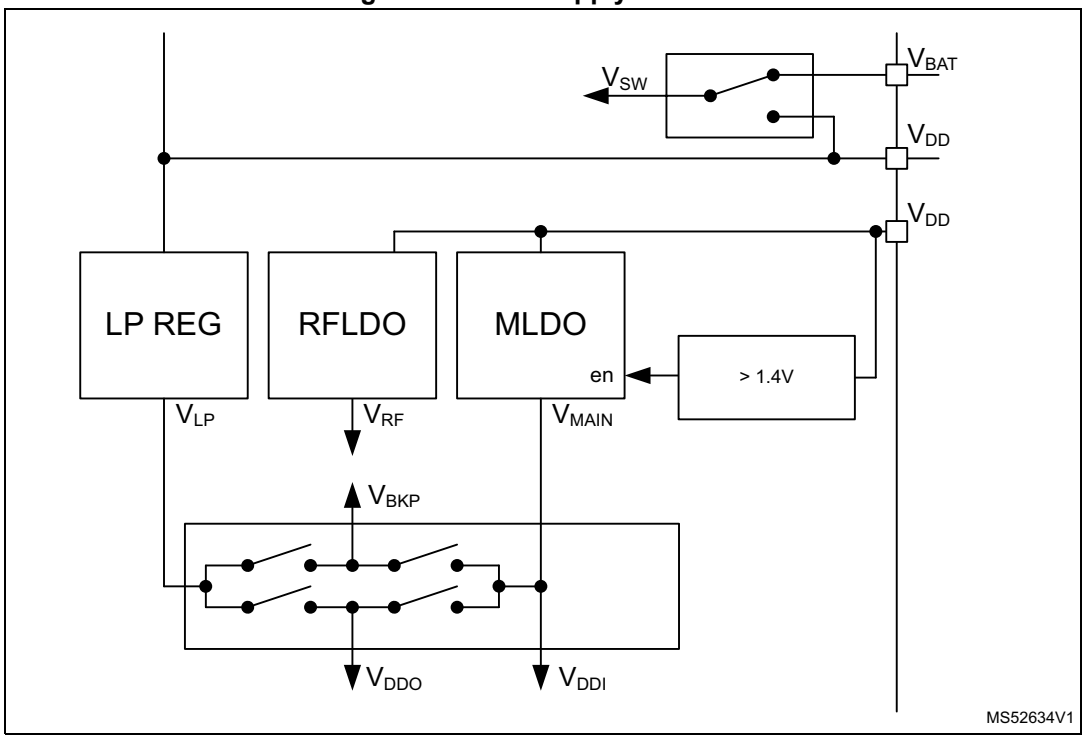
  $V_{BAT}$ is the power supply for RTC, external clock 32 kHz oscillator and backup registers (through power switch) when $V_{DD}$ is not present.

During power up and power down, the following power sequence is required:

- When $V_{DD}$ < 1 V the other power supplies ($V_{DDA}$) must remain below $V_{DD}$ + 0.3 V. During the power down $V_{DD}$ can temporarily become lower than the other supplies only if the energy provided to the MCU remains below 1 mJ. This allows the external decoupling capacitors to discharge with different time constants.
- When $V_{DD}$ ≥ 1 V all power supplies become independent.

An embedded linear voltage regulator is used to supply the internal digital power $V_{CORE}$. $V_{CORE}$ is the power supply for digital peripherals, SRAM1 and SRAM2. The flash memory is supplied by $V_{CORE}$ and $V_{DD}$.

**Figure 7. Power supply overview**




### 6.1.1 Independent analog peripherals supply

To improve ADC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply that can be separately filtered and shielded from noise on the PCB.

- The analog peripherals voltage supply input is available on a separate $V_{DDA}$ pin.
- An isolated supply ground connection is provided on $V_{SSA}$ pin.

The $V_{DDA}$ supply voltage can be different from $V_{DD}$. The presence of $V_{DDA}$ must be checked before enabling any of the analog peripherals supplied by $V_{DDA}$ (A/D converter, comparators, voltage reference buffer).

When a single supply is used, $V_{DDA}$ can be externally connected to $V_{DD}$ through the external filtering circuit in order to ensure a noise-free $V_{DDA}$ reference voltage.

#### ADC reference voltage

$V_{DDA}$ is also the reference voltage for the ADC.

### 6.1.2 Battery backup domain

To retain the content of the Backup registers and supply the RTC function when $V_{DD}$ is turned off, the $V_{BAT}$ pin can be connected to an optional backup voltage supplied by a battery or by another source.

The $V_{BAT}$ pin powers the RTC unit, the LSE oscillator and the PC14 to PC15 I/Os, allowing the RTC to operate even when the main power supply is turned off. The switch to the $V_{BAT}$ supply is controlled by the power-down reset embedded in the Reset block.

**Warning:** **During $t_{RSTTEMPO}$ (temporization at $V_{DD}$ startup) or after a PDR has been detected, the power switch between $V_{BAT}$ and $V_{DD}$ remains connected to $V_{BAT}$.**
**During the startup phase, if $V_{DD}$ is established in less than $t_{RSTTEMPO}$ (refer to the datasheet for its value) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into $V_{BAT}$ through an internal diode connected between $V_{DD}$ and the power switch ($V_{BAT}$).**
**If the power supply / battery connected to the VBAT pin cannot support this current injection, it is recommended to connect an external low-drop diode between this power supply and the VBAT pin.**

If no external battery is used in the application, it is recommended to connect $V_{BAT}$ to $V_{DD}$ supply, and add a 100 nF external ceramic decoupling capacitor on VBAT pin.

When the backup domain is supplied by $V_{DD}$ (analog switch connected to $V_{DD}$), the following pins are available:

- PC14 and PC15, which can be used as GPIO pins
- PC14 and PC15, which can be configured by RTC or LSE (refer to *Section 24.3: RTC functional description*)
- PA0/RTC_TAMP2 when configured by the RTC as tamper pin

*Note: As the analog switch can transfer only a limited amount of current (3 mA), the use of GPIO PC14 to PC15 in output mode is restricted: the speed must be limited to 2 MHz with a maximum load of 30 pF, and these I/Os cannot be used as current sources (e.g. to drive a LED).*

When the backup domain is supplied by $V_{BAT}$ (analog switch connected to $V_{BAT}$ because $V_{DD}$ is not present), the following functions are available:

- PC14 and PC15 can be controlled only by LSE (refer to *Section 24.3: RTC functional description*)
- PA0/RTC_TAMP2 when configured by the RTC as tamper pin

### Backup domain access

After a system reset, the backup domain (RTC registers and backup registers) is protected against possible unwanted write accesses. To enable access to the backup domain, proceed as follows:

1. Set the DBP bit in the *PWR control register 1 (PWR_CR1)* to enable access to the backup domain.

### VBAT battery charging

When $V_{DD}$ is present, it is possible to charge the external battery on VBAT through an internal resistance.

The VBAT charging is done either through a 5 kΩ resistor or through a 1.5 kΩ resistor, depending on the VBRS bit value in the *PWR control register 4 (PWR_CR4)*.

The battery charging is enabled by setting VBE bit in the *PWR control register 4 (PWR_CR4)*, and automatically disabled in VBAT mode.

### 6.1.3 Voltage regulator

Two embedded linear voltage regulators supply all the digital circuitries, except for the Standby circuitry and the backup domain.

The voltage regulators are always enabled after a reset. Depending on the application modes, the $V_{CORE}$ supply is provided either by the main regulator (MR) or by the low-power regulator (LPR).

- In Run, Sleep and Stop0 modes, both regulators are enabled and the main regulator (MR) supplies full power to the $V_{CORE}$ domain (core, memories and digital peripherals).
- In low-power run and low-power sleep modes, the main regulator is off and the low-power regulator (LPR) supplies low power to the $V_{CORE}$ domain, preserving the contents of the registers and internal SRAM1 and SRAM2.
- In Stop1 and Stop2 modes the main regulator is off and the low-power regulator (LPR) supplies low power to the $V_{CORE}$ domain, preserving the contents of the registers and of internal SRAM1 and SRAM2.
- In Standby mode with SRAM2a content preserved (RRS bit is set in the *PWR control register 3 (PWR_CR3)*), the main regulator (MR) is off and the low-power regulator (LPR) provides the supply to SRAM2a only. The core and digital peripherals (except Standby circuitry and backup domain), SRAM1 and SRAM2b are powered off.
- In Standby mode, both regulators are powered off. The contents of the registers and of SRAM1 and SRAM2 is lost except for the Standby circuitry and the backup domain.
- In Shutdown mode, both regulators are powered off. When exiting from Shutdown mode, a power-on reset is generated. Consequently, the contents of the registers and of both SRAM1 and SRAM2 is lost, except for the backup domain.

## 6.2 Power supply supervisor

### 6.2.1 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.

Five BOR thresholds can be selected through option bytes.

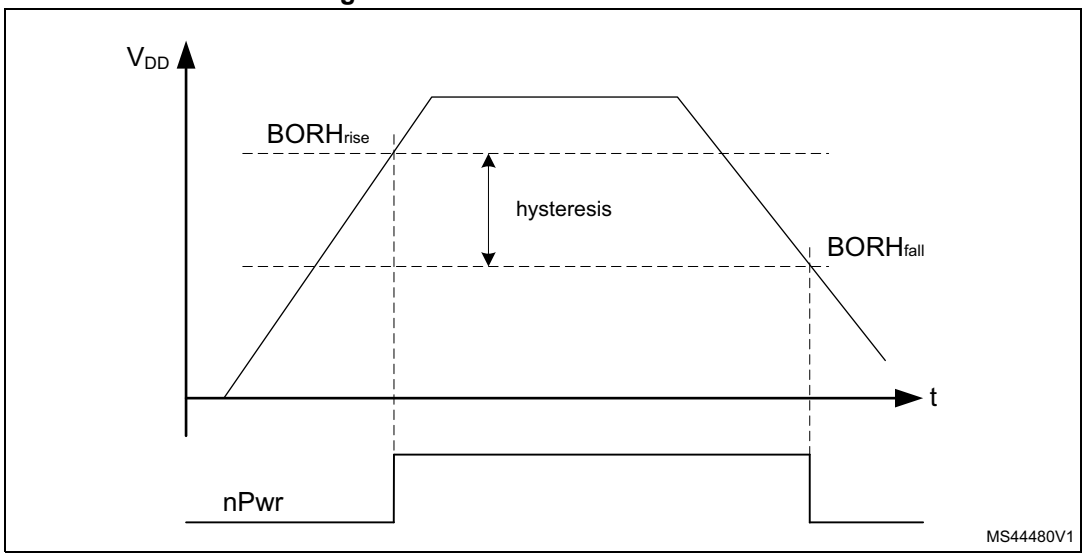
The BOR is active in all power modes except Shutdown mode, and cannot be disabled. The BOR mechanism needs to be enabled, and can be disabled at any time if needed.

#### Reset mode

During power-on, the BOR keeps the device under reset until the supply voltage $V_{DD}$ reaches the specified $V_{BORx}$ threshold. When $V_{DD}$ drops below the selected threshold, a device reset is generated. When $V_{DD}$ is above the $V_{BORx}$ upper limit, the device reset is released and the system can start.

For more details on the brown-out reset thresholds refer to the electrical characteristics section in the datasheet.

**Figure 8. Brown-out reset waveform**




1. The reset temporization $t_{RSTTEMPO}$ is present only for the BOR lowest threshold ($V_{BOR0}$).

### 6.2.2 Programmable voltage detector (PVD)

The PVD can be used to monitor the $V_{DD}$ power supply by comparing it to a threshold selected by the PLS[2:0] bits in the *PWR control register 2 (PWR_CR2)*.

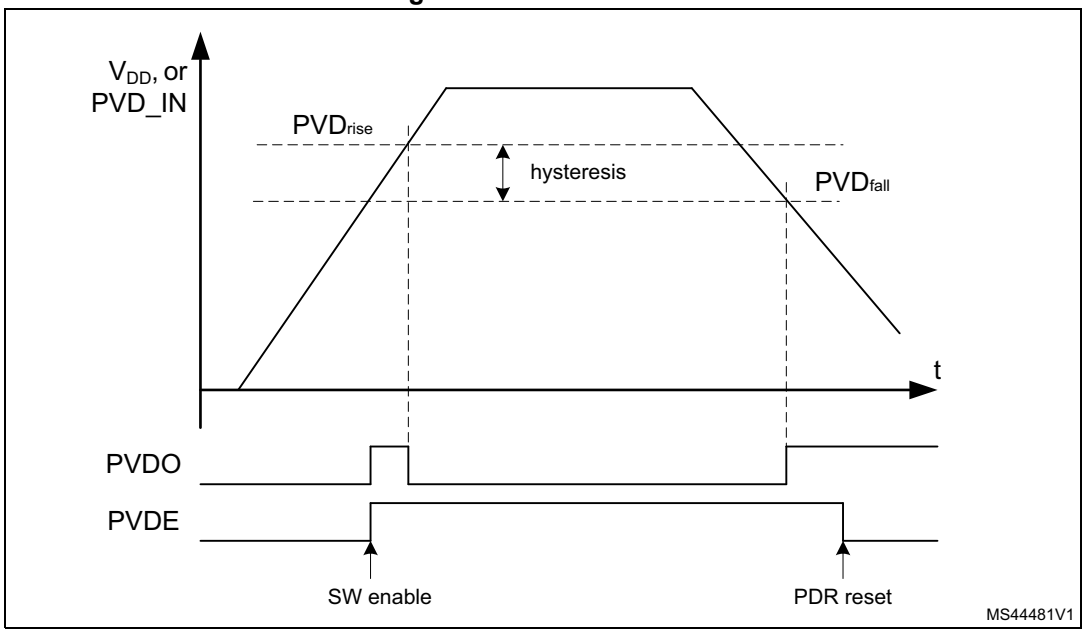
The PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case the voltage level on PVD_IN is compared to the internal VREFINT level.

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the *PWR status register 2 (PWR_SR2)*, to indicate if $V_{DD}$ or the voltage level on PVD_IN is higher or lower than the PVD threshold. This event is internally connected to the EXTI Line16 and can generate an interrupt if enabled through the EXTI registers.

The PVD output interrupt can be generated when $V_{DD}$ or the voltage level on PVD_IN drops below the PVD threshold, and/or when $V_{DD}$ or the voltage level on PVD_IN rises above the PVD threshold, depending on EXTI Line16 rising/falling edge configuration. If the EXTI Line16 is configured to rising/falling edge sensitivity (rising/falling edge of PVDO), the interrupt is generated when $V_{DD}$ or the voltage level on PVD_IN drops below/rises above the PVD threshold. As an example, the service routine can perform emergency shutdown tasks.

Figure 9. PVD thresholds

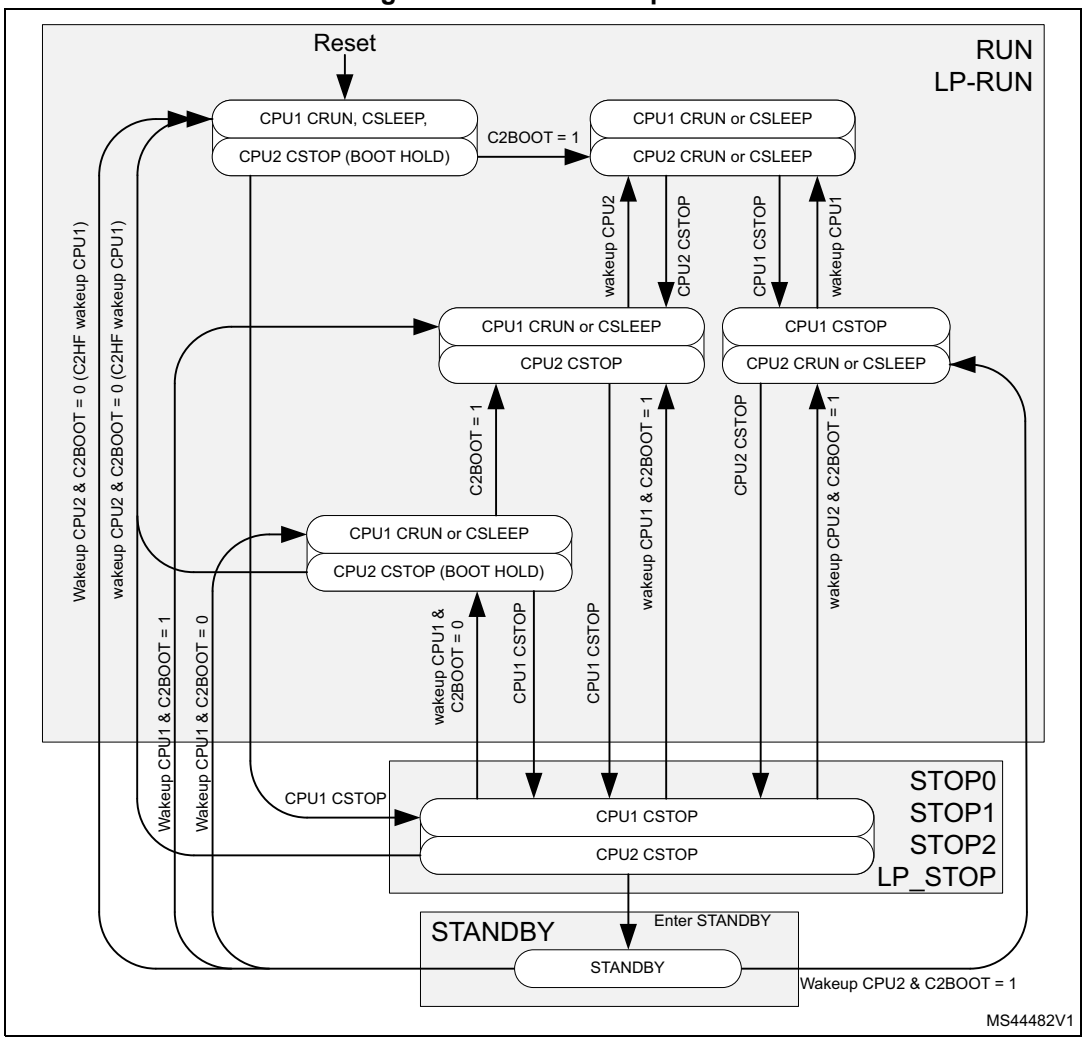



## 6.3 CPU2 boot

Booting of the CPU2 is controlled by the C2BOOT bit in *PWR control register 4 (PWR_CR4)* register. This allows the CPU1 to initialize the system after a reset or wake-up from system Low-power mode, before booting the CPU2.

- Following reset the CPU2 is prevented from booting by the C2BOOT bit. The CPU2 boots only once the CPU1 has set the C2BOOT bit.
- When exiting from system low power modes STOP0, STOP1 or STANDBY the C2BOOT bit controls the CPU2 booting.
  - When C2BOOT = 1, after a system low power mode the CPU2 boots when it is woken up via a wake-up source.
  - When C2BOOT = 0, after a system low power mode the CPU2 is prevented from booting. Instead on a CPU2 wake-up source the CPU1 is woken up via a C2HF. It is up to the CPU1 to boot the CPU2 by setting the C2BOOT bit.

**Figure 10. CPU2 boot options**




When the CPU2 is prevented from booting, the wake-up from Low-power mode boot procedure is the following:

- Before the CPU1 enters CSTOP mode it clears the C2BOOT bit.
- When CPU1 exits CSTOP:
  - When the system remains in Run mode, it sets the C2BOOT bit, and subsequently processes the wake-up event.
  - When the system exits from a low power mode from a CPU1 wake-up source, it initialize the system and sets the C2BOOT bit, and subsequently processes the wake-up event.
  - When the system exits from a low power mode from the C2HF wake-up source, it initializes the system and set the C2BOOT bit, and subsequently goes back to CSTOP.

- There is no special wake-up procedure for the CPU2. When the CPU2 wakes up, the system has been initialized by the CPU1. So the CPU2 can directly process the wake-up event.

When the system remains in Run mode (due to the Radio system) the CPU2 wakes up from CSTOP mode independently from the C2BOOT setting.

## 6.4 Low-power modes

By default, the microcontroller is in Run mode after a system or a power Reset and at least one CPU is in CRun mode executing code. Low-power modes are available to save power when the CPU does not need to be kept running, for example when it is waiting for an external event. The user has to select the mode that gives the best compromise between consumption, startup time and available wake-up sources.

The individual CPUs feature two low power modes, entered by the CPU when executing WFI, WFE or on return from an exception handler when SLEEPONEXIT is enabled.

- CSleep mode: when the CPU enters low power mode and SLEEPDEEP is disabled, Arm® “sleep mode”.
- CStop mode: when the CPU enters low power mode and SLEEPDEEP is enabled, Arm® “sleepdeep mode”.

The device features several low-power modes:

- *Sleep mode*: CPU clock off, all peripherals including CPU core peripherals (among them NVIC, SysTick) can run and wake up the CPU when an interrupt or an event occurs.
- *Low-power run mode (LP run)*: This mode is achieved when the system clock frequency is reduced below 2 MHz. The code is executed from the SRAM or from the flash memory. The regulator is in low-power mode to minimize the operating current.
- *Low-power sleep mode (LP sleep)*: This mode is entered from the Low-power run mode: CPU is off.
- *Stop0 mode*, *Stop1 mode* and *Stop2 mode*: the content of SRAM1, SRAM2 and of all registers is retained. All clocks in the $V_{CORE}$ domain are stopped, the PLL, the MSI, the HSI16 and the HSE are disabled. The LSI and the LSE can be kept running.

  The RTC can remain active (Stop mode with RTC, Stop mode without RTC).

  Some peripherals with the wake-up capability can enable the HSI16 RC during Stop mode to detect their wake-up condition.

  In Stop2 mode, most of the $V_{CORE}$ domain is put in a lower leakage mode. Stop1 offers the largest number of active peripherals and wake-up sources, a smaller wake-up time but a higher consumption compared with Stop2. In Stop0 mode, the main regulator remains ON, resulting in the fastest wake-up time, but with much higher consumption. The active peripherals and wake-up sources are the same as in Stop1 mode.

  The system clock, when exiting from Stop0, Stop1 or Stop2 mode, can be either MSI up to 48 MHz or HSI16, depending on the software configuration.
- *Standby mode*: $V_{CORE}$ domain is powered off. However, it is possible to preserve the SRAM2a contents:
  - Standby mode with SRAM2a retention when bit RRS is set in the *PWR control register 3 (PWR_CR3)*. In this case, SRAM2a is supplied by the low-power regulator.
  - Standby mode when bit RRS is cleared in the *PWR control register 3 (PWR_CR3)*.

In this case the main regulator and the low-power regulator are powered off.

All clocks in the $V_{CORE}$ domain are stopped, the PLL, the MSI, the HSI16 and the HSE are disabled. The LSI and the LSE can be kept running.

The RTC can remain active (Standby mode with RTC, Standby mode without RTC).

The system clock, when exiting Standby modes, is HSI16.

- *Shutdown mode*: $V_{CORE}$ domain is powered off. All clocks in the $V_{CORE}$ domain are stopped, the PLL, the MSI, the HSI16, the LSI and the HSE are disabled. The LSE can be kept running. The system clock, when exiting Shutdown mode, is MSI at 4 MHz. In this mode, the supply voltage monitoring is disabled and the product behavior is not guaranteed in case of a power voltage drop.

*Note: Stop, Standby, and Shutdown modes are only entered when both CPUs are in CStop mode.*

In addition, the power consumption in Run mode can be reduced by slowing down the system clocks, and/or by gating the clocks to the APB and AHB peripherals when they are unused.

The system operation mode depend on the CPU1, the CPU2 and the Radio sub-system operating mode. The system enters a low power mode only when all three sub-systems allow it to do so.

After a system reset the CPU1 is in CRUN mode. The CPU2 boots only if enabled by the CPU1 via the C2BOOT register bit. As long as the CPU1 does not boot the CPU2, the device operates as a single CPU system. The CPU1 can enter and wake up from system low power modes on its own.

When CPU2 has boot, the CPU1, CPU2 and Radio sub-systems can enter and wake up from system low power modes on their own. The different wake-up sources for the different sub-systems are detailed in *Table 21*.

**Table 21. Sub-system low power wake-up sources**

| Wake-up source | CPU1 | CPU2 | Radio |
|---|---|---|---|
| EXTI | From Stop modes | From Stop modes | Not available |
| RTC | From Stop and Standby modes | From Stop and Standby modes | Not available |
| WKUP | From Stop and Standby modes | From Stop and Standby modes | Not available |
| RADIO | From Stop | From Stop | Not available |
| RFWAKEUP | Not available | From Stop and Standby modes | From Stop and Standby modes |

The system low power mode to enter depends on the allowed mode selected by both CPUs in the LPMS bits of *PWR control register 1 (PWR_CR1)* and *PWR CPU2 control register 1 (PWR_C2CR1)*. This is also valid when CPU2 is kept in hold by C2BOOT.

*Figure 11* shows the operating modes state diagram. The CPU1, CPU2 and Radio sub-systems operate interdependently according to their own sub-system states. Each sub-system has its own wake-up sources, which allow it to wake up from Stop and Standby modes. For the device to be in Stop, Standby or Shutdown mode, all three sub-systems need to be in CStop. When one sub-system enters CRun mode the device enters Run mode.

**Figure 11. Low-power modes possible transitions**

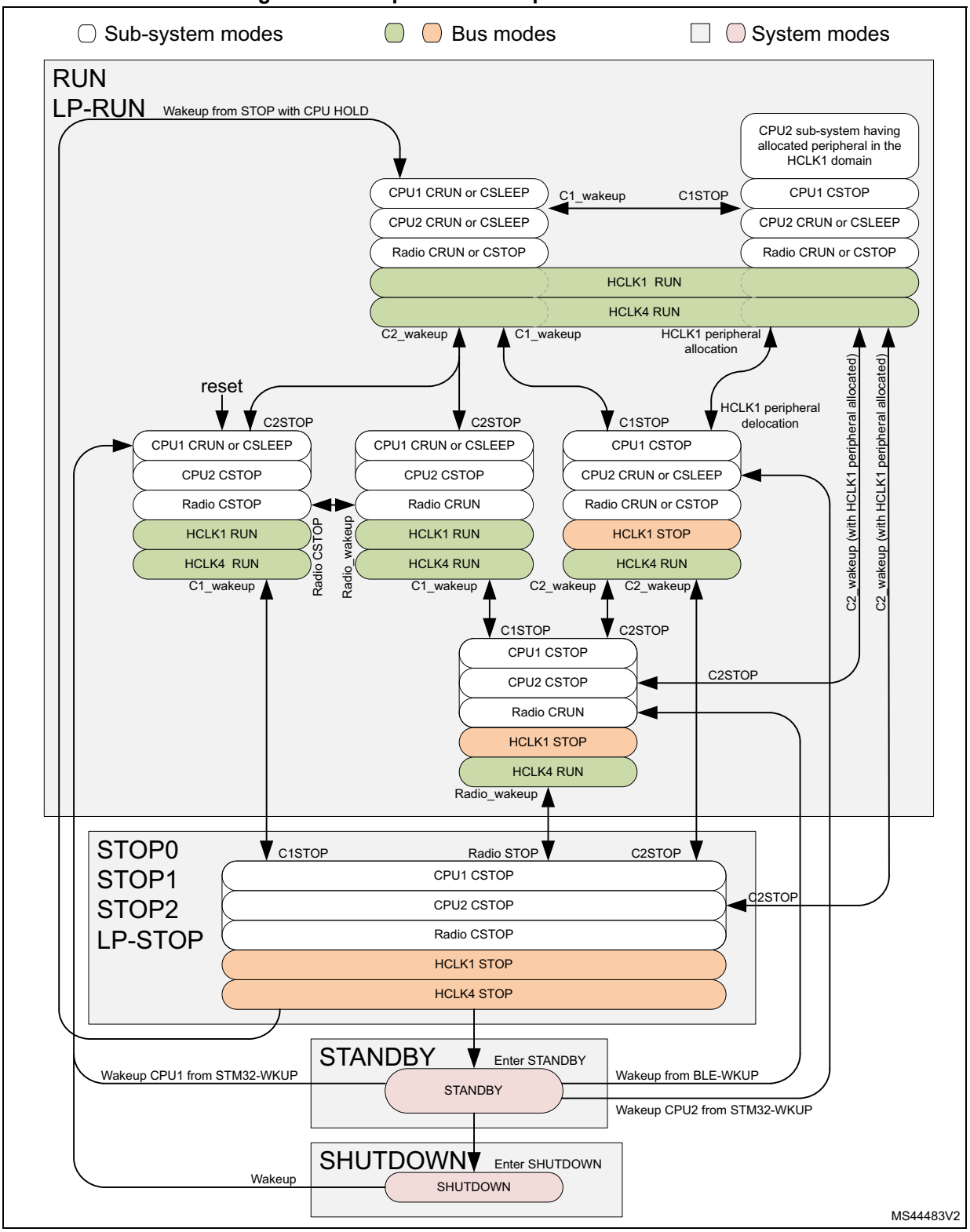


**Table 22. Low-power mode summary**

| Mode name | Entry | Wakeup source[1] | Wakeup system clock | Effect on clocks | Voltage regulator | |
|---|---|---|---|---|---|---|
| | | | | | MR | LPR |
| Sleep (Sleep-now or Sleep-on-exit) | WFI or Return from ISR | Any interrupt | Same as before entering Sleep mode | CPU clock OFF<br>No effect on other clocks or analog clock sources | ON | ON |
| | WFE | Wakeup event | | | | |
| Low-power run | Set LPR bit | Clear LPR bit | Same as Low-power run clock | None | OFF | |
| Low-power sleep | Set LPR bit + WFI or Return from ISR | Any interrupt | Same as before entering Low-power sleep mode | CPU clock OFF<br>No effect on other clocks or analog clock sources | | |
| | Set LPR bit + WFE | Wakeup event | | | | |
| Stop0 | LPMS = ”000” + SLEEPDEEP bit + WFI or Return from ISR or WFE | Any EXTI line (configured in the EXTI registers).<br>Specific peripherals events | HSI16 when STOPWUCK = 1 in RCC_CFGR.<br>MSI with the frequency before entering Stop mode when STOPWUCK = 0. | All clocks OFF except LSI and LSE | ON | ON |
| Stop1 | LPMS = ”001” + SLEEPDEEP bit + WFI or Return from ISR or WFE | | | | OFF | |
| Stop2 | LPMS = ”010” + SLEEPDEEP bit + WFI or Return from ISR or WFE | | | | | |
| Standby with SRAM2a | LPMS = ”011”+ Set RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE | WKUP pin edge, RTC event, LSECSS, external reset in NRST pin, IWDG reset | HSI16 | | | |
| Standby | LPMS = ”011” + Clear RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE | WKUP pin edge, RTC event, LSECSS, external reset in NRST pin, IWDG reset | | | | OFF |
| Shutdown | LPMS = ”1--” + SLEEPDEEP bit + WFI or Return from ISR or WFE | WKUP pin edge, RTC event, external reset in NRST pin | MSI 4 MHz | All clocks OFF except LSE | OFF | OFF |

1. Refer to *Table 23*.

**Table 23. Functionalities depending on system operating mode[1]**

| Peripheral | Run | Sleep | Low-power run | Low-power sleep | Stop0 | | Stop1 | | Stop2 | | Standby | | Shutdow | | VBAT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | - | Wakeup capability | - | Wakeup capability | - | Wakeup capability | - | Wakeup capability | - | Wakeup capability | |
| CPU1 | Y | - | Y | - | - | - | - | - | - | - | - | - | - | - | - |
| CPU2 | Y | - | Y | - | - | - | - | - | - | - | - | - | - | - | - |
| Radio-system (BLE, 802) | Y | Y | - | - | - | Y | - | Y | - | Y | - | Y[2] | - | - | - |
| Flash memory | Y | Y | O | O | R | - | R | - | R | - | R | - | R | - | R |
| SRAM1 | Y | O[4] | Y | O[4] | R | - | R | - | R | - | - | - | - | - | - |
| SRAM2a | Y | O[4] | Y | O[4] | R | - | R | - | R | - | O[3] | - | - | - | - |
| SRAM2b | Y | O[4] | Y | O[4] | R | - | R | - | R | - | - | - | - | - | - |
| Backup registers | Y | Y | Y | Y | R | - | R | - | R | - | R | - | R | - | R |
| Brown-out reset (BOR) | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | - | - | - |
| Programmable voltage detector (PVD) | O | O | O | O | O | O | O | O | O | O | - | - | - | - | - |
| DMA | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| High speed internal (HSI16) | O | O | O | O | O[5] | - | O[5] | - | - | - | - | - | - | - | - |
| Oscillator HSI48 | O | O | - | - | - | - | - | - | - | - | - | - | - | - | - |
| High speed external (HSE) | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| Low speed internal (LSI) | O | O | O | O | O | - | O | - | O | - | O | - | - | - | - |
| Low speed external (LSE) | O | O | O | O | O | - | O | - | O | - | O | - | O | - | O |
| Multi-speed internal (MSI) | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| Clock security system (CSS) | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| Clock security system on LSE | O | O | O | O | O | O | O | O | O | O | O | O | - | - | - |
| RTC / Auto wake-up | O | O | O | O | O | O | O | O | O | O | O | O | O | O | O |
| Number of RTC tamper pins | 1 | 1 | 1 | 1 | 1 | O | 1 | O | 1 | O | 1 | O | 1 | O | 1 |
| USART1 | O | O | O | O | O[6] | O[6] | O[6] | O[6] | - | - | - | - | - | - | - |
| I2C1 | O | O | O | O | O[7] | O[7] | O[7] | O[7] | - | - | - | - | - | - | - |
| SPIx (x=1) | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| ADC1 | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| Temperature sensor | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| Timers (TIMx) | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| Low-power timer 1 (LPTIM1) | O | O | O | O | O | O | O | O | O | O | - | - | - | - | - |
| Low-power timer 2 (LPTIM2) | O | O | O | O | O | O | O | O | - | - | - | - | - | - | - |

**Table 23. Functionalities depending on system operating mode[1] (continued)**

| Peripheral | Run | Sleep | Low-power run | Low-power sleep | Stop0 | | Stop1 | | Stop2 | | Standby | | Shutdow | | VBAT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | - | Wakeup capability | - | Wakeup capability | - | Wakeup capability | - | Wakeup capability | - | Wakeup capability | |
| Independent watchdog (IWDG) | O | O | O | O | O | O | O | O | O | O | O | O | - | - | - |
| Window watchdog (WWDG) | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| SysTick timer | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| True random number generator (RNG) | O | O | - | - | - | - | - | - | - | - | - | - | - | - | - |
| AES hardware accelerator | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| CRC calculation unit | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| IPCC | O | - | O | - | - | - | - | - | - | - | - | - | - | - | - |
| HSEM | O | - | O | - | - | - | - | - | - | - | - | - | - | - | - |
| PKA | O | O | O | O | - | - | - | - | - | - | - | - | - | - | - |
| GPIOs | O | O | O | O | O | O | O | O | O | O | (8) | 2 pins (9) | (10) | 2 pins (9) | - |

1. Legend: Y = Yes (enabled). O = Optional (disabled by default, can be enabled by software). R = Data retained. - = Not available. Gray cells indicate Wakeup capability.
2. SRAM2a content needs to be retained via the PWR_CR3.RRS bit.
3. The SRAM2a content can optionally be retained when the PWR_CR3.RRS bit is set,
4. The SRAM clock can be gated on or off.
5. Some peripherals with wake-up from Stop capability can request HSI16 to be enabled. In this case, HSI16 is woken up by the peripheral, and only feeds the peripheral that requested it. HSI16 is automatically put off when the peripheral does not need it anymore.
6. UART reception is functional in Stop mode, and generates a wake-up interrupt on Start, address match or received frame event.
7. I2C address detection is functional in Stop mode, and generates a wake-up interrupt in case of address match.
8. I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
9. The I/Os with wake-up from Standby/Shutdown capability are PA0 and PA2.
10. I/Os can be configured with internal pull-up, pull-down or floating in Shutdown mode but the configuration is lost when exiting Shutdown mode.

### Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop0, Stop1, Stop2, Standby or Shutdown mode while the debug features are used. This is because the CPU1 core is no longer clocked.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively. For more details refer to *Section 32.3.5: Debug and low power modes.*

### 6.4.1 Run mode

#### Slowing down system clocks

In Run mode, the speed of the system clocks (SYSCLK, HCLK, PCLK) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down the peripherals before entering Sleep mode.

For more details, refer to *Section 7.4.3: RCC clock configuration register (RCC_CFGR)*.

#### Peripheral clock gating

In Run mode, the HCLK and PCLK for individual peripherals and memories can be stopped at any time to reduce the power consumption.

To further reduce the power consumption in Sleep mode, the peripheral clocks can be disabled prior to executing WFI or WFE instructions.

The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

Disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bit in the RCC_AHBxSMENR and RCC_APBxSMENR registers.

### 6.4.2 Low-power run mode (LP run)

To further reduce the consumption when the system is in Run mode, the regulator can be configured in low-power mode. In this mode, the system frequency must not exceed 2 MHz. The Radio sub-system cannot be used in low-power Run mode.

Refer to the product datasheet for more details on voltage regulator and peripherals operating conditions.

#### I/O states in Low-power run mode

In Low-power run mode, all I/O pins keep the same state as in Run mode.

#### Entering Low-power run mode

To enter the Low-power run mode, proceed as follows:

1. Optional: Jump into the SRAM and power-down the flash memory by setting the FPDR bit in *PWR control register 1 (PWR_CR1)* and *PWR CPU2 control register 1 (PWR_C2CR1)*.
2. Decrease the system clock frequency below 2 MHz.
3. Force the regulator in low-power mode by setting the LPR bit in the *PWR control register 1 (PWR_CR1)*.

Refer to *Table 24* on how to enter the Low-power run mode.

#### Exiting Low-power run mode

To exit the Low-power run mode, proceed as follows (refer to *Table 24*):

1. Force the regulator in main mode by clearing the LPR bit in the *PWR control register 1 (PWR_CR1)*.
2. Wait until REGLPF bit is cleared in the *PWR status register 2 (PWR_SR2)*.
3. Increase the system clock frequency.

**Table 24. Low-power run**

| Low-power run mode | Description |
|---|---|
| Mode entry | Decrease the system clock frequency below 2 MHz<br>LPR = 1 |
| Mode exit | LPR = 0<br>Wait until REGLPF = 0<br>Increase the system clock frequency |
| Wakeup latency | Regulator wake-up time from low-power mode |

### 6.4.3 Entering Low-power mode

Low power modes are entered by the MCU by executing WFI (Wait for Interrupt), or WFE (Wait for Event) instructions, or when the SLEEPONEXIT bit in the CPU1 System control register is set on Return from ISR.
Entering Low-power mode through WFI or WFE is executed only if no interrupt is pending or no event is pending.

### 6.4.4 Exiting Low-power mode

From Sleep modes, and Stop modes the MCU exit Low-power mode depending on the way the mode was entered:

- If the WFI instruction or Return from ISR was used to enter the Low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the Low-power mode, the MCU exits the Low-power mode as soon as an event occurs. The wake-up event can be generated either by an NVIC IRQ interrupt, or by an event.
  - In the first case, when SEVONPEND = 0 in the CPU System control register, enabling an interrupt in the peripheral control register and in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared. Only NVIC interrupts with sufficient priority wake-up and interrupt the MCU.
    When SEVONPEND = 1 in the CPU System control register, enabling an interrupt in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared. All NVIC interrupts wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with sufficient priority wake up and interrupt the MCU.
  - In the second case, configuring an EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set. It may be necessary to clear the interrupt flag in the peripheral.

From Standby and Shutdown modes the MCU exits Low-power mode through an external reset (NRST pin), an IWDG reset, a rising/falling edge on one of the enabled WKUPx pins,

or an RTC event (see *Section 24: Real-time clock (RTC)*), or a Radio event (for Standby only).

After waking up from Standby or Shutdown mode, program execution restarts in the same way as after a Reset (boot pin sampling, option bytes loading, reset vector is fetched, etc.).

The system mode when the CPU wakes up from CStop mode can be determined from the CnSTOPF and CnSBF in *PWR extended status and status clear register (PWR_EXTSCR)*.

**Table 25. CPU CSTOP wake-up vs. system operating mode**

| System mode | CPU1 | | CPU2 | | CPU1 wake-up | CPU2 wake-up |
|---|---|---|---|---|---|---|
| | C1SBF | C1STOPF | C2SBF | C2STOPF | | |
| Run | 0 | 0 | 0 | 0 | Wakeup from Run | Wakeup from Run |
| | 0 | 1 | 0 | 0 | Wakeup from STOP, but system is already in Run due to CPU2 | Wakeup from Run |
| | 0 | 0 | 0 | 1 | Wakeup from Run | Wakeup from STOP, but system is already in Run due to CPU1 |
| | 1 | 0 | 0 | 0 | Wakeup from STANDBY, but system is already in Run due to CPU2 | Wakeup from Run |
| | 0 | 0 | 1 | 0 | Wakeup from Run | Wakeup from STANDBY, but system is already in Run due to CPU1 |
| | 1 | 1 | 0 | 0 | Wakeup from STANDBY followed by STOP, but system is already in Run due to CPU2 | Wakeup from Run |
| | 0 | 0 | 1 | 1 | Wakeup from Run | Wakeup from STANDBY followed by STOP, but system is already in Run due to CPU1 |
| Stop | 0 | 1 | 0 | 1 | Wakeup from STOP (CPU2 still in CSTOP) | Wakeup from STOP (CPU1 still in CSTOP) |
| | 1 | 1 | 0 | 1 | Wakeup from STOP after the system has been in STANDBY (CPU2 still in CSTOP) | Wakeup from STOP (CPU1 still in CSTOP) |
| | 0 | 1 | 1 | 1 | Wakeup from STOP (CPU2 is still in CSTOP) | Wakeup from STOP after the system having been in STANDBY (CPU1 still in CSTOP) |
| Standby | 1 | 0 | 1 | 0 | Wakeup from STANDBY (CPU2 still in CSTOP) | Wakeup from STANDBY (CPU1 still in CSTOP) |
| N.A. | Others | | | | Not valid, does not occur | |

## 6.4.5 Sleep mode

### I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

### Entering Sleep mode

Sleep mode is entered according to *Entering Low-power mode*, when the SLEEPDEEP bit in the CPU System control register is cleared (see *Table 26).*

### Exiting Sleep mode

The MCU exits from Sleep mode (see *Table 26)* as indicated in *Exiting Low-power mode*.

**Table 26. Sleep mode**

| Sleep-now mode | Description |
|---|---|
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while:<br>– SLEEPDEEP = 0<br>– No interrupt (for WFI) or event (for WFE) is pending<br>Refer to the Cortex® System control register. |
| | On return from ISR while:<br>– SLEEPDEEP = 0 and<br>– SLEEPONEXIT = 1<br>– No interrupt is pending<br>Refer to the Cortex® System control register. |
| Mode exit | If WFI or return from ISR was used for entry:<br>Interrupt: refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table*<br>If WFE was used for entry and SEVONPEND = 0:<br>Wakeup event: refer to *Section 13.5: Interrupt list*<br>If WFE was used for entry and SEVONPEND = 1:<br>Interrupt even when disabled in NVIC: refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table* or Wakeup event: refer to *Section 13.5: Interrupt list* |
| Wakeup latency | None |

## 6.4.6 Low-power sleep mode (LP sleep)

Refer to the product datasheet for more details on voltage regulator and peripherals operating conditions.

### I/O states in Low-power sleep mode

In Low-power sleep mode, all I/O pins keep the same state as in Run mode.

### Entering Low-power sleep mode

Low-power sleep mode is entered from Low-power run mode as described in *Section 6.4.3*, when the SLEEPDEEP bit in the Cortex® System control register is cleared.

Refer to *Table 27* for details on how to enter the Low-power sleep mode.

### Exiting Low-power sleep mode

The low-power Sleep mode is exited as described in *Section 6.4.4*. When exiting Low-power sleep mode by issuing an interrupt or an event, the MCU is in Low-power run mode.

Refer to *Table 27* for details on how to exit the Low-power sleep mode.

**Table 27. Low-power sleep**

| Low-power sleep-now mode | Description |
|---|---|
| Mode entry | Low-power sleep mode is entered from the Low-power run mode.<br>WFI (Wait for Interrupt) or WFE (Wait for Event) while:<br>– SLEEPDEEP = 0<br>– No interrupt (for WFI) or event (for WFE) is pending<br>Refer to the Cortex® System control register. |
| | Low-power sleep mode is entered from the Low-power run mode.<br>On return from ISR while:<br>– SLEEPDEEP = 0 and<br>– SLEEPONEXIT = 1<br>– No interrupt is pending<br>Refer to the Cortex® System control register. |
| Mode exit | If WFI or Return from ISR was used for entry:<br>Interrupt: refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table*<br>If WFE was used for entry and SEVONPEND = 0:<br>Wakeup event: refer to *Section 13.5: Interrupt list*<br>If WFE was used for entry and SEVONPEND = 1:<br>Interrupt even when disabled in NVIC: refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table*<br>Wakeup event: refer to *Section 13.5: Interrupt list*<br>After exiting the Low-power sleep mode, the MCU is in Low-power run mode. |
| Wakeup latency | None |

## 6.4.7 Stop0 mode

The Stop0 mode is based on the CPU deep sleep mode combined with the peripheral clock gating. The voltage regulator is configured in main regulator mode. In Stop0 mode, all clocks in the $V_{CORE}$ domain are stopped; the PLL, the MSI, the HSI16 and the HSE oscillators are disabled. Some peripherals with wake-up capability (I2Cx (x=1), and USART1) can switch on the HSI16 to receive a frame, and switch off the HSI16 after receiving the frame if it is not a wake-up frame. In this case, the HSI16 clock is propagated only to the peripheral requesting it.

SRAM1, SRAM2 and register contents are preserved.

The BOR is always available in Stop0 mode. The consumption increases when thresholds higher than $V_{BOR0}$ are used.

### I/O states in Stop0 mode

In the Stop0 mode, all I/O pins keep the same state as in the Run mode.

### Entering Stop0 mode

The Stop0 mode is entered according to *Section 6.4.3*, when the SLEEPDEEP bit in the

Cortex System control register is set (see *Table 28*). Before entering the mode, the system clock must be set to HSI16 clock.

If flash memory programming is ongoing, the Stop0 mode entry is delayed until the operation is completed.

If an access to the APB domain is ongoing, the Stop0 mode entry is delayed until the APB access is finished.

In Stop0 mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started, it cannot be stopped except by a reset. See *Section 25.3: IWDG functional description*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the *RCC backup domain control register (RCC_BDCR)*.
- Internal RC oscillator (LSI): this is configured by the LSIxON bit in the *RCC control/status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the *RCC backup domain control register (RCC_BDCR)*.

Some peripherals can be used in Stop0 mode and add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock, namely LPTIM1, LPTIM2, I2Cx (x=1), USART1.

The PVD can be used in Stop0 mode. If not needed, it must be disabled by software to reduce power consumption.

The ADC and the temperature sensor can consume power during the Stop0 mode, unless they are disabled before entering this mode.

### Exiting Stop0 mode

The Stop0 mode is exited according to what indicated in *Section 6.4.4*.

Refer to *Table 28* for details on how to exit Stop0 mode.

When exiting Stop0 mode by issuing an interrupt or a wake-up event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in *RCC clock configuration register (RCC_CFGR)*. The MSI oscillator is selected as system clock if the bit STOPWUCK is cleared. The wake-up time is shorter when HSI16 is selected as wake-up system clock. The MSI selection enables wake-up at higher frequency, up to 48 MHz.

When the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop0 mode with HSI16. By keeping the internal regulator ON during Stop0 mode, the consumption is higher but the startup time is reduced.

When exiting Stop0 mode, the MCU is either in Run mode or in Low-power run mode if the bit LPR is set in the same register.

**Table 28. Stop0 mode**

| Stop0 mode | Description |
|---|---|
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– No interrupt (for WFI) or event (for WFE) is pending<br>– LPMS = “000” in PWR_CR1 and/or PWR_C2CR1 or higher |
| | On Return from ISR while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– SLEEPONEXIT = 1<br>– No interrupt is pending<br>– LPMS = “000” in PWR_CR1 and/or PWR_C2CR1 or higher |
| | *Note: To enter Stop0 mode, all EXTI line pending bits (in EXTI pending register (EXTI_PR1), and EXTI pending register (EXTI_PR2)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop0 mode entry procedure is ignored and program execution continues.* |
| Mode exit | If WFI or Return from ISR was used for entry:<br>Any EXTI line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table*.<br>If WFE was used for entry and SEVONPEND = 0:<br>Any EXTI line configured in event mode. Refer to *Section 13.5: Interrupt list*.<br>If WFE was used for entry and SEVONPEND = 1:<br>Any EXTI line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table*.<br>Wakeup event: refer to *Section 13.5: Interrupt list* |
| Wakeup latency | Longest wake-up time between: MSI or HSI16 wake-up time and flash memory wake-up time from Stop0 mode. |

### 6.4.8 Stop1 mode

The Stop1 mode is the same as Stop0 mode except that the main regulator is OFF, and only the low-power regulator is ON. Stop1 mode can be entered from Run mode and from Low-power run mode. Before entering the mode, the system clock must be set to HSI16 clock.

Refer to *Table 29* for details on how to enter and exit Stop1 mode.

**Table 29. Stop1 mode**

| Stop1 mode | Description |
|---|---|
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– No interrupt (for WFI) or event (for WFE) is pending<br>– LPMS = “001” in PWR_CR1 and/or PWR_C2CR1 or higher |
| | On Return from ISR while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– SLEEPONEXIT = 1<br>– No interrupt is pending<br>– LPMS = “001” in PWR_CR1 and/or PWR_C2CR1 or higher |
| | *Note: To enter Stop1 mode, all EXTI line pending bits (in EXTI pending register (EXTI_PR1), and EXTI pending register (EXTI_PR2)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop1 mode entry procedure is ignored and program execution continues.* |
| Mode exit | If WFI or Return from ISR was used for entry:<br>Any EXTI line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table*.<br>If WFE was used for entry and SEVONPEND = 0:<br>Any EXTI line configured in event mode. Refer to *Section 13.5: Interrupt list*.<br>If WFE was used for entry and SEVONPEND = 1:<br>Any EXTI line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table*.<br>Wakeup event: refer to *Section 13.5: Interrupt list*. |
| Wakeup latency | Longest wake-up time between: MSI or HSI16 wake-up time and regulator wake-up time from Low-power mode + flash memory wake-up time from Stop1 mode. |

## 6.4.9 Stop2 mode

The Stop2 mode is based on the CPU deepsleep mode combined with peripheral clock gating. In Stop2 mode, all clocks in the $V_{CORE}$ domain are stopped, the PLL, the MSI, the HSI16 and the HSE oscillators are disabled.

SRAM1, SRAM2 and register contents are preserved.

The BOR is always available in Stop2 mode. The consumption is increased when thresholds higher than $V_{BOR0}$ are used.

*Note: The comparators, LPUART and LPTIM1 outputs are forced to low speed (OSPEEDy = 00) during Stop2 mode.*

### I/O states in Stop2 mode

In the Stop2 mode, all I/O pins keep the same state as in the Run mode.

### Entering Stop2 mode

The Stop2 mode is entered as described in *Section 6.4.3*, when the SLEEPDEEP bit in the Cortex System control register is set (see *Table 30*). Before entering in this mode, the system clock must be set to HSI16 clock.

Stop2 mode can only be entered from Run mode. It is not possible to enter Stop2 mode from the Low-power run mode.

If flash memory programming is ongoing, the Stop2 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop2 mode entry is delayed until the APB access is finished.

In Stop2 mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset. See *Section 25.3: IWDG functional description*.
- real-time clock (RTC): this is configured by the RTCEN bit in the *RCC backup domain control register (RCC_BDCR)*
- Internal RC oscillator (LSI): this is configured by the LSIxON bit in the *RCC control/status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the *RCC backup domain control register (RCC_BDCR)*.

Some peripherals can be used in Stop2 mode and add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock, namely LPTIM1.

The PVD can be used in Stop2 mode. If not needed, it must be disabled by software to reduce power consumption.

The ADC and the temperature sensor can consume power during Stop2 mode, unless they are disabled before entering this mode.

All the peripherals that cannot be enabled in Stop2 mode must be either disabled by clearing the Enable bit in the peripheral itself, or put under reset state by setting the corresponding bit in the

- *RCC AHB1 peripheral reset register (RCC_AHB1RSTR)*
- *RCC AHB2 peripheral reset register (RCC_AHB2RSTR)*
- *RCC AHB3 and AHB4 peripheral reset register (RCC_AHB3RSTR)*
- *RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)*
- *RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)*
- *RCC APB2 peripheral reset register (RCC_APB2RSTR)*
- *RCC APB3 peripheral reset register (RCC_APB3RSTR)*

### Exiting Stop2 mode

The Stop2 mode is exited according to *Section 6.4.4* (see *Table 30*).

When exiting Stop2 mode by issuing an interrupt or a wake-up event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in *RCC clock configuration register (RCC_CFGR)*. The MSI oscillator is selected as system clock if the bit STOPWUCK is

cleared. The wake-up time is shorter when HSI16 is selected as wake-up system clock. The MSI selection allows wake-up at higher frequency, up to 48 MHz.

When exiting Stop2 mode, the MCU is in Run mode (Range 1 or Range 2 depending on VOS bit in PWR_CR1).

**Table 30. Stop2 mode**

| Stop2 mode | Description |
|---|---|
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– No interrupt (for WFI) or event (for WFE) is pending<br>– LPMS = “010” in PWR_CR1 and/or PWR_C2CR1 or higher |
| | On return from ISR while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– SLEEPONEXIT = 1<br>– No interrupt is pending<br>– LPMS = “010” in PWR_CR1 and/or PWR_C2CR1 or higher<br>*Note: To enter Stop2 mode, all EXTI line pending bits in EXTI pending register (EXTI_PR1), and EXTI pending register (EXTI_PR2)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop mode entry procedure is ignored and program execution continues.* |
| Mode exit | If WFI or Return from ISR was used for entry:<br>Any EXTI line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table*.<br>If WFE was used for entry and SEVONPEND = 0:<br>Any EXTI line configured in event mode. Refer to *Section 13.5: Interrupt list*.<br>If WFE was used for entry and SEVONPEND = 1:<br>Any EXTI line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to *Table 54: CPU1 vector table* and *Table 55: CPU2 vector table*.<br>Any EXTI line configured in event mode. Refer to *Section 13.5: Interrupt list*. |
| Wakeup latency | Longest wake-up time between: MSI or HSI16 wake-up time and regulator wake-up time from Low-power mode + flash wake-up time from Stop2 mode. |

### 6.4.10 Standby mode

Standby mode makes it possible to achieve the lowest power consumption with BOR. It is based on the CPU deepsleep mode, with the voltage regulators disabled (except when SRAM2 content is preserved). The PLL, the HSI16, the MSI and the HSE oscillators are also switched off.

SRAM1 and register contents are lost except for registers in the Backup domain and Standby circuitry (see *Figure 7*). SRAM2 content can be preserved if the bit RRS is set in the *PWR control register 3 (PWR_CR3)*. In this case the Low-power regulator is ON and provides the supply to SRAM2 only.

The BOR is always available in Standby mode. The consumption is increased when thresholds higher than $V_{BOR0}$ are used.

### I/O states in Standby mode

In Standby mode, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x=A,B,C,E,H)), or with a pull-down (refer to PWR_PDCRx registers (x=A,B,C,E,H)), or can be kept in analog state.

The RTC outputs on PC14 and PC15 used for LSE are functional. Two wake-up pins (WKUPx, x=1,4) and the RTC tamper are available.

### Entering Standby mode

Standby mode is entered according to *Section 6.4.3*, when the SLEEPDEEP bit in the Cortex System control register is set.

Refer to *Table 31* for details on how to enter Standby mode.

In Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. See *Section 25.3: IWDG functional description*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR).
- Internal RC oscillator (LSI): this is configured by the LSIxON bit in the Control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)

### Exiting Standby mode

Standby mode is exited according to *Section 6.4.4*. The SBF status flag in the *PWR control register 3 (PWR_CR3)* indicates that the MCU was in Standby mode. All registers are reset after wake-up from Standby except for *PWR control register 3 (PWR_CR3)*.

Refer to *Table 31* for more details on how to exit Standby mode.

**Table 31. Standby mode**

| Standby mode | Description |
|---|---|
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– No interrupt (for WFI) or event (for WFE) is pending<br>– LPMS = “011” in PWR_CR1 and/or PWR_C2CR1 or higher<br>– WUFx bits are cleared in power status register 1 (PWR_SR1) |
| | On return from ISR while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– SLEEPONEXIT = 1<br>– No interrupt is pending<br>– LPMS = “011” in PWR_CR1 and/or PWR_C2CR1 or higher<br>– WUFx bits are cleared in power status register 1 (PWR_SR1)<br>The RTC flag corresponding to the chosen wake-up source (RTC Alarm A, RTC Alarm B, RTC wake-up, tamper or timestamp flags) is cleared. |

**Table 31. Standby mode (continued)**

| Standby mode | Description |
|---|---|
| Mode exit | WKUPx pin edge, RTC event, external Reset in NRST pin, IWDG Reset, BOR reset |
| Wakeup latency | Reset phase |

### 6.4.11 Shutdown mode

The Shutdown mode allows to achieve the lowest power consumption. It is based on the deepsleep mode, with the voltage regulator disabled. The $V_{CORE}$ domain is consequently powered off. The PLL, the HSI16, the MSI, the LSI and the HSE oscillators are also switched off.

SRAM1, SRAM2 and register contents are lost except for registers in the Backup domain. The BOR is not available in Shutdown mode. No power voltage monitoring is possible in this mode, therefore the switch to Backup domain is not supported.

#### I/O states in Shutdown mode

In the Shutdown mode, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x=A,B,C,E,H)), or with a pull-down (refer to PWR_PDCRx registers (x=A,B,C,E,H)), or can be kept in analog state.

However this configuration is lost when exiting Shutdown mode due to the power-on reset.

PC14 and PC15 used for LSE are also functional. Two wake-up pins (WKUPx, x=1,4) and the RTC tamper are available.

#### Entering Shutdown mode

The Shutdown mode is entered according to *Entering Low-power mode*, when the SLEEPDEEP bit in the Cortex System control register is set.

Refer to *Table 32* for details on how to enter Shutdown mode.

In Shutdown mode, the following features can be selected by programming individual control bits:

- Real-time clock (RTC): this is configured by the RTCEN bit in the *RCC backup domain control register (RCC_BDCR)*. Caution: in case of $V_{DD}$ power-down the RTC content is lost.
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the *RCC backup domain control register (RCC_BDCR)*.

#### Exiting Shutdown mode

The Shutdown mode is exited according to *Exiting Low-power mode*. A power-on reset occurs when exiting from Shutdown mode. All registers (except for the ones in the Backup domain) are reset after wake-up from Shutdown.

Refer to *Table 32* for more details on how to exit Shutdown mode.

**Table 32. Shutdown mode**

| Shutdown mode | Description |
|---|---|
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– No interrupt (for WFI) or event (for WFE) is pending<br>– LPMS = “1XX” in PWR_CR1 and PWR_C2CR1<br>– WUFx bits are cleared in power status register 1 (PWR_SR1) |
| | On return from ISR while:<br>– SLEEPDEEP bit is set in Cortex System control register<br>– SLEEPONEXT = 1<br>– No interrupt is pending<br>– LPMS = “1XX” in PWR_CR1 and PWR_C2CR1<br>– WUFx bits are cleared in power status register 1 (PWR_SR1)<br>The RTC flag corresponding to the chosen wake-up source (RTC Alarm A, RTC Alarm B, RTC wake-up, tamper or timestamp flags) is cleared. |
| Mode exit | WKUPx pin edge, RTC event, external Reset in NRST pin |
| Wakeup latency | Reset phase |

### 6.4.12 Auto wake-up from Low-power mode

The RTC can be used to wake up the MCU from Low-power mode without depending on an external interrupt (Auto-wake-up mode). The RTC provides a programmable time base for waking up from Stop (0, 1 or 2) or Standby mode at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the *RCC backup domain control register (RCC_BDCR)*:

- Low-power 32.768 kHz external crystal oscillator (LSE OSC)
  This clock source provides a precise time base with very low-power consumption.
- Low-power internal RC oscillator (LSI)
  This oscillator is designed to add minimum power consumption.

To wake up from Stop mode with an RTC alarm event, it is necessary to:

- Configure the EXTI Line 18 to be sensitive to rising edge
- Configure the RTC to generate the RTC alarm

To wake up from Standby mode, there is no need to configure the EXTI Line 18.

To wake up from Stop mode with an RTC wake-up event, it is necessary to:

- Configure the EXTI Line 20 to be sensitive to rising edge
- Configure the RTC to generate the RTC alarm

To wake up from Standby mode, there is no need to configure the EXTI Line 20.

## 6.5 Real-time radio information

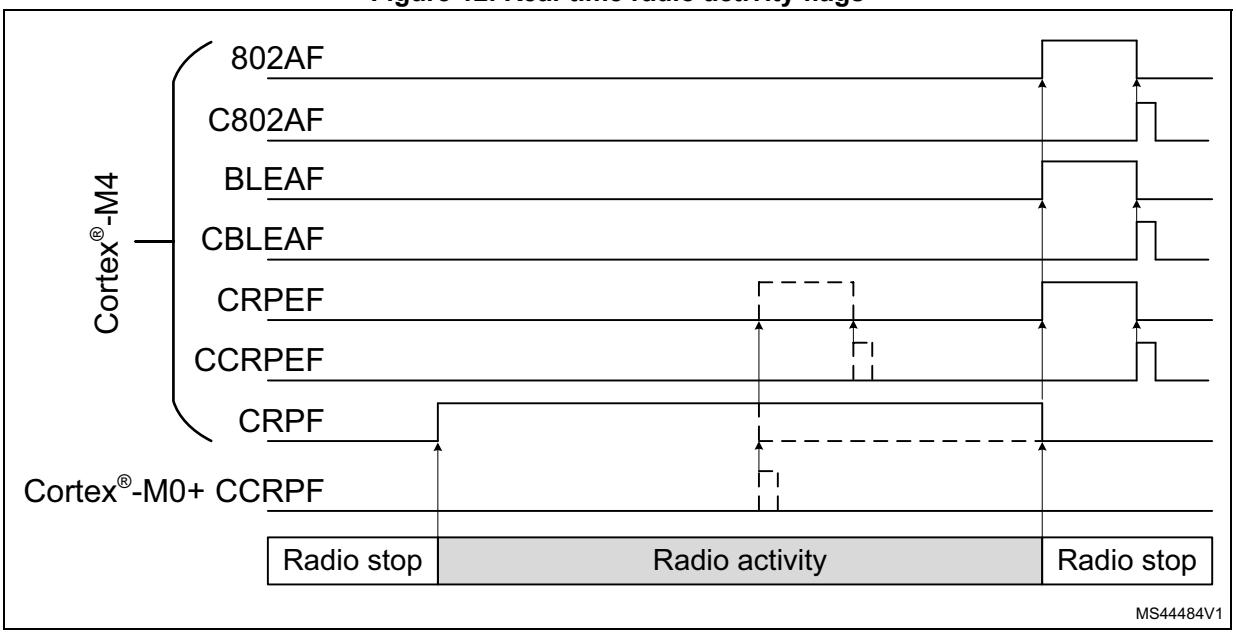
The PWR provides flags indicating the real time operation on the radio:

- IEEE802.15.4 radio activity end interrupt flag
- BLE radio activity end interrupt flag
- Critical radio phase flag
- Critical radio phase end interrupt flag

These flags may be used by the CPU1 to determine the radio activity.

The basic timing relation for the different flags is shown in *Figure 11*.

**Figure 12. Real-time radio activity flags**




When enabled, the radio activity end interrupt flag 802AF indicates the end of an IEEE802.15.4 radio activity period, and is generated when the radio enters Cstop mode.

When enabled, the radio activity end interrupt flag BLEAF indicates the end of a BLE radio activity period, and is generated when the radio enters Cstop mode.

The critical radio phase flag indicates the critical real time phase of the radio, where access to the flash memory must not be blocked by erase or program operations (all erase and program operations are suspended by the CPU2 in the flash memory interface). The end of the critical radio phase may be triggered by the CPU2 clearing the critical radio phase flag CRPF, and is eventually cleared at the end of the IEEE802.15.4 or BLE radio activity.

When enabled, the critical radio phase end interrupt flag CRPEF indicates the end of a radio critical phase, and is generated when the radio critical phase ends.

## 6.6 PWR registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 6.6.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0x0000 0200. This register is reset after wake-up from Standby mode, except for bits [2:0].

| 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. | LPR | Res. | Res. | Res. | Res. | Res. | DBP | Res. | Res. | FPDS | FPDR | Res. | LPMS[2:0] | | |
| | rw | | | | | | rw | | | rw | rw | | rw | rw | rw |

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

Bit 14 **LPR**: Low-power run

When this bit is set, the regulator is switched from main mode (MR) to low-power mode (LPR).

*Note: Stop2 mode cannot be entered when LPR bit is set. Stop1 is entered instead.*

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **DBP**: Disable backup domain write protection

In reset state, the RTC and backup registers are protected against parasitic write access. This bit must be set to enable write access to these registers.

0: Access to RTC and Backup registers disabled
1: Access to RTC and Backup registers enabled

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

Bit 5 **FPDS**: Flash memory power down mode during LPSleep for CPU1

This bit selects whether the flash memory is in power down mode or idle mode when both CPUs are in Sleep mode. flash memory is set in power down mode only when the system is in LPSleep mode and the PWR_C2CR1.FPDS bit from CPU2 also allows this.

0: Flash memory in Idle mode when system is in LPSleep mode
1: Flash memory in power down mode when system is in LPSleep mode

Bit 4 **FPDR**: Flash memory power down mode during LPRun for CPU1

This bit can only be written to 1 after unlocking this register bit, by first writing (code 0xC1B0) into this register (when writing the code, the register bits are not updated). Selects whether the flash memory is in power down mode or idle mode when in LPRun mode. (flash memory can only be in power down mode when code is executed from SRAM). Flash memory is set in power down mode only when the system is in LPRun mode, and the PWR_C2CR1.FPDR bit from CPU2 too allows so.

0: Flash memory in Idle mode when system is in LPRun mode
1: Flash memory in power down mode when system is in LPRun mode

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LPMS[2:0]**: Low-power mode selection for CPU1

These bits are not reset when exiting Standby mode.

These bits select the low-power mode allowed when CPU1 enters the deepsleep mode. The entered system low-power mode depends also upon the PWR_C2CR1.LPMS allowed low-power mode from CPU2.

000: Stop0 mode
001: Stop1 mode
010: Stop2 mode
011: Standby mode
1xx: Shutdown mode

*Note: If LPR bit is set, Stop2 mode cannot be selected and Stop1 mode must be entered instead of Stop2.*
*In Standby mode, SRAM2 can be preserved or not, depending on RRS bit configuration in PWR_CR3.*

## 6.6.2 PWR control register 2 (PWR_CR2)

Address offset: 0x004

Reset value: 0x0000 0000. This register is reset when exiting 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. | PLS[2:0] | | | PVDE |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bits 3:1 **PLS[2:0]**: Programmable voltage detector level selection.

These bits select the voltage threshold detected by the Programmable voltage detector:

000: $V_{PVD0}$ ~ 2.0 V
001: $V_{PVD1}$ ~ 2.2 V
010: $V_{PVD2}$ ~ 2.4 V
011: $V_{PVD3}$ ~ 2.5 V
100: $V_{PVD4}$ ~ 2.6 V
101: $V_{PVD5}$ ~ 2.8 V
110: $V_{PVD6}$ ~ 2.9 V
111: External input analog voltage PVD_IN (compared internally to $V_{DDA}$). The I/O used as PVD_IN input must be configured in analog mode in the GPIO register.

*Note: These bits are write-protected when the bit PVDL (PVD lock) is set in the SYSCFG_CBR register, they are reset only by a system reset.*

Bit 0 **PVDE**: Programmable voltage detector enable

0: Programmable voltage detector disabled
1: Programmable voltage detector enabled

*Note: This bit is write-protected when bit PVDL (PVD lock) is set in the SYSCFG_CBR register. This bit is reset only by a system reset.*

### 6.6.3 PWR control register 3 (PWR_CR3)

Address offset: 0x008

Reset value: 0x0000 8000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EIWUL | EC2H | E802A | EBLEA | ECRPE | APC | RRS | EBORH SMPSFB | Res. | Res. | Res. | Res. | EWUP4 | Res. | Res. | EWUP1 |
| rw | rw | rw | rw | rw | rw | rw | rw | | | | | rw | | | rw |

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

Bit 15 **EIWUL**: Enable internal wake-up line for CPU1
0: Internal wake-up line interrupt to CPU1 disabled
1: Internal wake-up line interrupt to CPU1 enabled

Bit 14 **EC2H**: Enable CPU2 Hold interrupt for CPU1
Enable CPU2 kept in hold, due to C2BOOT, interrupt to CPU1.
0: Interrupt to CPU1 disabled
1: interrupt to CPU1 enabled

Bit 13 **E802A**: Enable 802.15.4 end of activity interrupt for CPU1
0: Interrupt to CPU1 disabled
1: interrupt to CPU1 enabled

Bit 12 **EBLEA**: Enable BLE end of activity interrupt for CPU1
0: Interrupt to CPU1 disabled
1: interrupt to CPU1 enabled

Bit 11 **ECRPE**: Enable critical radio phase end of activity interrupt for CPU1
0: Interrupt to CPU1 disabled
1: interrupt to CPU1 enabled

Bit 10 **APC**: Apply pull-up and pull-down configuration from CPU1
When this bit for CPU1 or the PWR_C2CR3.APC bit for CPU2 is set, the I/O pull-up and pull-down configurations defined in the PWR_PUCRx and PWR_PDCRx registers are applied. When both bits are cleared, the PWR_PUCRx and PWR_PDCRx registers are not applied to the I/Os.

Bit 9 **RRS**: SRAM2a retention in Standby mode
0: SRAM2a powered off in Standby mode (content is lost).
1: SRAM2a powered by the low-power regulator in Standby mode (content is kept).

Bit 8 **EBORHSMPSFB**: Enable BORH interrupts for CPU1
0: Interrupts BORHF to CPU1 disabled
1: interrupts BORHF to CPU1 enabled

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

Bit 3 **EWUP4**: Enable wake-up pin WKUP4 for CPU1

When this bit is set, the external wake-up pin WKUP4 is enabled and triggers an interrupt and wake-up from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP4 bit in the *PWR control register 4 (PWR_CR4)*.

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

Bit 0 **EWUP1**: Enable wake-up pin WKUP1 for CPU1

When this bit is set, the external wake-up pin WKUP1 is enabled and triggers an interrupt and wake-up from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP1 bit in the *PWR control register 4 (PWR_CR4)*.

### 6.6.4 PWR control register 4 (PWR_CR4)

Address offset: 0x00C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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 |
| C2BOOT | Res. | Res. | Res. | Res. | Res. | VBRS | VBE | Res. | Res. | Res. | Res. | WP4 | Res. | Res. | WP1 |
| rw | | | | | | rw | rw | | | | | rw | | | rw |

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

Bit 15 **C2BOOT**: Boot CPU2 after reset or wake-up from Stop or Standby modes.

0: CPU2 does not boot after reset or wake-up from Stop or Standby modes.

1: CPU2 boots after wake-up from Stop or Standby modes.

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

Bit 9 **VBRS**: $V_{BAT}$ battery charging resistor selection

0: Charge $V_{BAT}$ through a 5 kΩ resistor

1: Charge $V_{BAT}$ through a 1.5 kΩ resistor

Bit 8 **VBE**: $V_{BAT}$ battery charging enable

0: $V_{BAT}$ battery charging disabled

1: $V_{BAT}$ battery charging enabled

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

Bit 3 **WP4**: Wakeup pin WKUP4 polarity

This bit defines the polarity used for an event detection on external wake-up pin, WKUP4

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

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

Bit 0 **WP1**: Wakeup pin WKUP1 polarity

This bit defines the polarity used for an event detection on external wake-up pin, WKUP1
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

### 6.6.5 PWR status register 1 (PWR_SR1)

Address offset: 0x010

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Two additional APB cycles are needed to read this register vs. a standard APB read.

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WUFI | C2HF | 802AF | BLE AF | CRPEF | | BLE WUF | BORHF | Res. | Res. | Res. | Res. | WUF4 | Res. | Res. | WUF1 |
| r | r | | r | r | r | r | r | | | | | r | | | r |

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

Bit 15 **WUFI**: Internal wake-up interrupt flag

This bit is set when a wake-up is detected on the internal wake-up line. It is cleared when all internal wake-up sources are cleared.

Bit 14 **C2HF**: CPU2 Hold interrupt flag

This bit is set when a CPU2 wake-up is detected when C2BOOT = 0. It is cleared by PWR_SCR.CC2HF.

Bit 13 **802AF**: 802.15.4 end of activity interrupt flag

This bit is set when a 802.15.4 activity ends. It is cleared by PWR_SCR.C802AF.

Bit 12 **BLEAF**: BLE end of activity interrupt flag

This bit is set when a BLE activity ends. It is cleared by PWR_SCR.CBLEAF.

Bit 11 **CRPEF**: Enable critical radio phase end of activity interrupt flag

This bit is set when Radio phase activity ends. It is cleared by PWR_SCR.CCRPEF.

Bit 10 **802WUF** 802.15.4 wake-up interrupt flag

This bit is set when a wake-up is detected on the 802.15.4 line. It is cleared by PWR_SCR.C802WUF.

Bit 9 **BLEWUF** BLE wake-up interrupt flag

This bit is set when a wake-up is detected on the BLE line. It is cleared by PWR_SCR.CBLEWUF.

Bit 8 **BORHF**: BORH interrupt flag

This bit is set when the $V_{DD}$ rises above the BORH threshold. It is cleared by PWR_SCR.CBORHF.

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

Bit 3 **WUF4**: Wakeup flag 4

This bit is set when a wake-up event is detected on wake-up pin,WKUP4. It is cleared by writing ‘1’ in the CWUF4 bit of the *PWR status clear register (PWR_SCR)*.

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

Bit 0 **WUF1**: Wakeup flag 1

This bit is set when a wake-up event is detected on wake-up pin, WKUP1. It is cleared by writing ‘1’ in the CWUF1 bit of the *PWR status clear register (PWR_SCR)*.

## 6.6.6 PWR status register 2 (PWR_SR2)

Address offset: 0x014

Reset value: 0x0000 0002. This register is partially reset when exiting Standby/Shutdown modes.

| 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. | PVDO | Res. | REGLPF | REGLPS | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | r | | r | r | | | | | | | | |

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **PVDO**: Programmable voltage detector output

0: $V_{DD}$ or voltage level on PVD_IN is above the selected PVD threshold
1: $V_{DD}$ or voltage level on PVD_IN is below the selected PVD threshold

Bit 10 Reserved, must be kept at reset value.

Bit 9 **REGLPF**: Low-power regulator flag

This bit is set by hardware when the MCU is in Low-power run mode. When the MCU exits from the Low-power run mode, this bit remains at 1 until the regulator is ready in main mode. A polling on this bit must be done before increasing the product frequency.
This bit is cleared by hardware when the regulator is ready.
0: The regulator is ready in main mode (MR)
1: The regulator is in low-power mode (LPR)

Bit 8 **REGLPS**: Low-power regulator started

This bit provides the information whether the low-power regulator is ready after a power-on reset or a Standby/Shutdown. If the Standby mode is entered while REGLPS bit is still cleared, the wake-up from Standby mode time may be increased.
0: The low-power regulator is not ready
1: The low-power regulator is ready

Bits 7:0 Reserved, must be kept at reset value.

## 6.6.7 PWR status clear register (PWR_SCR)

Address offset: 0x018

Reset value: 0x0000 0000

Access: Three additional APB cycles are needed to write this register vs. a standard APB write.

| 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. | CC2HF | C802 AF | CFBLE AF | CCRP EF | C802 WUF | CBLE WUF | CBORH F | Res. | Res. | Res. | Res. | CWUF4 | Res. | Res. | CWUF1 |
| | w | w | w | w | w | w | w | | | | | w | | | w |

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

Bit 14 **CC2HF**: Clear CPU2 Hold interrupt flag
Setting this bit clears the C2HF flag in the PWR_SR1. This bit is always read 0.

Bit 13 **C802AF**: Clear 802.15.4 end of activity interrupt flag
Setting this bit clears the 802AF flag in the PWR_SR1. This bit is always read 0.

Bit 12 **CBLEAF**: Clear BLE end of activity interrupt flag
Setting this bit clears the BLEAF flag in the PWR_SR1. This bit is always read 0.

Bit 11 **CCRPEF**: Clear critical radio phase end of activity interrupt flag
Setting this bit clears the CRPEF flag in the PWR_SR1. This bit is always read 0.

Bit 10 **C802WUF** Clear 802.15.4 wake-up interrupt flag
Setting this bit clears the 802WUF flag in the PWR_SR1. This bit is always read 0.

Bit 9 **CBLEWUF** Clear BLE wake-up interrupt flag
Setting this bit clears the BLEWUF flag in the PWR_SR1. This bit is always read 0.

Bit 8 **CBORHF**: Clear BORH interrupt flag
Setting this bit clears the SBORHF flag in the PWR_SR1. This bit is always read 0.

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

Bit 3 **CWUF4**: Clear wake-up flag 4
Setting this bit clears the WUF4 flag in the PWR_SR1 register.

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

Bit 0 **CWUF1**: Clear wake-up flag 1
Setting this bit clears the WUF1 flag in the PWR_SR1 register.

### 6.6.8 PWR Port A pull-up control register (PWR_PUCRA)

Address offset: 0x020

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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 |
| PU15 | PU14 | PU13 | PU12 | PU11 | PU10 | PU9 | PU8 | PU7 | PU6 | PU5 | PU4 | PU3 | PU2 | PU1 | PU0 |
| 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 **PUy**: Port A pull-up bit y (y=0...15)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*. The pull-up is not activated if the corresponding PDy bit is also set.

### 6.6.9 PWR Port A pull-down control register (PWR_PDCRA)

Address offset: 0x024

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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 |
| PD15 | PD14 | PD13 | PD12 | PD11 | PD10 | PD9 | PD8 | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| 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 **PDy**: Port A pull-down bit y (y=0...15)

When set, this bit activates the pull-down on PA[y] when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*.

### 6.6.10 PWR Port B pull-up control register (PWR_PUCRB)

Address offset: 0x028

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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. | PU9 | PU8 | PU7 | PU6 | PU5 | PU4 | PU3 | PU2 | PU1 | PU0 |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 9:0 **PUy**: Port B pull-up bit y (y=0...9)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*. The pull-up is not activated if the corresponding PDy bit is also set.

### 6.6.11 PWR Port B pull-down control register (PWR_PDCRB)

Address offset: 0x02C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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. | PD9 | PD8 | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 9:0 **PDy**: Port B pull-down bit y (y=0...9)

When set, this bit activates the pull-down on PB[y] when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*.

### 6.6.12 PWR Port C pull-up control register (PWR_PUCRC)

Address offset: 0x030

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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 |
| PU15 | PU14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| rw | rw | | | | | | | | | | | | | | |

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

Bits 15:14 **PUy**: Port C pull-up bit y (y=14,15)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*. The pull-up is not activated if the corresponding PDy bit is also set.

Bits 13:0 Reserved, must be kept at reset value.

### 6.6.13 PWR Port C pull-down control register (PWR_PDCRC)

Address offset: 0x034

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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 |
| PD15 | PD14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| rw | rw | | | | | | | | | | | | | | |

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

Bits 15:14 **PDy**: Port C pull-down bit y (y=14,15)

When set, this bit activates the pull-down on PC[y] when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*.

Bits 13:0 Reserved, must be kept at reset value.

### 6.6.14 PWR Port E pull-up control register (PWR_PUCRE)

Address offset: 0x040

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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. | PU4 | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | rw | | | | |

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

Bit 4 **PU4**: Port E pull-up bit 4

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*. The pull-up is not activated if the corresponding PD4 bit is also set.

Bits 3:0 Reserved, must be kept at reset value.

### 6.6.15 PWR Port E pull-down control register (PWR_PDCRE)

Address offset: 0x044

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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. | PD4 | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | rw | | | | |

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

Bit 4 **PD4**: Port E pull-down bit 4

When set, this bit activates the pull-down on PE4 when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*.

Bits 3:0 Reserved, must be kept at reset value.

### 6.6.16 PWR Port H pull-up control register (PWR_PUCRH)

Address offset: 0x058

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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. | PU3 | Res. | Res. | Res. |
| | | | | | | | | | | | | rw | | | |

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

Bit 3 **PU3:** Port H pull-up bit 3

When set, this bit activates the pull-up on PH[y] when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*. The pull-up is not activated if the corresponding PDy bit is also set.

Bits 2:0 Reserved, must be kept at reset value.

### 6.6.17 PWR Port H pull-down control register (PWR_PDCRH)

Address offset: 0x05C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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. | PD3 | Res. | Res. | Res. |
| | | | | | | | | | | | | rw | | | |

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

Bit 3 **PD3:** Port H pull-down bit 3

When set, this bit activates the pull-down on PH[y] when one of the APC bits is set in *PWR control register 3 (PWR_CR3)* and in *PWR CPU2 control register 3 (PWR_C2CR3)*.

Bits 2:0 Reserved, must be kept at reset value.

## 6.6.18 PWR CPU2 control register 1 (PWR_C2CR1)

Address offset: 0x080

Reset value: 0x0000 0000. This register is reset after wake-up from Standby mode, except for bits [2:0].

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 802 EWKUP | BLE EWKUP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | FPDS | FPDR | Res. | LPMS[2:0] | | |
| rw | rw | | | | | | | | | rw | rw | | rw | rw | rw |

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

Bit 15 **802EWKUP**: 802.15.4 external wake-up signal

When set this bit forces a wake-up of the 802.15.4 controller. It is automatically reset when 802.15.4 controller exits its sleep mode.

0: No action.

1: Wakeup 802.15.4 controller form its sleep mode.

Bit 14 **BLEEWKUP**: BLE external wake-up

When set this bit forces a wake-up of the BLE controller. It is automatically reset when BLE controller exits its sleep mode.

0: No action

1: Wakeup BLE controller from its sleep mode

Bits 13:6 Reserved, must be kept at reset value.

Bit 5 **FPDS**: Flash memory power down mode during LPSleep for CPU2

This bit selects whether the flash memory is in power down mode or idle mode when both CPUs are in Sleep mode. flash memory is set in power down mode only when the system is in LPSleep mode and the PWR_CR1.FPDS bit from CPU1 also allows so.

0: Flash memory in Idle mode when system is in LPSleep mode

1: Flash memory in power down mode when system is in LPSleep mode

Bit 4 **FPDR**: Flash memory power down mode during LPRun for CPU2

This bit can only be written to 1 after unlocking this register bit, by first writing (code 0xC1B0) into this register (when writing the code register bits are not updated). Selects whether the flash memory is in power down mode or idle mode when in LPRun mode. (flash memory can only be in power down mode when code is executed from SRAM). Flash memory is set in power down mode only when the system is in LPRun mode, and the PWR_CR1.FPDR bit from CPU1 also allows so.

0: Flash memory in Idle mode when system is in LPRun mode

1: Flash memory in power down mode when system is in LPRun mode

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LPMS[2:0]**: Low-power mode selection for CPU2

These bits are not reset when exiting Standby mode.

These bits select the low-power mode entered when CPU2 enters the deepsleep mode. The system low-power mode entered depend also on the PWR_CR1.LPMS allowed low-power mode from CPU1.

000: Stop0 mode

001: Stop1 mode

010: Stop2 mode

011: Standby mode

1xx: Shutdown mode

*Note: If LPR bit is set, Stop2 mode cannot be selected and Stop1 mode must be entered instead of Stop2.*

*In Standby mode, SRAM2 can be preserved or not, depending on RRS bit configuration in PWR control register 3 (PWR_CR3).*

## 6.6.19 PWR CPU2 control register 3 (PWR_C2CR3)

Address offset: 0x084

Reset value: 0x0000 8000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EI WUL | Res. | Res. | APC | Res. | E802 WUP | EBLE WUP | Res. | Res. | Res. | Res. | Res. | EWUP4 | Res. | Res. | EWUP1 |
| rw | | | rw | | rw | rw | | | | | | rw | | | rw |

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

Bit 15 **EIWUL**: Enable internal wake-up line for CPU2

0: Internal wake-up line to CPU2 disabled

1: Internal wake-up line to CPU2 enabled

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **APC**: Apply pull-up and pull-down configuration for CPU2

When this bit for CPU2 or the PWR_CR3.APC bit for CPU1 is set, the I/O pull-up and pull-down configurations defined in the PWR_PUCRx and PWR_PDCRx registers are applied. When both bits are cleared, the PWR_PUCRx and PWR_PDCRx registers are not applied to the I/Os.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **E802WUP** Enable 802.15.4 host wake-up interrupt for CPU2

0: Interrupt to CPU2 disabled
1: interrupt to CPU2 enabled

Bit 9 **EBLEWUP** Enable BLE host wake-up interrupt for CPU2

0: Interrupt to CPU2 disabled
1: interrupt to CPU2 enabled

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

Bit 3 **EWUP4**: Enable wake-up pin WKUP4 for CPU2

When this bit is set, the external wake-up pin WKUP4 is enabled and triggers an interrupt and wake-up from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU2. The active edge is configured via the WP4 bit in the *PWR control register 4 (PWR_CR4)*.

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

Bit 0 **EWUP1**: Enable wake-up pin WKUP1 for CPU2

When this bit is set, the external wake-up pin WKUP1 is enabled and triggers an interrupt and wake-up from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU2. The active edge is configured via the WP1 bit in the *PWR control register 4 (PWR_CR4)*.

### 6.6.20 PWR extended status and status clear register (PWR_EXTSCR)

Address offset: 0x088

Reset value: 0x0000 0000

Access: Three additional APB cycles are needed to write this register vs. a standard APB write.

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C2DS | C1DS | CRPF | Res. | C2STOPF | C2SBF | C1STOPF | C1SBF | Res. | Res. | Res. | Res. | Res. | CCRPF | C2CSSF | C1CSSF |
| r | r | r | | r | r | r | r | | | | | | w | w | w |

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

Bit 15 **C2DS**: CPU2 deepsleep mode

This bit is set by hardware when CPU2 enters deepsleep mode or is hold by C2BOOT.
0: CPU2 is running or in sleep
1: CPU2 is in deepsleep or hold by C2BOOT

Bit 14 **C1DS**: CPU1 deepsleep mode
This bit is set by hardware when CPU1 enters deepsleep mode.
0: CPU1 is running or in sleep
1: CPU1 is in deepsleep

Bit 13 **CRPF**: Critical Radio system phase
This bit is set by hardware when the Radio system wakes up. It is reset either by hardware when the Radio system enters Low-power mode or by software when writing CCRPF.
0: Not a critical Radio system phase
1: Critical Radio system phase ongoing

Bit 12 Reserved, must be kept at reset value.

Bit 11 **C2STOPF**: System Stop flag for CPU2.
This bit is set by hardware and cleared only by any reset or by setting C2CSSF bit.
0: System has not been in Stop mode
1: System has been in Stop mode

Bit 10 **C2SBF**: System Standby flag for CPU2.
This bit is set by hardware and cleared only by a POR reset or by setting C2CSSF bit.
0: System has not been in Standby mode
1: System has been in Standby mode

Bit 9 **C1STOPF**: System Stop flag for CPU1.
This bit is set by hardware and cleared only by any reset or by setting C1CSSF bit.
0: System has not been in Stop mode
1: System has been in Stop mode

Bit 8 **C1SBF**: System Standby flag for CPU1.
This bit is set by hardware and cleared only by a POR reset or by setting C1CSSF bit.
0: System has not been in Standby mode
1: System has been in Standby mode

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **CCRPF**: Clear critical Radio system phase
Setting this bit clears the CRPF bit.

Bit 1 **C2CSSF**: Clear CPU2 Stop Standby flags
Setting this bit clears the C2STOPF and C2SBF bits.

Bit 0 **C1CSSF**: Clear CPU1 Stop Standby flags
Setting this bit clears the C1STOPF and C1SBF bits.

## 6.6.21 PWR register map and reset value table

**Table 33. PWR 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 | PWR_CR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LPR | Res. | Res. | Res. | Res. | Res. | DBP | Res. | Res. | FPDS | FPDR | Res. | LPMS [2:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | 0 | | | | | | 0 | | | 0 | 0 | | 0 | 0 | 0 |
| 0x004 | PWR_CR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PLS [2:0] | | | PVDE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x008 | PWR_CR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EIWUL | EC2H | E802A | EBLEA | ECRPE | APC | RRS | Res. | Res. | Res. | Res. | Res. | EWUP4 | Res. | Res. | EWUP1 |
| | Reset value | | | | | | | | | | | | | | | | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | 0 | | | 0 |
| 0x00C | PWR_CR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | C2BOOT | Res. | Res. | Res. | Res. | Res. | VBRS | VBE | Res. | Res. | Res. | Res. | WP4 | Res. | Res. | WP1 |
| | Reset value | | | | | | | | | | | | | | | | | 0 | | | | | | 0 | 0 | | | | | 0 | | | 0 |
| 0x010 | PWR_SR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WUFI | C2HF | 802AF | BLEAF | CRPEF | 802WUF | BLEWUF | BORHF | Res. | Res. | Res. | Res. | WUF4 | Res. | Res. | WUF1 |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | 0 | | | 0 |
| 0x014 | PWR_SR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PVDO | Res. | REGLPF | REGLPS | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | 0 | | 0 | 0 | | | | | | | | |
| 0x018 | PWR_SCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CC2HF | C802AF | CBLEAF | CCRPEF | C802WUF | CBLEWUF | CBORHF | Res. | Res. | Res. | Res. | CWUF4 | Res. | Res. | CWUF1 |
| | Reset value | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | 0 | | | 0 |
| 0x01C | 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. |
| 0x020 | PWR_PUCRA | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PU15 | PU14 | PU13 | PU12 | PU11 | PU10 | PU9 | PU8 | PU7 | PU6 | PU5 | PU4 | PU3 | PU2 | PU1 | PU0 |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x024 | PWR_PDCRA | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PD15 | PD14 | PD13 | PD12 | PD11 | PD10 | PD9 | PD8 | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x028 | PWR_PUCRB | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PU9 | PU8 | PU7 | PU6 | PU5 | PU4 | PU3 | PU2 | PU1 | PU0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x02C | PWR_PDCRB | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PD9 | PD8 | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x030 | PWR_PUCRC | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PU15 | PU14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | | | | | | | | |
| 0x034 | PWR_PDCRC | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PD15 | PD14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | | | | | | | | |
| 0x038 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x03C | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x040 | PWR_PUCRE | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PU4 | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | |

**Table 33. PWR 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 | PWR_PDCRE | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PD4 | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | |
| 0x058 | PWR_PUCRH | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PU3 | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | |
| 0x05C | PWR_PDCRH | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PD3 | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | |
| 0x080 | PWR_C2CR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 802WKUP | BLEEWKUP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | FPDS | FPDR | Res. | LPMS [2:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | | | 0 | 0 | | 0 | 0 | 0 |
| 0x084 | PWR_C2CR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EIWUL | Res. | Res. | APC | Res. | E802IWUP | EBLEWUP | Res. | Res. | Res. | Res. | Res. | EWUP4 | Res. | Res. | EWUP1 |
| | Reset value | | | | | | | | | | | | | | | | | 1 | | | 0 | | 0 | 0 | | | | | | 0 | | | 0 |
| 0x088 | PWR_EXTSCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | C2DS | C1DS | CRPF | Res. | C2STOPF | C2SBF | C1STOPF | C1SBF | Res. | Res. | Res. | Res. | Res. | CCRPF | C2CSSF | C1CSSF |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | | | | | | 0 | 0 | 0 |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 7 Reset and clock control (RCC)

## 7.1 Reset

There are three types of reset, namely:

1. a system reset
2. a power reset
3. a backup domain reset

### 7.1.1 Power reset

A power reset is generated when one of the following events occurs:

1. a brown-out reset (BOR)
2. when exiting from Standby mode
3. when exiting from Shutdown mode

A brown-out reset, including power-on or power-down reset (POR/PDR), sets all registers to their reset values except the Backup domain.

When exiting Standby mode, all registers in the $V_{CORE}$ domain are set to their reset value. Registers outside the $V_{CORE}$ domain (RTC, WKUP, IWDG, and Standby/Shutdown modes control) are not impacted.

When exiting Shutdown mode, a Brown-out reset is generated, resetting all registers except those in the Backup domain.

### 7.1.2 System reset

A system reset sets all registers to their reset values, unless specified otherwise in the register description.

A system reset is generated when one of the following events occurs:

1. A low level on the NRST pin (external reset)
2. Window watchdog event (WWDG reset)
3. Independent watchdog event (IWDG reset)
4. A software (SW) reset (see *Software reset*)
5. Low-power mode security reset (see *Low-power mode security reset*)
6. Option byte loader reset (see *Option byte loader reset*)
7. A brown-out reset

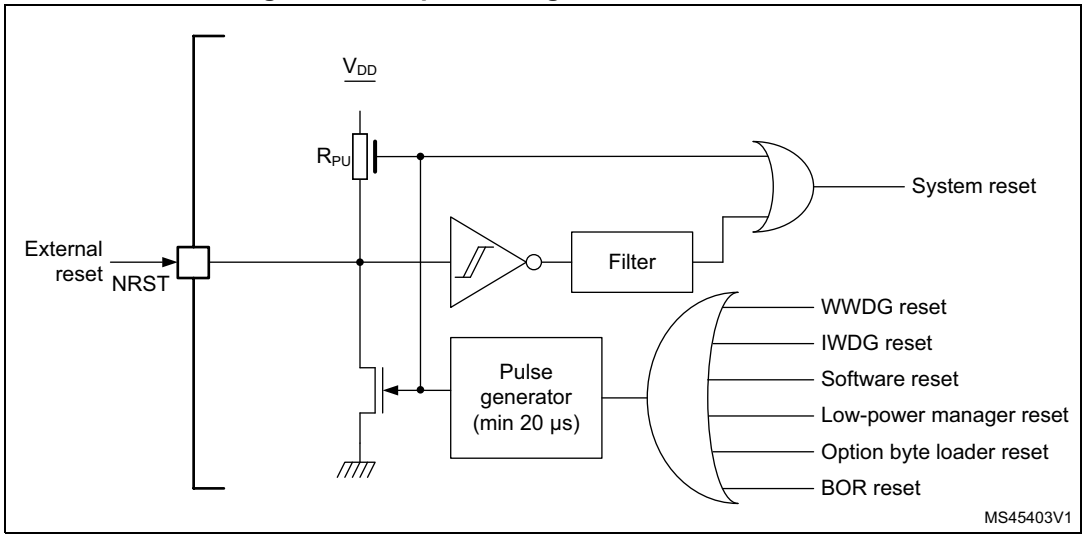
The reset source can be identified by checking the reset flags in the *RCC control/status register (RCC_CSR)*.

These sources act on the NRST pin, always kept low during the delay phase. The CPU1 RESET service routine vector is selected via the BOOT0 and BOOT1.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

In case of an internal reset, the internal pull-up $R_{PU}$ is deactivated to reduce power consumption through the pull-up resistor.

**Figure 13. Simplified diagram of the reset circuit**




### Software reset

The SYSRESETREQ bit in CPU1 application interrupt and reset control register may be set to force a software reset on the device (refer to PM0214 "*STM32 Cortex®-M4 MCUs and MPUs programming manual*", available on *www.st.com*).

The SYSRESETREQ bit in CPU2 application interrupt and reset control register may be set to force a software reset on the device.

### Low-power mode security reset

To prevent that critical applications enter a low-power mode by mistake, two low-power mode security resets are available. If enabled in option bytes, the resets are generated in the following conditions:

1. When entering Standby mode: reset is enabled by resetting nRST_STDBY bit in User option Bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
2. When entering Stop mode: reset is enabled by resetting nRST_STOP bit in User option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.
3. When entering Shutdown mode: reset is enabled by resetting nRST_SHDW bit in User option bytes. In this case, whenever a Shutdown mode entry sequence is successfully executed, the device is reset instead of entering Shutdown mode.

For further information on the User option bytes refer to *Section 3.4.1: Option bytes description*.

### Option byte loader reset

The option byte loader reset is generated when the OBL_LAUNCH bit is set in the FLASH_CR register. This bit is used to launch the option byte loading by software.

### 7.1.3 Backup domain reset

The backup domain has two specific resets.

A backup domain reset is generated when one of the following events occurs:

1. Software reset, triggered by setting the BDRST bit in the *RCC backup domain control register (RCC_BDCR)*.
2. $V_{DD}$ or $V_{BAT}$ power on, if both supplies have previously been powered off.

A backup domain reset only affects the LSE oscillator, the RTC, the Backup registers and the RCC Backup domain control register.

## 7.2 Clocks

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI16 (high speed internal)16 MHz RC oscillator clock
- MSI (multispeed internal) RC oscillator clock from 100 kHz to 48 MHz.
- HSE 32 MHz oscillator clock
- PLL clock

The MSI is used as system clock source after startup from Reset, configured at 4 MHz.

The devices have the following additional clock sources:

- LSI1: 32 kHz low speed internal RC, which may drive the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes (shall not be used for RF system Auto-wakeup).
- LSI2: 32 kHz low speed low drift internal RC, which may drive the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes.
- LSE: 32.768 kHz low speed external crystal, which optionally drives the RTC used for Auto-wakeup or the RF system Auto-wakeup from Stop and Standby modes, or the real-time clock (RTCCLK).
- HSI48: RC 48 MHz internal clock sources to potentially drive the true RNG.

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Several prescalers can be used to configure the AHB frequencies (HCLK1, HCLK2 and HCLK4) the high speed APB (PCLK2) and the low speed APB (PCLK1) domains. The maximum frequency of the AHB (HCLK1 and HCLK4), and of the PCLK1 and PCLK2 domains is 64 MHz. The maximum frequency of the AHB (HCLK2) domain is 32 MHz.

Most peripheral clocks are derived from their bus clock (HCLK, PCLK) except:

- The 48 MHz clock, used for true RNG. This clock is derived (selected by software) from one of the following sources:
  - PLL VCO (PLLQCLK only available in Run mode)
  - MSI clock (only available in Run mode)
  - HSI48 internal oscillator (only available in Run mode)
- The ADCs clock, which is derived (selected by software) from one of the three following sources:
  - system clock (SYSCLK only available in Run mode)
  - PLL VCO (PLLAPCLK only available in Run mode)
- The U(S)ARTs clocks, which are derived (selected by software) from one of the four following sources:
  - system clock (SYSCLK only available in Run mode)
  - HSI16 clock (only available in Run and Stop modes)
  - LSE clock ( only available in Run and Stop modes)
  - APB clock (PCLK depending on which APB is mapped the U(S)ART only available in CRun when enabled in U(S)ARTxEN and CSleep when also enabled in U(S)ARTxSMEN)

  The wakeup from Stop mode is supported only when the clock is HSI16 or LSE.
- The $I^2C$s clocks, derived (selected by software) from one of the three following sources:
  - system clock (SYSCLK only available in Run mode)
  - HSI16 clock (only available in Run and Stop modes)
  - APB clock (PCLK depending on which APB is mapped the $I^2C$ only available in CRun when enabled in I2CxEN and CSleep when also enabled in I2CxSMEN)

  The wakeup from Stop mode is supported only when the clock is HSI.
- The low-power timer (LPTIMx) clocks, which are derived (selected by software) from one of the five following sources:
  - LSI clock (LSI1 or LSI2 only available in Run and Stop modes)
  - LSE clock (only available in Run and Stop modes)
  - HSI16 clock (only available in Run mode)
  - APB clock (PCLK depending on which APB is mapped the LPTIMx only available in CRun when enabled in LPTIMxEN and CSleep when also enabled in LPTIMxSMEN)
  - External clock mapped on LPTIMx_IN1 (only available in Run and Stop modes)

  The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE, or in external clock mode.
- The RTC clock, which is derived (selected by software) from one of the three following sources:
  - LSE clock
  - LSI clock (LSI1 or LSI2)
  - HSE clock divided by 32

  The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE.

- The IWDG clock, which is always the LSI clock (LSI1 or LSI2).
- The RF system wakeup clock, which is derived (selected by software) from one of the two following sources:
  - LSE clock
  - HSE clock divided by 1024

  The functionality in Stop mode (including wakeup) is supported only when the clock is LSE. When HSE/1024 is selected as RF system wakeup source HSEON bit must be set by the application.
- The RF system clock is derived (selected by hardware) from one of the two following sources:
  - HSI16 clock
  - HSE clock

  The functionality in Stop mode is supported only when the clock is HSI16 is selected as wakeup clock by STOPWUCK.

The RCC feeds the CPU1 system timer (SysTick) external clock with the AHB clock (HCLK1) divided by 8. The SysTick can work either with this clock or directly with the CPU1 clock (HCLK1), configurable in the SysTick Control and status register.

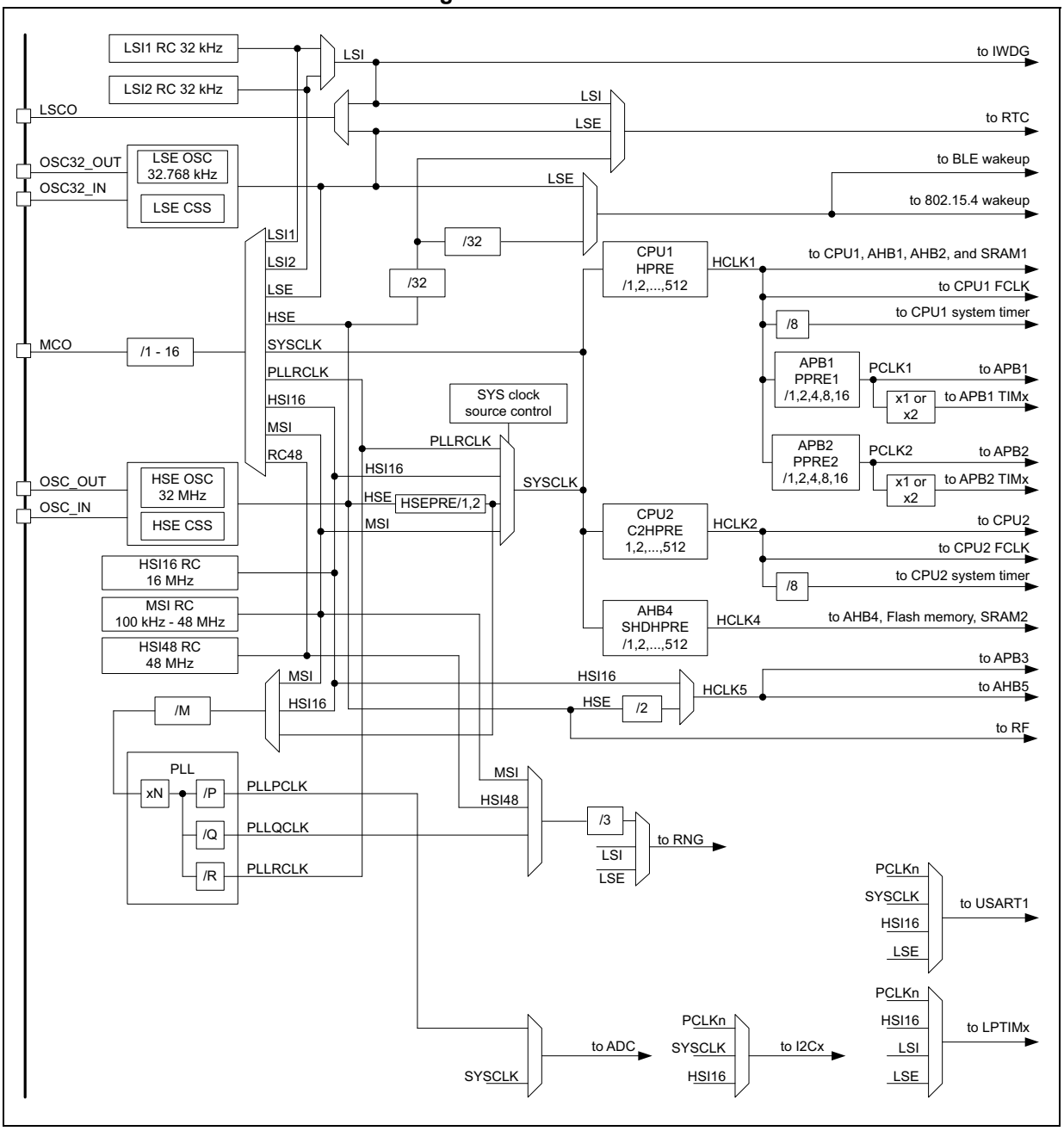
FCLK1 acts as CPU1 free-running clock. For more details refer to PM0214.

The RCC feeds the CPU2 system timer (SysTick) external clock with the AHB clock (HCLK2) divided by 8. The SysTick can work either with this clock or directly with the CPU2 clock (HCLK2), configurable in the SysTick Control and status Register.

FCLK2 acts as CPU2 free-running clock.

The clock tree is detailed in *Figure 14*.

**Figure 14. Clock tree**




2. For full details about the internal and external clock source characteristics refer to the “Electrical characteristics” section in the device datasheet.
3. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is 1, the AHB prescaler must be equal to 1.

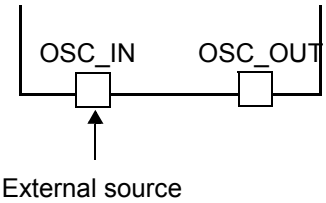
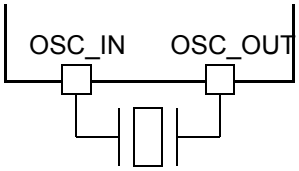
### 7.2.1 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources (see *Figure 15*):

- HSE external crystal
- HSE user external clock

The crystal has to be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time. The loading capacitance are integrated and can be adjusted according to *Frequency tuning*.

**Figure 15. HSE clock sources**

| Clock source | Hardware configuration |
|---|---|
| External clock |   |
| Crystal |   |

### External crystal (HSE crystal)

The 32 MHz external oscillator has the advantage of producing a very accurate rate on the main clock, and is mandatory for any Radio operation.

The associated hardware configuration is shown in *Figure 15*. Refer to the electrical characteristics section of the *datasheet* for more details.

The HSERDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSE oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

The HSE Crystal can be switched on and off using the HSEON bit in the *RCC clock control register (RCC_CR)*.

### External source

In this mode, selectable by setting the HSEON bit in the *RCC clock control register (RCC_CR)*, an external clock source must be provided, with a frequency of 32 MHz. The external clock signal (sinus) with ~45 to 55 % duty cycle (refer to the datasheet) has to drive the OSC_IN pin, while the OSC_OUT pin must be left not connected (see *Figure 15*).

*Note: For details on pin availability, refer to the pinout section in the corresponding device datasheet.*

### Frequency tuning

The HSE oscillator frequency can vary from one chip to another due to manufacturing process variations and used crystal. User can tune the HSE frequency in the application by

writing the HSETUNE bits in *RCC clock HSE register (RCC_HSECR)*. The HSE frequency can be measured by outputting the HSE clock on the MCO.

HSE oscillator gain and sense can be controlled by HSEGMC and HSES bits in *RCC clock HSE register (RCC_HSECR)*. Refer to AN5042 for the HSE trimming procedure.

### 7.2.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz oscillator.

The HSI16 oscillator has the advantage of providing a clock source at low cost. It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

The HSI16 clock can be selected as system clock after wakeup from Stop modes (Stop0, Stop1 or Stop2), see *Section 7.3: Low-power modes*. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails, see *Section 7.2.11*.

When the RF system is enabled the HSI16 must be selected as system clock after wakeup from Stop modes.

The HSI16 clock is used as system clock after restart wakeup from Standby.

#### Calibration

RC oscillator frequencies can vary from one chip to another because of manufacturing process variations, this is why each device is factory calibrated by ST for 1 % accuracy at $T_A$ = 25 °C.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the *RCC internal clock sources calibration register (RCC_ICSCR)*.

The RC oscillator speed can be affected by voltage or temperature variations, the user can trim the HSI16 frequency in the application using the HSITRIM bits in the *RCC internal clock sources calibration register (RCC_ICSCR)*.

For more details on how to measure the HSI16 frequency variation, refer to *Section 7.2.20*.

The HSIRDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSI16 RC is stable or not. At startup, the HSI16 RC output clock is not released until this bit is set by hardware.

The HSI16 RC can be switched on and off using the HSION bit in the *RCC clock control register (RCC_CR)*.

The HSI16 signal can also be used as a backup source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 7.2.11*.

### 7.2.3 MSI clock

The MSI clock signal is generated from an internal RC oscillator. Its frequency range can be adjusted by software by using the MSIRANGE[3:0] bits in the *RCC clock control register (RCC_CR)*. Twelve frequency ranges are available: 100 kHz, 200 kHz, 400 kHz, 800 kHz, 1 MHz, 2 MHz, 4 MHz (default value), 8 MHz, 16 MHz, 24 MHz, 32 MHz and 48 MHz.

The MSI clock is used as system clock after restart from Reset, wakeup from Shutdown low-power modes. After restart from Reset, the MSI frequency is set to its default value 4 MHz. Refer to *Section 7.3: Low-power modes*.

The MSI clock can be selected as system clock after a wakeup from Stop mode (Stop 0, Stop 1 or Stop 2). Refer to *Section 7.3: Low-power modes*. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 7.2.11*.

In addition, when used in PLL-mode with the LSE, it provides a very accurate clock source which can be used by the PLL to run the system at the maximum speed 64 MHz.

The MSIRDY flag in the *RCC clock control register (RCC_CR)* indicates if the MSI RC is stable or not. At startup, the MSI RC output clock is not released until this bit is set by hardware. The MSI RC can be switched on and off by using the MSION bit in the *RCC clock control register (RCC_CR)*.

#### Hardware auto calibration with LSE (PLL-mode)

When a 32.768 kHz external oscillator is present in the application, it is possible to configure the MSI in a PLL-mode by setting the MSIPLLEN bit in the *RCC clock control register (RCC_CR)*. When configured in PLL-mode, the MSI automatically calibrates itself thanks to the LSE. This mode is available for all MSI frequency ranges.

#### Software calibration

The MSI RC oscillator frequency can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1 % accuracy at 25 °C ambient temperature ($T_A$). After reset, the factory calibration value is loaded in the MSICAL[7:0] bits in the *RCC internal clock sources calibration register (RCC_ICSCR)*. If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. The MSI frequency in the application can be trimmed by using the MSITRIM[7:0] bits in the RCC_ICSCR register. For more details on how to measure the MSI frequency variation refer to *Section 7.2.20*.

### 7.2.4 HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator and can be used for the random number generator (true RNG).

The HSI48RDY flag in the Clock recovery register (RCC_CRRCR) indicates whether the HSI48 RC oscillator is stable or not. At startup, the HSI48 RC oscillator output clock is not released until this bit is set by hardware.

The HSI48 can be switched on and off using the HSI48ON bit in the Clock recovery register (RCC_CRRCR).

### 7.2.5 PLL

The device embeds a PLL providing three independent outputs. The internal PLL can be used to multiply the HSI, HSE or MSI output clock frequency. The PLL input frequency must be between 2.66 and 16 MHz. The selected clock source is divided by a programmable factor PLLM from 1 to 8 to provide a clock frequency in the requested input range. Refer to *Figure 14: Clock tree* and *RCC PLL configuration register (RCC_PLLCFGR)*.

The PLL configuration (selection of the input clock and multiplication factor) must be done before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLON to 0 in *RCC clock control register (RCC_CR)*.
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, PLLREN in *RCC PLL configuration register (RCC_PLLCFGR)*.

An interrupt can be generated when the PLL is ready, if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

The PLL output frequency must not exceed 64 MHz.

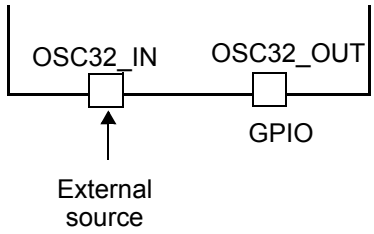
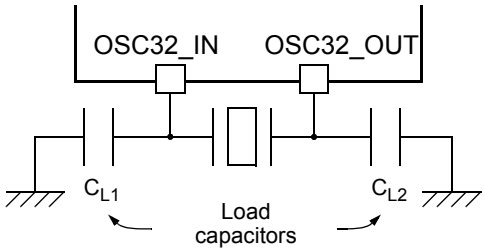
The enable bit of the PLL output clock (PLLPEN, PLLQEN, PLLREN) can be modified at any time without stopping the corresponding PLL. PLLREN cannot be cleared if PLLRCLK is used as system clock.

### 7.2.6 LSE clock

The LSE crystal is a 32.768 kHz low-speed external crystal or ceramic resonator. It has the advantage of providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

**Figure 16. LSE clock sources**

| Clock source | Hardware configuration |
|---|---|
| External clock |  |
| Crystal / ceramic resonators |  |

The LSE crystal is switched on and off using the LSEON bit in *RCC backup domain control register (RCC_BDCR)*. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the *RCC backup domain control register (RCC_BDCR)* to

obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other side. The LSE drive can be decreased to the lower drive capability (LSEDRV = 00) when the LSE is ON. However, once LSEDRV is selected, the drive capability cannot be increased if LSEON = 1.

The LSERDY flag in the *RCC backup domain control register (RCC_BDCR)* indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

#### External source (LSE bypass)

In this mode, selectable by setting the LSEBYP and LSEON bits in the *RCC AHB1 peripheral clocks enable in Sleep modes register (RCC_AHB1SMENR)* an external clock source must be provided, with a frequency of up to 1 MHz. The external clock signal (square, sinus or triangle) with ~50 % duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO, see *Figure 16*.

### 7.2.7 LSI1 clock

The LSI1 RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG), RTC and RF wakeup. The clock frequency is 32 kHz. For more details refer to the electrical characteristics section of the datasheet.

The LSI1 RC can be switched on and off using the LSI1ON bit in the *RCC control/status register (RCC_CSR)*.

The LSI1RDY flag in the *RCC control/status register (RCC_CSR)* indicates if the LSI1 oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

### 7.2.8 LSI2 clock

The LSI2 RC acts as a low drift low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG), RTC wakeup. The clock frequency is ~ 32 kHz. For more details refer to the electrical characteristics section of the datasheet.

The LSI2 RC can be switched on and off using the LSI2ON bit in the *RCC control/status register (RCC_CSR)*.

The LSI2RDY flag in the *RCC control/status register (RCC_CSR)* indicates if the LSI2 oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

After any system reset NRST and before using the LSI2, its oscillator must get the trimming information (LSI2TRIM field in register RCC_SCR).

### 7.2.9 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

- MSI oscillator
- HSI16 oscillator
- HSE oscillator
- PLLRCLK

The system clock maximum frequency is 64 MHz. After a system reset, the MSI oscillator, at 4 MHz, is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch will occur when the clock source becomes ready. Status bits in the *RCC internal clock sources calibration register (RCC_ICSCR)* indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

When waking up from Standby mode the HSI16 is selected as system clock.

### 7.2.10 Clock source frequency

*Table 34* gives the different clock source frequencies.

**Table 34. Maximum clock source frequency**

| MSI | HSI | HSE | PLL |
|---|---|---|---|
| 48 MHz | 16 MHz | 32 MHz | 64 MHz<br>(VCO max = 344 MHz) |

### 7.2.11 Clock security system (CSS) on HSE

The clock security system can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers (TIM1 and TIM16/17) and an interrupt is generated to inform the software about the failure (clock security system interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the CPU1 and CPU2 NMI (non-maskable interrupt) exception vector.

*Note: Once the HSE CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and a NMI is automatically generated. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the RCC clock interrupt clear register (RCC_CICR).*

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock, and the PLL clock is used as system clock), a detected failure causes a switch of the system clock to the MSI or the HSI16 oscillator depending on the STOPWUCK configuration in the *RCC clock configuration register (RCC_CFGR)*, and the disabling of the HSE oscillator. If the HSE clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

### 7.2.12 Clock security system on LSE (LSECSS)

The clock security system on LSE can be activated by software writing the LSECSSON bit in the *RCC control/status register (RCC_CSR)*. This bit can be disabled only by a hardware reset or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE and LSI are enabled (LSEON and LSI1ON enabled) and ready (LSERDY and LSIRDY set by hardware), and after the RTC clock has been selected by RTCSEL.

The CSS on LSE is working in all modes except VBAT. It is working also under system reset (excluding power on reset). If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC but no hardware action is made to the registers. If the MSI was in PLL-mode, this mode is disabled.

In Standby mode a wakeup is generated. In other modes an interrupt can be sent to wakeup the software (see *RCC clock interrupt enable register (RCC_CIER)*, *RCC clock interrupt flag register (RCC_CIFR)*, *RCC clock interrupt clear register (RCC_CICR)*).

The software MUST then disable the LSECSSON bit, stop the defective 32 kHz oscillator (disabling LSEON), and change the RTC clock source (no clock or LSI or HSE, with RTCSEL), or take any required action to secure the application.

### 7.2.13 LSI source selection

The LSI used in the system can be selected to come either from LSI1 or LSI2. Whenever LSI2 is turned on by LSI2ON register bit, the LSI2 (when ready) is selected as LSI source.

To switch from LSI2 to LSI1 without interrupting the LSI clock, the LSI1 must first be switched on by LSI1ON register bit. The FW must verify that the LSI1 is ready by LSI1RDY register bit before disabling LSI2 in LSI2ON register bit.

### 7.2.14 ADC clock

The ADC clock is derived from the system clock, or from the PLL output. It can reach 64 MHz and can be divided by the prescaler values 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128 or 256 by configuring the ADC1_CCR register. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bit fields in the ADC1_CCR.

If the programmed factor is ‘1’, the AHB prescaler must be set to ‘1’.

### 7.2.15 RTC clock

The RTCCLK clock source can be either the HSE/32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the *RCC backup domain control register (RCC_BDCR)*. This selection cannot be modified without resetting the Backup domain. The system must always be configured so as to get a PCLK frequency greater then or equal to the RTCCLK frequency for a proper operation of the RTC.

The LSE clock is in the Backup domain, whereas the HSE and LSI clocks are not. Consequently:

- If LSE is selected as RTC clock the RTC continues to work even if the $V_{DD}$ supply is switched off, provided the $V_{BAT}$ supply is maintained.
- If LSI is selected as the RTC clock the RTC state is not guaranteed if the $V_{DD}$ supply is powered off.
- If the HSE clock divided by a prescaler is used as the RTC clock the RTC state is not guaranteed if the $V_{DD}$ supply is powered off or if the internal voltage regulator is powered off (removing power from the $V_{CORE}$ domain).

When the RTC clock is LSE or LSI, the RTC remains clocked and functional under system reset.

### 7.2.16 Timer clock

The timer clock frequencies are automatically defined by hardware. There are two cases:

1. If the APB prescaler equals 1, the timer clock frequencies are set to the same frequency as that of the APB domain.
2. Otherwise, they are set to twice (×2) the frequency of the APB domain.

### 7.2.17 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI clock is forced on.

When neither the LSI1 oscillator nor the LSI2 oscillator is enabled when starting the IWDG, the LSI1 oscillator is forced on. After the LSI1 oscillator temporization, the clock is provided to the IWDG.

When LSI2 is switched on with the LSI2ON bit, the LSI clock is switched from LSI1 to LSI2.

When LSI2 is disabled by LSI2ON (when the IWDG is running), the LSI1 oscillator is forced on. The LSI clock is switched to LSI1 when ready, and only then LSI2 is stopped.

### 7.2.18 True RNG clock

The true random number generator (RNG) seed clock is derived from the MSI, HSI48, PLL LSI or LSE clock.

### 7.2.19 Clock-out capability

- MCO

  The microcontroller clock output (MCO) capability enables the clock to be output on the external MCO pin. One of the following clock signals can be selected as MCO clock:

  – LSI1 (available in Run and Stop modes)
  – LSI2 (available in Run and Stop modes)
  – LSE (available in Run and Stop modes)
  – MSI (available in Run mode)
  – HSI (available in Run mode, when enabled by HSION)
  – HSI48 (available in Run mode)
  – HSE (available in Run mode)
  – PLLRCLK (available in Run mode)
  – SYSCLK (available in Run mode)

  The selection is controlled by the MCOSEL[3:0] bits of the *RCC clock configuration register (RCC_CFGR)*. The selected clock can be divided with the MCOPRE[2:0] field of the *RCC clock configuration register (RCC_CFGR)*.

  The clock on MCO is available only in Run down to Stop1 mode. Low frequency clocks (LSIx, LSE) are available down to Stop1 mode. The clock MCO is not available in Stop2, Standby, and Shutdown modes.

- LSCO

  The LSCO output enables a low speed clock to be output on the external LSCO pin:

  – LSI (LSI1 or LSI2)
  – LSE

  The selection is controlled by the LSCOSEL, and enabled with the LSCOEN in the *RCC backup domain control register (RCC_BDCR)*.

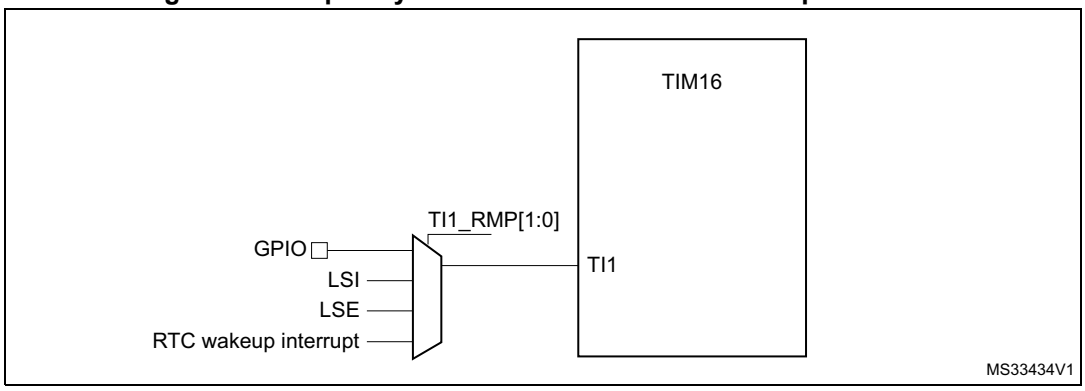
  The clock on LSCO is available in Run, Stop, and on one GPIO in Standby and Shutdown modes.

The configuration registers of the corresponding GPIO port must be programmed in alternate function mode.

### 7.2.20 Internal/external clock measurement with TIM16/TIM17

It is possible to indirectly measure the frequency of all on-board clock sources by mean of the TIM16 or TIM17 channel 1 input capture, as shown on *Figure 17* and *Figure 18*.

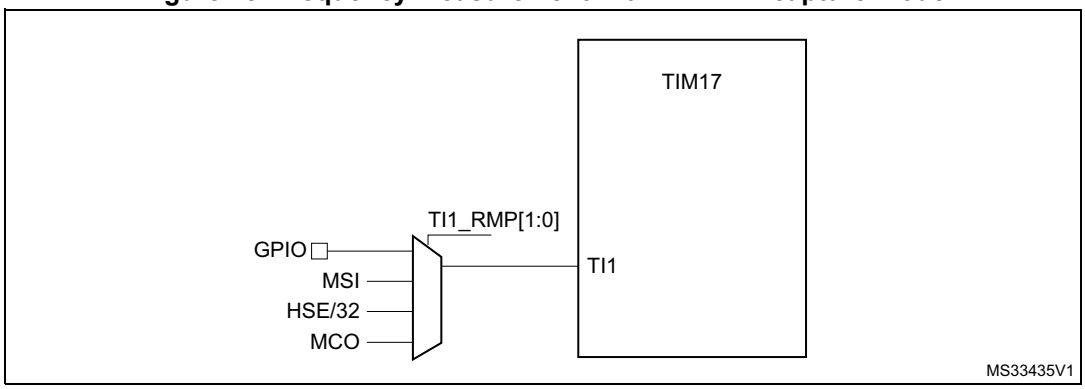
**Figure 17. Frequency measurement with TIM16 in capture mode**




The input capture channel of TIM16 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1_RMP[1:0] bits in the TIM16_OR register. The possibilities are the following ones:

- TIM16 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM16 Channel1 is connected to the LSI clock.
- TIM16 Channel1 is connected to the LSE clock.
- TIM16 Channel1 is connected to the RTC wakeup interrupt signal. In this case the RTC interrupt should be enabled.

**Figure 18. Frequency measurement with TIM17 in capture mode**




The input capture channel of the Timer 17 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1_RMP[1:0] bits in the TIM17_OR register. The possibilities are the following ones:

- TIM17 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheet.
- TIM17 Channel1 is connected to the MSI clock.
- TIM17 Channel1 is connected to the HSE/32 clock.
- TIM17 Channel1 is connected to the microcontroller clock output (MCO), this selection is controlled by the MCOSEL[3:0] bits of the Clock configuration register (RCC_CFGR).

### Calibration of the HSI16 and the MSI

For TIM16, the primary purpose of connecting the LSE to the channel 1 input capture is to be able to precisely measure the HSI16 and MSI system clocks (for this, either the HSI16 or

MSI should be used as the system clock source). The number of HSI16 (MSI, respectively) clock counts between consecutive edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few ppm tens), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process- and/or temperature- and voltage-related frequency deviations.

Both the MSI and HSI16 oscillators have dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (e.g. the HSI/LSE ratio): the precision is therefore closely related to the ratio between the two clock sources (the higher the ratio, the better the measurement).

If LSE is not available, HSE32 is the better option to reach the most precise possible calibration.

It is however impossible to have a good enough resolution when the MSI clock is low (typically below 1 MHz). In this case, it is advised to:

- accumulate the results of several captures in a row
- use the timer input capture prescaler (up to one capture every eight periods)
- use the RTC wakeup interrupt signal (when the RTC is clocked by the LSE) as the input for the channel1 input capture. This improves the measurement precision. For this purpose the RTC wakeup interrupt must be enable.

#### Calibration of the LSI

The calibration of the LSI will follow the same pattern that for the HSI, but changing the reference clock. It is necessary to connect LSI clock to the channel 1 input capture of the TIM16. Then define the HSE as system clock source, the number of his clock counts between consecutive edges of the LSI signal provides a measure of the internal low speed clock period.

The basic concept consists in providing a relative measurement (e.g. the HSE/LSI ratio). The precision is therefore closely related to the ratio between the two clock sources, higher ratios result in more accurate measurements.

### 7.2.21 Peripheral clocks enable

Most peripheral bus and kernel clocks can individually be enabled per CPU. The RCC_AHBxENR, and RCC_APBxENRy enable peripheral clocks for CPU1 and RCC_C2_AHBxENR, and RCC_C2_APBxENR for CPU2. The peripheral clocks will follow the CPU(s) state for which they are enabled, see *Table 35*.

Peripheral bus clock activity during the CPU Sleep mode is controlled by the peripheral clock CPU sleep mode enable bit of the RCC_AHBxSMENR, and RCC_APBxSMENRy for CPU1 and RCC_C2_AHBxSMENR, and RCC_C2_APBxSMENRy for CPU2 registers.The peripheral bus clock during Sleep mode will follow the CPU(s) state for which is it enabled, see *Table 35*.

**Table 35. Peripheral clock enable**

| Peripheral | | | | | |
|---|---|---|---|---|---|
| **xxxEN** | **xxxSMEN** | **CPU mode** | **System mode** | **Bus clock** | **Kernel clock(1)** |
| 0 | x | Any | Any | Stopped | Stopped |
| 1 | | CRun | Run | Clocked | Clocked |
| | 0 | CSleep and CStop | Run | Stopped | Clocked |
| | 1 | CSleep | Run | Clocked | Clocked |
| | | CStop | Run | Stopped | Clocked |
| | | | Stop | Stopped | Clocked when selected from<br>– HSI16<br>– LSI<br>– LSE. |
| | | | | | Stopped when selected from<br>– bus clock<br>– SYSCLK<br>– PLL clocks<br>– MSI<br>– HSI48 |
| | | | Standby or Shutdown | Stopped | Stopped |

1. Only the peripherals I2C, LPTIM, USART, true RNG and ADC have a kernel clock.

When the peripheral bus clock is not active, the peripheral registers read or write accesses are not supported.

When the peripheral bus clock is active the peripheral can be accessed by both CPUs regardless of which CPU has enabled the peripheral bus clock in its CPU xxxEN bit. However, when the peripheral bus clock is enabled by only one CPU, and this CPU enters low power mode, the peripheral bus clock is stopped (also depending on this CPUs xxxSMEN setting) and peripheral access for the other CPU is no longer supported. It is therefor good practice to enable the peripheral bus clock with the CPU dedicated clock enable.

When the peripheral kernel clock is not active, the peripheral functionality is stopped.

The enable bit has a synchronization mechanism to create a glitch free clock for the peripheral. After the enable bit is set, there is a two clock cycles delay before the clock becomes active.

**Caution:** Just after enabling the clock for a peripheral, software must wait for a delay before accessing the peripheral registers.

*Note: The BLE-IP when active, will activate the BLE bus and SRAM2 bus interface clocks.*

## 7.3 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and Low-power sleep modes stop the CPU clock. The memory interface clocks (Flash and SRAM1 and SRAM2 interfaces) can be stopped by software during Sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.
- Stop modes (Stop 0, Stop 1 and Stop 2) stops most clocks in the $V_{CORE}$ domain and disables the PLL, the MSI and the HSE oscillators. The HSI16 may be kept running when requested by the IPs (USART1, I2C1) that make it possible to wakeup from Stop modes.

  All U(S)ARTs and $I^2C$s have the capability to enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as the clock source for that peripheral).

  All U(S)ARTs and LPTIMs can also be driven by the LSE oscillator when the system is in Stop mode (if LSE is selected as clock source for that peripheral) and the LSE oscillator is enabled (LSEON). In that case the LSE remains always ON in Stop mode (they do not have the capability to turn on the LSE oscillator).

  The LPTIMs can also be driven by the LSI oscillator when the system is in Stop mode (if LSI is selected as clock source for that peripheral) and the LSI oscillator is enabled (LSI1ON or LSI2ON).
- Standby and Shutdown modes stops all the clocks in the $V_{CORE}$ domain and disables the PLL, the HSI, the MSI and the HSE oscillators.

The low power modes mode can be overridden for debugging by setting the DBG_SLEEP, DBG_STOP or DBG_STANDBY bits in the DBGMCU_CR register. In addition the EXTI CDBGPWRUPREQ events can be used to allow debugging in Stop modes (see *Table 36*).

**Table 36. Single core Low power debug configurations(1)**

| Mode | CDBGPWRUPREQ | DBGMCU | | | Debug |
|---|---|---|---|---|---|
| | CPU1 | DBG_STANDBY | DBG_STOP | DBG_SLEEP | CPU1 |
| Sleep | X | X | X | X | Enabled |
| Stop0 and Stop1 | Disabled | X | Disabled | - | Disabled |
| | Enabled | | | | Enabled |
| Stop0, Stop1 and Stop2 | X | | Enabled | | Enabled |
| Standby | X | Disabled | - | - | Disabled |
| | | Enabled | | | Enabled |

1. X = Do not use.

When leaving the Stop modes (Stop0, Stop1 or Stop2), the system clock is either MSI or HSI, depending on the software configuration of the STOPWUCK bit in the RCC_CFGR register. When the RF system is enabled the HSI16 shall be selected as in STOPWUCK. When STOPWUCK select the HSI16 clock when leaving Stop mode, the C2HPRE is reset. to select divide by 1. The frequency (range and user trim) of the MSI oscillator is the one configured before entering Stop mode. The user trim of HSI16 is kept. If the MSI was in PLL-mode before entering Stop mode, the PLL-mode stabilization time must be waited for after wakeup even if the LSE was kept ON during the Stop mode.

When leaving the Standby mode, the system clock is HSI.

When leaving the Shutdown modes, the system clock is MSI. The MSI frequency at wakeup from Shutdown mode is 4 MHz. The user trim is lost.

If a Flash memory programming operation is on going, Stop, Standby and Shutdown modes entry is delayed until the Flash memory interface access is finished. If an access to the APB domain is ongoing, Stop, Standby and Shutdown modes entry is delayed until the APB access is finished.

# 7.4 RCC registers

## 7.4.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0061 (after POR reset), 0x0000 0160 (after wakeup from Standby reset)

Access: No wait state, word, half-word and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | PLL RDY | PLL ON | Res. | Res. | Res. | HSEP RE | CSS ON | Res. | HSE RDY | HSE ON |
| | | | | | | r | rw | | | | rw | rs | | r | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | HSIKE RDY | HSI ASFS | HSI RDY | HSIKER ON | HSI ON | MSIRANGE[3:0] | | | | Res. | MSI PLLEN | MSI RDY | MSI ON |
| | | | r | rw | r | rw | rw | rw | rw | rw | rw | | rw | r | rw |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **PLLRDY:** System PLL clock ready flag

Set by hardware to indicate that the system PLL is locked.
0: PLL unlocked
1: PLL locked

Bit 24 **PLLON:** System PLL enable

Set and cleared by software to enable the PLL.
Cleared by hardware when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the PLL clock is used as the system clock.
0: PLL OFF
1: PLL ON

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **HSEPRE:** HSE system clock and PLL M divider prescaler

Set and cleared by software to control the division factor of the system clock and PLL M divider input when selecting HSE clock.
0: SYSCLK and PLL M divider input clocks are not divided (HSE)
1: SYSCLK and PLL M divider input clocks are divided by 2 (HSE/2)

Bit 19 **CSSON:** HSE clock security system enable

Set by software to enable the clock security system. When CSSON is set, the HSE lock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if a HSE clock failure is detected. This bit is set only and is cleared by reset.
0: HSE clock security system OFF (clock detector OFF)
1: HSE clock security system ON (clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 18 Reserved, must be kept at reset value.

Bit 17 **HSERDY:** HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE oscillator not ready

1: HSE oscillator ready

*Note: Once the HSEON bit is cleared, HSERDY goes low after six HSE clock cycles. Regardless of the value of the HSEON bit, the HSE oscillator starts on RF demand when needed. It is possible to have HSEON = 0 and HSERDY = 1.*

Bit 16 **HSEON:** HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF

1: HSE oscillator ON

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

Bit 12 **HSIKERDY**: HSI16 kernel clock ready flag for peripherals requests.

Set by hardware to indicate that HSI16 oscillator is stable when enabled by HSIKERON or a peripheral kernel clock request. Not set when HSI16 is enabled by software by setting HSION, or by wakeup from Standby.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

Bit 11 **HSIASFS:** HSI16 automatic start from Stop

Set and cleared by software. When the system wakeup clock is MSI, this bit is used to wakeup the HSI16 is parallel of the system wakeup.

0: HSI16 oscillator is not enabled by hardware when exiting Stop mode with MSI as wakeup clock.

1: HSI16 oscillator is enabled by hardware when exiting Stop mode with MSI as wakeup clock.

Bit 10 **HSIRDY:** HSI16 clock ready flag. (After wakeup from Standby this bit is read ‘b1 once the HSI16 is ready)

Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION, or by wakeup from Standby. Not set when HSI16 is enabled by HSIKERON or by IP request.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

*Note: Once the HSION bit is cleared, HSIRDY goes low after 6 HSI16 clock cycles.*

Bit 9 **HSIKERON:** HSI16 always enable for peripheral kernel clocks.

Set and cleared by software to force HSI16 ON even in Stop modes. The HSI16 enabled by HSIKERON can only feed USARTs and $I^2C$s peripherals configured with HSI16 as kernel clock. Keeping the HSI16 ON in Stop mode avoids slowing down the communication speed because of the HSI16 startup time. This bit has no effect on HSION value.

0: No effect on HSI16 oscillator.

1: HSI16 oscillator is forced ON even in Stop mode.

Bit 8 **HSION:** HSI16 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSI16 oscillator when entering Stop, Standby or Shutdown mode.

Set by hardware to force the HSI16 oscillator ON when STOPWUCK = 1 or HSIASFS = 1 when leaving Stop modes, or in case of failure of the HSE crystal oscillator.

This bit is set by hardware if the HSI16 is used directly or indirectly as system clock.

0: HSI16 oscillator OFF

1: HSI16 oscillator ON

Bits 7:4 **MSIRANGE[3:0]:** MSI clock ranges

These bits are configured by software to choose the frequency range of MSI when MSIRANGE is set.Twelve frequency ranges are available:

0000: range 0, ~100 kHz

0001: range 1, ~200 kHz

0010: range 2, ~400 kHz

0011: range 3, ~800 kHz

0100: range 4, ~1M Hz

0101: range 5, ~2 MHz

0110: range 6, ~4 MHz (reset value)

0111: range 7, ~8 MHz

1000: range 8, ~16 MHz

1001: range 9, ~24 MHz

1010: range 10, ~32 MHz

1011: range 11, ~48 MHz

others: not allowed (hardware write protection)

*Note: Warning: MSIRANGE can be modified when MSI is OFF (MSION = 0) or when MSI is ready (MSIRDY = 1). MSIRANGE must NOT be modified when MSI is ON and NOT ready (MSION = 1 and MSIRDY = 0).*

Bit 3 Reserved, must be kept at reset value.

Bit 2 **MSIPLLEN:** MSI clock PLL enable

Set and cleared by software to enable/ disable the PLL part of the MSI clock source.

MSIPLLEN must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware).There is a hardware protection to avoid enabling MSIPLLEN if LSE is not ready.

This bit is cleared by hardware when LSE is disabled (LSEON = 0) or when the clock security system on LSE detects a LSE failure (refer to RCC_CSR register).

0: MSI PLL OFF

1: MSI PLL ON

Bit 1 **MSIRDY:** MSI clock ready flag (After reset this bit is read ‘b1 once the MSI is ready)

This bit is set by hardware to indicate that the MSI oscillator is stable.

0: MSI oscillator not ready

1: MSI oscillator ready

*Note: Once the MSION bit is cleared, MSIRDY goes low after 6 MSI clock cycles.*

Bit 0 **MSION:** MSI clock enable

This bit is set and cleared by software.

Cleared by hardware to stop the MSI oscillator when entering Stop, Standby or Shutdown mode.

Set by hardware to force the MSI oscillator ON when exiting Standby or Shutdown mode.

Set by hardware to force the MSI oscillator ON when STOPWUCK = 0 when exiting from Stop modes, or in case of a failure of the HSE oscillator

Set by hardware when used directly or indirectly as system clock.

0: MSI oscillator OFF

1: MSI oscillator ON

## 7.4.2 RCC internal clock sources calibration register (RCC_ICSCR)

Address offset: 0x004

Reset value: 0x40XX 00XX (X is factory-programmed)

Access: No wait state, word, half-word and byte access

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSITRIM[6:0]:** HSI16 clock trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI16.

The default value is 64, which, when added to the HSICAL value, should trim the HSI16 to 16 MHz ± 1 %.

Bits 23:16 **HSICAL[7:0]:** HSI16 clock calibration

These bits are initialized at startup with the factory-programmed HSI16 calibration trim value. When HSITRIM is written, HSICAL is updated with the sum of HSITRIM and the factory trim value.

Bits 15:8 **MSITRIM[7:0]:** MSI clock trimming

These bits provide an additional user-programmable trimming value that is added to the MSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the MSI.

The default value is 0, which, when added to the MSICAL value, should trim the MSI to its mid frequency.

Bits 7:0 **MSICAL[7:0]:** MSI clock calibration

These bits are initialized at startup with the factory-programmed MSI calibration trim value. When MSITRIM is written, MSICAL is updated with the sum of MSITRIM and the factory trim value.

### 7.4.3 RCC clock configuration register (RCC_CFGR)

Address offset: 0x008

Reset value: 0x0007 0000 (after POR reset), 0x0007 0001 (after wakeup from Standby)

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

One or two wait states inserted only if the access occurs during clock source switch.

From 0 to 15 wait states inserted if the access occurs when the APB or AHB prescalers values update is on going.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | MCOPRE[2:0] | | | MCOSEL[3:0] | | | | Res. | Res. | Res. | Res. | Res. | PPRE2F | PPRE1F | HPREF |
| | rw | rw | rw | rw | rw | rw | rw | | | | | | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STOP WUCK | Res. | PPRE2[2:0] | | | PPRE1[2:0] | | | HPRE[3:0] | | | | SWS[1:0] | | SW[1:0] | |
| rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | r | r | rw | rw |

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **MCOPRE[2:0]:** Microcontroller clock output prescaler

These bits are set and cleared by software.

It is highly recommended to change this prescaler before MCO output is enabled.

000: MCO is divided by 1
001: MCO is divided by 2
010: MCO is divided by 4
011: MCO is divided by 8
100: MCO is divided by 16
Others: not allowed

Bits 27:24 **MCOSEL[3:0]:** Microcontroller clock output

Set and cleared by software.

0000: MCO output disabled, no clock on MCO
0001: SYSCLK system clock selected
0010: MSI clock selected.
0011: HSI16 clock selected.
0100: HSE clock selected (after stabilization, after HSERDY = 1)
0101: Main PLLRCLK clock selected
0110: LSI1 clock selected
0111: LSI2 clock selected
1000: LSE clock selected
1001: Internal HSI48 clock selected
1100: HSE clock selected (before stabilization, after HSEON = 1)
Others: Reserved

*Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.*

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **PPRE2F:** PCLK2 prescaler flag (APB2)

Set and reset by hardware to acknowledge PCLK2 prescaler programming

Reset when a new prescaler value is programmed in PPRE2. set when the programmed value is actually applied.

0: PCLK2 prescaler value not yet applied
1: PCLK2 prescaler value applied

Bit 17 **PPRE1F:** PCLK1 prescaler flag (APB1)

Set and reset by hardware to acknowledge PCLK1 prescaler programming

Reset when a new prescaler value is programmed in PPRE1. set when the programmed value is actually applied.

0: PCLK1 prescaler value not yet applied
1: PCLK1 prescaler value applied

Bit 16 **HPREF:** HCLK1 prescaler flag (CPU1, AHB1, AHB2, AHB3 and SRAM1)

Set and reset by hardware to acknowledge HCLK1 prescaler programming

Reset when a new prescaler value is programmed in HPRE. set when the programmed value is actually applied.

0: HCLK1 prescaler value not yet applied
1: HCLK1 prescaler value applied

Bit 15 **STOPWUCK:** Wakeup from Stop and CSS backup clock selection

Set and cleared by software to select the system clock used when exiting Stop mode.

The selected clock is also used as emergency clock for the clock security system on HSE.

Warning: STOPWUCK must not be modified when the HSE clock security system is enabled by CSSON in *RCC clock control register (RCC_CR)* register and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW= 10).

0: MSI oscillator selected as wakeup from stop clock and CSS backup clock.
1: HSI16 oscillator selected as wakeup from stop clock and CSS backup clock

Bit 14 Reserved, must be kept at reset value.

Bits 13:11 **PPRE2[2:0]:** PCLK2 high-speed prescaler (APB2)

Set and cleared by software to control the division factor of the PCLK2 clock (APB2).

The PPRE2F flag can be checked to know if the programmed PPRE2 prescaler value is applied.

0xx: HCLK1 not divided
100: HCLK1 divided by 2
101: HCLK1 divided by 4
110: HCLK1 divided by 8
111: HCLK1 divided by 16

Bits 10:8 **PPRE1[2:0]:** PCLK1 low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the PCLK1 clock (APB1).

The PPRE1F flag can be checked to know if the programmed PPRE1 prescaler value is applied.

0xx: HCLK1 not divided
100: HCLK1 divided by 2
101: HCLK1 divided by 4
110: HCLK1 divided by 8
111: HCLK1 divided by 16

Bits 7:4 **HPRE[3:0]:** HCLK1 prescaler (CPU1, AHB1, AHB2, AHB3 and SRAM1.)

Set and cleared by software to control the division factor of the HCLK1 clock (CPU1, AHB1, AHB2, AHB3 and SRAM1).

The HPREF flag can be checked to know if the programmed HPRE prescaler value is applied.

**Caution:** The software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details refer to *Section 7.2.10: Clock source frequency*). After a write operation to these bits the register bit HPREF must be read to be sure that the new value has been taken into account.

0001: SYSCLK divided by 3
0010: SYSCLK divided by 5
0101: SYSCLK divided by 6
0110: SYSCLK divided by 10
0111: SYSCLK divided by 32
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512
Others: SYSCLK not divided

Bits 3:2 **SWS[1:0]:** System clock switch status

Set and cleared by hardware to indicate which clock source is used as system clock.
00: MSI oscillator used as system clock
01: HSI16 oscillator used as system clock
10: HSE used as system clock
11: PLL used as system clock

Bits 1:0 **SW[1:0]:** System clock switch

Set and cleared by software to select system clock source (SYSCLK).
Configured by HW to force MSI oscillator selection when exiting Shutdown mode.
Configured by HW to force HSI16 oscillator selection when exiting Standby mode.
Configured by HW to force MSI or HSI16 oscillator selection when exiting Stop mode or in case of failure of the HSE oscillator, depending on STOPWUCK value.
00: MSI selected as system clock
01: HSI16 selected as system clock
10: HSE selected as system clock
11: PLL selected as system clock

*Note: The latency when changing SW and SWS bits is set according to the new configuration when switching the SYSCLK clock source, in a maximum of three cycles of the previous clock.*

### 7.4.4 RCC PLL configuration register (RCC_PLLCFGR)

Address offset: 0x00C

Reset value: 0x2204 0100

Access: No wait state, word, half-word and byte access

This register is used to configure the PLL clock outputs according to the following formulas:

- f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)
- f(PLL_P) = f(VCO clock) / PLLP
- f(PLL_Q) = f(VCO clock) / PLLQ
- f(PLL_R) = f(VCO clock) / PLLR

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PLLR[2:0] | | | PLLREN | PLLQ[2:0] | | | PLLQEN | Res. | Res. | PLLP[4:0] | | | | | PLLPEN |
| 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. | PLLN[6:0] | | | | | | | Res. | PLLM[2:0] | | | Res. | Res. | PLLSRC[1:0] | |
| | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | | | rw | rw |

Bits 31:29 **PLLR[2:0]:** Main PLL division factor for PLLRCLK

Set and cleared by software to control the frequency of the main PLL output clock PLLRCLK. This output can be selected as system clock. These bits can be written only if the PLL is disabled.

PLLRCLK output clock frequency = VCO frequency / PLLR with PLLR = 2, 3, 4,... or 8 [VCO frequency / (N + 1)]

000: reserved
001: PLLR = 2
010: PLLR = 3
011: PLLR = 4
100: PLLR = 5
101: PLLR = 6
110: PLLR = 7
111: PLLR = 8

The software has to set these bits so that 64 MHz is not exceeded on this domain.

Bit 28 **PLLREN:** Main PLL PLLRCLK output enable

Set and reset by software to enable the PLLRCLK output of the main PLL (used as system clock).

This bit cannot be written when PLLRCLK output of the PLL is used as System clock.

To save power, when the PLLRCLK output of the PLL is not used, the value of PLLREN should be 0.

0: PLLRCLK output disabled
1: PLLRCLK output enabled

Bits 27:25 **PLLQ[2:0]:** Main PLL division factor for PLLQCLK

Set and cleared by software to control the frequency of the main PLL output clock PLLQCLK. This output can be selected for true RNG clock. These bits can be written only if PLL is disabled.

PLLQCLK output clock frequency = VCO frequency / PLLQ with PLLQ = 2, 3, 4,... or 8 [VCO frequency / (N + 1)]

000: reserved
001: PLLQ = 2
010: PLLQ = 3
011: PLLQ = 4
100: PLLQ = 5
101: PLLQ = 6
110: PLLQ = 7
111: PLLQ = 8

The software has to set these bits so that 64 MHz is not exceeded on this domain.

Bit 24 **PLLQEN:** Main PLL PLLQCLK output enable

Set and reset by software to enable the PLLQCLK output of the main PLL (used as system clock).

In order to save power, when the PLLQCLK output of the PLL is not used, the value of PLLQEN should be 0.

0: PLLQCLK output disable
1: PLLQCLK output enable

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

Bits 21:17 **PLLP[4:0]:** Main PLL division factor for PLLPCLK.

Set and cleared by software to control the frequency of the main PLL output clock PLLPCLK. This output can be selected for ADC. These bits can be written only if PLL is disabled.

PLLPCLK output clock frequency = VCO frequency / PLLP with PLLP = 2, 3, 4,... or 32 [VCO frequency / (N + 1)]

0000: reserved
00001: PLLP = 2
00010: PLLP = 3
00011: PLLP = 4
00100: PLLP = 5
...
11111: PLLP = 32

The software has to set these bits so that 64 MHz is not exceeded on this domain.

Bit 16 **PLLPEN:** Main PLL PLLPCLK output enable

Set and reset by software to enable the PLLPCLK output of the main PLL.

In order to save power, when the PLLPCLK output of the PLL is not used, the value of PLLPEN should be 0.

0: PLLPCLK output disable
1: PLLPCLK output enable

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLN[6:0]:** Main PLL multiplication factor for VCO

Set and cleared by software to control the multiplication factor of the VCO. These bits can be written only when the PLL is disabled.

VCO output frequency = VCO input frequency x PLLN with 6 ≤ PLLN ≤ 127

0000000: PLLN = 0 reserved (must not be used)

0000001: PLLN = 1 reserved (must not be used)

...

0000101: PLLN = 5 reserved (must not be used)

0000110: PLLN = 6

1111111: PLLN = 127

The software has to set these bits to ensure that the VCO output frequency is between 96 and 344 MHz.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PLLM[2:0]:** Division factor for the main PLL

Set and cleared by software to divide the PLL input clock before the VCO. These bits can be written only when the PLLis disabled.

VCO input frequency = PLL input clock frequency / PLLM with 1 ≤ PLLM ≤ 8

000: PLLM = 1

001: PLLM = 2

010: PLLM = 3

011: PLLM = 4

100: PLLM = 5

101: PLLM = 6

110: PLLM = 7

111: PLLM = 8

The software has to set these bits to ensure that the VCO input frequency ranges from 2.66 to 16 MHz.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSRC[1:0]:** Main PLL entry clock source

Set and cleared by software to select PLL clock source. These bits can be written only when PLLis disabled.

In order to save power, when no PLL is used, the value of PLLSRC should be 00.

00: No clock sent to PLL

01: MSI clock selected as PLL clock entry

10: HSI16 clock selected as PLL clock entry

11: HSE clock selected as PLL clock entry

### 7.4.5 RCC clock interrupt enable register (RCC_CIER)

Address offset: 0x018

Reset value: 0x0000 0000

Access: No wait state, word, half-word and byte access

| 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. | LSI2 RDYIE | HSI48 RDYIE | LSE CSSIE | Res. | Res. | Res. | PLL RDYIE | HSE RDYIE | HSI RDYIE | MSI RDYIE | LSE RDYIE | LSI1 RDYIE |
| | | | | rw | rw | rw | | | | rw | rw | rw | rw | rw | rw |

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **LSI2RDYIE:** LSI2 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSI2 oscillator stabilization.

0: LSI2 ready interrupt disabled

1: LSI2 ready interrupt enabled

Bit 10 **HSI48RDYIE:** HSI48 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the internal HSI48 oscillator.

0: HSI48 ready interrupt disabled

1: HSI48 ready interrupt enabled

Bit 9 **LSECSSIE:** LSE clock security system interrupt enable

Set and cleared by software to enable/disable interrupt caused by the clock security system on LSE.

0: Clock security interrupt caused by LSE clock failure disabled

1: Clock security interrupt caused by LSE clock failure enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **PLLRDYIE:** PLL ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by PLL lock.

0: PLL lock interrupt disabled

1: PLL lock interrupt enabled

Bit 4 **HSERDYIE:** HSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization.

0: HSE ready interrupt disabled

1: HSE ready interrupt enabled

Bit 3 **HSIRDYIE:** HSI16 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSI16 oscillator stabilization.

0: HSI16 ready interrupt disabled

1: HSI16 ready interrupt enabled

Bit 2 **MSIRDYIE:** MSI ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the MSI oscillator stabilization.

0: MSI ready interrupt disabled

1: MSI ready interrupt enabled

Bit 1 **LSERDYIE:** LSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled
1: LSE ready interrupt enabled

Bit 0 **LSI1RDYIE:** LSI1 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the LSI1 oscillator stabilization.
0: LSI1 ready interrupt disabled
1: LSI1 ready interrupt enabled

### 7.4.6 RCC clock interrupt flag register (RCC_CIFR)

Address offset: 0x01C

Reset value: 0x0000 0000

Access: No wait state, word, half-word and byte access

| 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. | LSI2 RDYF | HSI48R DYF | LSE CSSF | CSSF | Res. | Res. | PLL RDYF | HSE RDYF | HSI RDYF | MSI RDYF | LSE RDYF | LSI1 RDYF |
| | | | | r | r | r | r | | | r | r | r | r | r | r |

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **LSI2RDYF:** LSI2 ready interrupt flag
Set by hardware when the LSI2 clock becomes stable and LSI2RDYDIE is set.
Cleared by software setting the LSI2RDYC bit.
0: No clock ready interrupt caused by the LSI2 oscillator
1: Clock ready interrupt caused by the LSI2 oscillator

Bit 10 **HSI48RDYF:** HSI48 ready interrupt flag
Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set in a response to setting the HSI48ON (refer to *RCC clock recovery RC register (RCC_CRRCR)*).
Cleared by software setting the HSI48RDYC bit.
0: No clock ready interrupt caused by the HSI48 oscillator
1: Clock ready interrupt caused by the HSI48 oscillator

Bit 9 **LSECSSF:** LSE clock security system interrupt flag
Set by hardware when a failure is detected in the LSE oscillator.
Cleared by software setting the LSECSSC bit.
0: No clock security interrupt caused by LSE clock failure
1: Clock security interrupt caused by LSE clock failure

Bit 8 **CSSF:** HSE clock security system interrupt flag
Set by hardware when a failure is detected in the HSE oscillator.
Cleared by software setting the CSSC bit.
0: No clock security interrupt caused by HSE clock failure
1: Clock security interrupt caused by HSE clock failure

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

Bit 5 **PLLRDYF:** PLL ready interrupt flag
Set by hardware when the PLL locks and PLLRDYDIE is set.
Cleared by software setting the PLLRDYC bit.
0: No clock ready interrupt caused by PLL lock
1: Clock ready interrupt caused by PLL lock

Bit 4 **HSERDYF:** HSE ready interrupt flag
Set by hardware when the HSE clock becomes stable and HSERDYDIE is set.
Cleared by software setting the HSERDYC bit.
0: No clock ready interrupt caused by the HSE oscillator
1: Clock ready interrupt caused by the HSE oscillator

Bit 3 **HSIRDYF:** HSI16 ready interrupt flag
Set by hardware when the HSI16 clock becomes stable and HSIRDYDIE is set in a response to setting the HSION (refer to *RCC clock control register (RCC_CR)*). When HSION is not set but the HSI16 oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated.
Cleared by software setting the HSIRDYC bit.
0: No clock ready interrupt caused by the HSI16 oscillator
1: Clock ready interrupt caused by the HSI16 oscillator

Bit 2 **MSIRDYF:** MSI ready interrupt flag
Set by hardware when the MSI clock becomes stable and MSIRDYDIE is set.
Cleared by software setting the MSIRDYC bit.
0: No clock ready interrupt caused by the MSI oscillator
1: Clock ready interrupt caused by the MSI oscillator

Bit 1 **LSERDYF:** LSE ready interrupt flag
Set by hardware when the LSE clock becomes stable and LSERDYDIE is set.
Cleared by software setting the LSERDYC bit.
0: No clock ready interrupt caused by the LSE oscillator
1: Clock ready interrupt caused by the LSE oscillator

Bit 0 **LSI1RDYF:** LSI1 ready interrupt flag
Set by hardware when the LSI1 clock becomes stable and LSI1RDYDIE is set.
Cleared by software setting the LSI1RDYC bit.
0: No clock ready interrupt caused by the LSI1 oscillator
1: Clock ready interrupt caused by the LSI1 oscillator

### 7.4.7 RCC clock interrupt clear register (RCC_CICR)

Address offset: 0x020

Reset value: 0x0000 0000

Access: No wait state, word, half-word and byte access

| 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. | LSI2 RDYC | HSI48 RDYC | LSE CSSC | CSSC | Res. | Res. | PLL RDYC | HSE RDYC | HSI RDYC | MSI RDYC | LSE RDYC | LSI1 RDYC |
| | | | | w | w | w | w | | | w | w | w | w | w | w |

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **LSI2RDYC:** LSI2 ready interrupt clear
This bit is set by software to clear the LSI2RDYF flag.
0: No effect
1: LSI2RDYF cleared

Bit 10 **HSI48RDYC:** HSI48 oscillator ready interrupt clear
This bit is set by software to clear the HSI48RDYF flag.
0: No effect
1: Clear the HSI48RDYC flag

Bit 9 **LSECSSC:** LSE clock security system interrupt clear
This bit is set by software to clear the LSECSSF flag.
0: No effect
1: Clear LSECSSF flag

Bit 8 **CSSC:** HSE clock security system interrupt clear
This bit is set by software to clear the HSE CSSF flag.
0: No effect
1: Clear HSE CSSF flag

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

Bit 5 **PLLRDYC:** PLL ready interrupt clear
This bit is set by software to clear the PLLRDYF flag.
0: No effect
1: Clear PLLRDYF flag

Bit 4 **HSERDYC:** HSE ready interrupt clear
This bit is set by software to clear the HSERDYF flag.
0: No effect
1: Clear HSERDYF flag

Bit 3 **HSIRDYC:** HSI16 ready interrupt clear
This bit is set software to clear the HSIRDYF flag.
0: No effect
1: Clear HSIRDYF flag

Bit 2 **MSIRDYC:** MSI ready interrupt clear
This bit is set by software to clear the MSIRDYF flag.
0: No effect
1: MSIRDYF cleared

Bit 1 **LSERDYC:** LSE ready interrupt clear
This bit is set by software to clear the LSERDYF flag.
0: No effect
1: LSERDYF cleared

Bit 0 **LSI1RDYC:** LSI1 ready interrupt clear
This bit is set by software to clear the LSI1RDYF flag.
0: No effect
1: LSI1RDYF cleared

### 7.4.8 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x028

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access

| 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. | CRC RST | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1 RST | Res. | DMA1 RST |
| | | | rw | | | | | | | | | | rw | | rw |

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

Bit 12 **CRCRST:** CRC reset
Set and cleared by software.
0: No effect
1: Reset CRC

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1RST:** DMAMUX reset
Set and cleared by software.
0: No effect
1: Reset DMAMUX1

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMA1RST:** DMA1 reset
Set and cleared by software.
0: No effect
1: Reset DMA1

### 7.4.9 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x02C

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access

| 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. | ADC RST | Res. | Res. | Res. | Res. | Res. | GPIOH RST | Res. | Res. | GPIOE RST | Res. | GPIOC RST | GPIOB RST | GPIOA RST |
| | | rw | | | | | | rw | | | rw | | rw | rw | rw |

Bits 31:148 Reserved, must be kept at reset value.

Bit 13 **ADCRST:** ADC reset
Set and cleared by software.
0: No effect
1: Reset ADC interface

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRST:** IO port H reset
Set and cleared by software.
0: No effect
1: Reset IO port H

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOERST:** IO port E reset
Set and cleared by software.
0: No effect
1: Reset IO port E

Bit 3 Reserved, must be kept at reset value.

Bit 2 **GPIOCRST:** IO port C reset
Set and cleared by software.
0: No effect
1: Reset IO port C

Bit 1 **GPIOBRST:** IO port B reset
Set and cleared by software.
0: No effect
1: Reset IO port B

Bit 0 **GPIOARST:** IO port A reset
Set and cleared by software.
0: No effect
1: Reset IO port A

### 7.4.10 RCC AHB3 and AHB4 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x030

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | FLASH RST | Res. | Res. | Res. | Res. | IPCC RST | HSEM RST | RNG RST | AES2 RST | PKA RST |
| | | | | | | 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHRST:** Flash interface reset

This bit can only be set when the Flash memory is in power-down. Set and cleared by software.

0: No effect

1: Reset Flash interface.

Bits 24:21 Reserved, must be kept at reset value.

Bit 20 **IPCCRST:** IPCC interface reset

Set and cleared by software.

0: No effect

1: Reset IPCC

Bit 19 **HSEMRST:** HSEM reset

Set and cleared by software.

0: No effect

1: Reset HSEM

Bit 18 **RNGRST:** True RNG reset

Set and cleared by software.

0: No effect

1: Reset true RNG

Bit 17 **AES2RST:** AES2 hardware accelerator reset

Set and cleared by software.

0: No effect

1: Reset AES2

Bit 16 **PKARST:** PKA hardware accelerator reset

Set and cleared by software.

0: No effect

1: Reset PKA

Bits 15:0 Reserved, must be kept at reset value.

### 7.4.11 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)

Address offset: 0x038

Reset value: 0x0000 0000

Access: No wait state, word, half-word and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPTIM1 RST | Res. | Res. | Res. | Res. | Res. | Res. | .Res. | Res. | Res. | I2C1 RST | 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. | TIM2 RST |
| | | | | | | | | | | | | | | | rw |

Bit 31 **LPTIM1RST:** Low power timer 1 reset
Set and cleared by software.
0: No effect
1: Reset LPTIM1

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **I2C1RST:** I2C1 reset
Set and cleared by software.
0: No effect
1: Reset I2C1

Bits 20:1 Reserved, must be kept at reset value.

Bit 0 **TIM2RST:** TIM2 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM2

### 7.4.12 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x03C

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access

| 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. | LPTIM2 RST | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | rw | | | | | |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2RST:** Low-power timer 2 reset
Set and cleared by software.
0: No effect
1: Reset LPTIM2

Bits 4:0 Reserved, must be kept at reset value.

### 7.4.13 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x040

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access

| 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. | TIM17 RST | TIM16 RST | Res. |
| | | | | | | | | | | | | | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | USART1 RST | Res. | SPI1 RST | TIM1 RST | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | rw | | rw | rw | | | | | | | | | | | |

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

Bit 18 **TIM17RST:** TIM17 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM17 timer

Bit 17 **TIM16RST:** TIM16 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM16 timer

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1RST:** USART1 reset
Set and cleared by software.
0: No effect
1: Reset USART1

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST:** SPI1 reset
Set and cleared by software.
0: No effect
1: Reset SPI1

Bit 11 **TIM1RST:** TIM1 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM1 timer

Bits 10:0 Reserved, must be kept at reset value.

### 7.4.14 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x044

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access

| 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. | RFRST |
| | | | | | | | | | | | | | | | rw |

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

Bit 0 **RFRST:** Radio system BLE and 802.15.4 reset.
Set and cleared by software.
0: No effect
1: Reset radio system BLE and 802.15.4. The reset status of the radio system can be obtained from RFRSTS in *RCC control/status register (RCC_CSR)*.

### 7.4.15 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x048

Reset value: 0x0000 0000

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not 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. | CRCEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1EN | Res. | DMA1EN |
| | | | rw | | | | | | | | | | rw | | rw |

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

Bit 12 **CRCEN:** CPU1 CRC clock enable
Set and cleared by software.
0: CRC clock disable for CPU1
1: CRC clock enable for CPU1

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1:** CPU1 DMAMUX1 clock enable
Set and cleared by software.
0: DMAMUX1 clock disable for CPU1
1: DMAMUX1 clock enable for CPU1

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMA1EN:** CPU1 DMA1 clock enable
Set and cleared by software.
0: DMA1 clock disable for CPU1
1: DMA1 clock enable for CPU1

### 7.4.16 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x04C

Reset value: 0x0000 0000

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not 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. | ADC EN | Res. | Res. | Res. | Res. | Res. | GPIOH EN | Res. | Res. | GPIOE EN | Res. | GPIOC EN | GPIOB EN | GPIOA EN |
| | | rw | | | | | | rw | | | rw | | rw | rw | rw |

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

Bit 13 **ADCEN:** CPU1 ADC clocks enable
Set and cleared by software.
0: ADC bus and kernel clocks disabled for CPU1
1: ADC bus and kernel clocks enabled for CPU1

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHEN:** CPU1 IO port H clock enable
Set and cleared by software.
0: IO port H clock disabled for CPU1
1: IO port H clock enabled for CPU1

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOEEN:** CPU1 IO port E clock enable
Set and cleared by software.
0: IO port E clock disabled for CPU1
1: IO port E clock enabled for CPU1

Bit 3 Reserved, must be kept at reset value.

Bit 2 **GPIOCEN:** CPU1 IO port C clock enable
Set and cleared by software.
0: IO port C clock disabled for CPU1
1: IO port C clock enabled for CPU1

Bit 1 **GPIOBEN:** CPU1 IO port B clock enable
Set and cleared by software.
0: IO port B clock disabled for CPU1
1: IO port B clock enabled for CPU1

Bit 0 **GPIOAEN:** CPU1 IO port A clock enable
Set and cleared by software.
0: IO port A clock disabled for CPU1
1: IO port A clock enabled for CPU1

### 7.4.17 RCC AHB3 and AHB4 peripheral clock enable register (RCC_AHB3ENR)

Address offset: 0x050

Reset value: 0x0208 0000

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | FLASH EN | Res. | Res. | Res. | Res. | IPCC EN | HSEM EN | RNG EN | AES2 EN | PKA EN |
| | | | | | | 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHEN** CPU1 Flash memory interface clock enable

This bit can only be cleared when the Flash memory is in Power down. Set and cleared by software.

0: Flash interface clock disable for CPU1
1: Flash interface clock enable for CPU1

Bits 24:21 Reserved, must be kept at reset value.

Bit 20 **IPCCEN** CPU1 IPCC interface clock enable

Set and cleared by software.
0: IPCC clock disable for CPU1
1: IPCC clock enable for CPU1

Bit 19 **HSEMEN** CPU1 HSEM clock enable

Set and cleared by software.
0: HSEM clock disable for CPU1
1: HSEM clock enable for CPU1

Bit 18 **RNGEN** CPU1 true RNG clocks enable

Set and cleared by software.
0: True RNG bus and kernel clocks disable for CPU1
1: True RNG bus and kernel clocks enable for CPU1

Bit 17 **AES2EN** CPU1 AES2 accelerator clock enable

Set and cleared by software.
0: AES2 clock disable for CPU1
1: AES2 clock enable for CPU1

Bit 16 **PKAEN** CPU1 PKA accelerator clock enable

Set and cleared by software.
0: PKA clock disable for CPU1
1: PKA clock enable for CPU1

Bits 15:0 Reserved, must be kept at reset value.

### 7.4.18 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address: 0x058

Reset value: 0x0000 0400

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPTIM1 EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | I2C1 EN | 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. | WWDG EN | RTCAPB EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM2 EN |
| | | | | rs | rw | | | | | | | | | | rw |

Bit 31 **LPTIM1EN:** CPU1 Low power timer 1 clocks enable
Set and cleared by software.
0: LPTIM1 bus and kernel clocks disabled for CPU1
1: LPTIM1 bus and kernel clocks enabled for CPU1

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN:** CPU1 I2C1 clocks enable
Set and cleared by software.
0: I2C1 bus and kernel clocks disabled for CPU1
1: I2C1 bus and kernel clocks enabled for CPU1

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN:** CPU1 Window watchdog clock enable
Set by software to enable the window watchdog clock. Reset by hardware system reset. This bit can also be set by hardware if the WWDG_SW option bit is reset.
0: Window watchdog clock disabled for CPU1
1: Window watchdog clock enabled for CPU1

Bit 10 **RTCAPBEN:** CPU1 RTC APB clock enable
Set and cleared by software
0: RTC APB clock disabled for CPU1
1: RTC APB clock enabled for CPU1

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **TIM2EN:** CPU1 TIM2 timer clock enable
Set and cleared by software.
0: TIM2 clock disabled for CPU1
1: TIM2 clock enabled for CPU1

### 7.4.19 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x05C

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not 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. | LPTIM2 EN | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | rw | | | | | |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2EN** CPU1 Low power timer 2 clocks enable
Set and cleared by software.
0: LPTIM2 bus and kernel clocks disabled for CPU1
1: LPTIM2 bus and kernel clocks enabled for CPU1

Bits 4:0 Reserved, must be kept at reset value.

## 7.4.20 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x060

Reset value: 0x0000 0000

Access: word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not 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. | TIM17 EN | TIM16 EN | Res. |
| | | | | | | | | | | | | | rw | rw | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | USART1 EN | Res. | SPI1 EN | TIM1 EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | rw | | rw | rw | | | | | | | | | | | |

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

Bit 18 **TIM17EN:** CPU1 TIM17 timer clock enable
Set and cleared by software.
0: TIM17 timer clock disabled for CPU1
1: TIM17 timer clock enabled for CPU1

Bit 17 **TIM16EN:** CPU1 TIM16 timer clock enable
Set and cleared by software.
0: TIM16 timer clock disabled for CPU1
1: TIM16 timer clock enabled for CPU1

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1EN:** CPU1 USART1 clocks enable
Set and cleared by software.
0: USART1 bus and kernel clocks disabled for CPU1
1: USART1 bus and kernel clocks enabled for CPU1

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN:** CPU1 SPI1 clock enable
Set and cleared by software.
0: SPI1 clock disabled for CPU1
1: SPI1 clock enabled for CPU1

Bit 11 **TIM1EN:** CPU1 TIM1 timer clock enable
Set and cleared by software.
0: TIM1 timer clock disabled for CPU1
1: TIM1P timer clock enabled for CPU1

Bits 10:0 Reserved, must be kept at reset value.

### 7.4.21 RCC AHB1 peripheral clocks enable in Sleep modes register (RCC_AHB1SMENR)

Address offset: 0x068

Reset value: 0x0001 1207

Access: No wait state, word, half-word and byte access

| 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. | CRC SMEN | Res. | Res. | SRAM1 SMEN | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1 SMEN | Res. | DMA1 SMEN |
| | | | rw | | | rw | | | | | | | rw | | rw |

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

Bit 12 **CRCSMEN:** CRC clock enable during CPU1 CSleep mode
Set and cleared by software.
0: CRC clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: CRC clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAM1SMEN:** SRAM1 interface clock enable during CPU1 CSleep mode
Set and cleared by software.
0: SRAM1 interface clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: SRAM1 interface clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1SMEN:** DMAMUX1 clock enable during CPU1 CSleep mode
Set and cleared by software during Sleep mode.
0: DMAMUX1 clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: DMAMUX1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMA1SMEN:** DMA1 clock enable during CPU1 CSleep mode
Set and cleared by software.
0: DMA1 clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: DMA1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

### 7.4.22 RCC AHB2 peripheral clocks enable in Sleep modes register (RCC_AHB2SMENR)

Address offset: 0x06C

Reset value: 0x0001 209F

Access: No wait state, word, half-word and byte access

| 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. | ADC SMEN | Res. | Res. | Res. | Res. | Res. | GPIOH SMEN | Res. | Res. | GPIOE SMEN | Res. | GPIOC SMEN | GPIOB SMEN | GPIOA SMEN |
| | | rw | | | | | | rw | | | rw | | rw | rw | rw |

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

Bit 13 **ADCSMEN:** ADC clock enable during CPU1 Csleep and CStop modes
Set and cleared by software.
0: ADC bus clock disabled by the clock gating during CPU1 Csleep and CStop modes.
1: ADC bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHSMEN:** IO port H clock enable during CPU1 CSleep mode
Set and cleared by software.
0: IO port H clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: IO port H clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOESMEN:** IO port E clock enable during CPU1 CSleep mode
Set and cleared by software.
0: IO port E clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: IO port E clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bit 3 Reserved, must be kept at reset value.

Bit 2 **GPIOCSMEN:** IO port C clock enable during CPU1 CSleep mode
Set and cleared by software.
0: IO port C clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: IO port C clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bit 1 **GPIOBSMEN:** IO port B clock enable during CPU1 CSleep mode
Set and cleared by software.
0: IO port B clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: IO port B clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bit 0 **GPIOASMEN:** IO port A clock enable during CPU1 CSleep mode
Set and cleared by software.
0: IO port A clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: IO port A clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

### 7.4.23 RCC AHB3 and AHB4 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR)

Address offset: 0x070

Reset value: 0x0307 0100

Access: No wait state, word, half-word and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | FLASH SMEN | SRAM2S MEN | Res. | Res. | Res. | Res. | Res. | RNG SMEN | AES2 SMEN | PKA SMEN |
| | | | | | | 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHSMEN:** Flash memory interface clock enable during CPU1 CSleep mode
Set and cleared by software.
0: Flash interface clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: Flash interface clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 24 **SRAM2SMEN:** SRAM2a and SRAM2b memory interface clock enable during CPU1 CSleep mode
Set and cleared by software.
0: SRAM2 clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: SRAM2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **RNGSMEN:** True RNG clock enable during CPU1 Csleep and CStop modes
Set and cleared by software.
0: True RNG bus clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: True RNG bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 17 **AES2MEN:** AES2 accelerator clock enable during CPU1 CSleep mode
Set and cleared by software.
0: AES2 clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: AES2 clock enabled by the clock gating during CPU1 CSleep mode, .disabled during CPU1 CStop mode

Bit 16 **PKASMEN:** PKA accelerator clock enable during CPU1 CSleep mode
Set and cleared by software.
0: PKA clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: PKA clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 15:0 Reserved, must be kept at reset value.

### 7.4.24 RCC APB1 peripheral clocks enable in Sleep mode register 1 (RCC_APB1SMENR1)

Address: 0x078

Reset value: 0x85A0 4E01

Access: No wait state, word, half-word and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPTIM1 SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | I2C1 SMEN | 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. | WWDG SMEN | RTCAPB SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM2 SMEN |
| | | | | rw | rw | | | | | | | | | | rw |

Bit 31 **LPTIM1SMEN:** Low power timer 1 clock enable during CPU1 Csleep and CStop mode
Set and cleared by software.
0: LPTIM1 bus clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: LPTIM1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **I2C1SMEN:** I2C1 clock enable during CPU1 Csleep and CStop modes
Set and cleared by software.
0: I2C1 bus clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: I2C1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **WWDGSMEN:** Window watchdog clocks enable during CPU1 CSleep mode

Set and cleared by software. This bit is forced to ‘1’ by hardware when the hardware WWDG_SW option is reset.

0: Window watchdog clock disabled by the clock gating during CPU1 Csleep and CStop modes

1: Window watchdog clocks enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 10 **RTCAPBSMEN:** RTC bus clock enable during CPU1 CSleep mode

Set and cleared by software

0: RTC bus clock disabled during by the clock gating during CPU1 Csleep and CStop modes.

1: RTC bus clock enabled during by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **TIM2SMEN:** TIM2 timer clock enable during CPU1 CSleep mode

Set and cleared by software.

0: TIM2 clock disabled by the clock gating during CPU1 Csleep and CStop modes

1: TIM2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

### 7.4.25 RCC APB1 peripheral clocks enable in Sleep mode register 2 (RCC_APB1SMENR2)

Address offset: 0x07C

Reset value: 0x0000 0021

Access: No wait state, word, half-word and byte access

| 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. | LPTIM2 SMEN | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | rw | | | | | |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2SMEN** Low power timer 2 clock enable during CPU1 Csleep and CStop modes

Set and cleared by software.

0: LPTIM2 bus clock disabled by the clock gating during CPU1 Csleep and CStop modes.

1: LPTIM2 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 4:0 Reserved, must be kept at reset value.

### 7.4.26 RCC APB2 peripheral clocks enable in Sleep mode register (RCC_APB2SMENR)

Address: 0x080

Reset value: 0x0026 5800

Access: word, half-word and byte access

| 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. | TIM17 SMEN | TIM16 SMEN | Res. |
| | | | | | | | | | | | | | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | USART1 SMEN | Res. | SPI1 SMEN | TIM1 SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | rw | | rw | rw | | | | | | | | | | | |

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

Bit 18 **TIM17SMEN:** TIM17 timer clock enable during CPU1 CSleep mode
Set and cleared by software.
0: TIM17 timer clock disabled by the clock gating during CPU1 Csleep and CStop mode
1: TIM17 timer clocks enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 17 **TIM16SMEN:** TIM16 timer clock enable during CPU1 CSleep mode
Set and cleared by software.
0: TIM16 timer clock disabled by the clock gating during CPU1 Csleep and CStop mode
1: TIM16 timer clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN:** USART1 clock enable during CPU1 Csleep and CStop modes.
Set and cleared by software.
0: USART1 bus clock disabled by the clock gating during CPU1 Csleep and CStop mode
1: USART1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SMEN:** SPI1 clock enable during CPU1 CSleep mode
Set and cleared by software.
0: SPI1 clock disabled by the clock gating during CPU1 Csleep and CStop mode
1: SPI1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 11 **TIM1SMEN:** TIM1 timer clock enable during CPU1 CSleep mode
Set and cleared by software.
0: TIM1 timer clock disabled by the clock gating during CPU1 Csleep and CStop mode
1: TIM1 timer clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 10:0 Reserved, must be kept at reset value.

### 7.4.27 RCC peripherals independent clock configuration register (RCC_CCIPR)

Address: 0x088

Reset value: 0x0000 0000

Access: no wait states, word, half-word and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RNGSEL[1:0] | | ADCSEL[1:0] | | CLK48SEL[1:0] | | Res. | Res. | Res. | Res. | LPTIM2SEL[1:0] | | LPTIM1SEL[1:0] | | Res. | Res. |
| 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. | I2C1SEL[1:0] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | USART1SEL[1:0] | |
| | | rw | rw | | | | | | | | | | | rw | rw |

Bits 31:30 **RNGSEL[1:0]:** RNG clock source selection

These bits are set and cleared by software to select the clock source used by the true RNG.
00: Use clock as selected by CLK48SEL
01: Select LSI clock
10: Select LSE clock
11: Reserved

Bits 29:28 **ADCSEL[1:0]:** ADC clock source selection

These bits are set and cleared by software to select the clock source used by the ADC interface.
00: No clock selected
01: Reserved
10: PLL “P” clock (PLLPCLK) selected as ADC clock
11: System clock (SYSCLK) selected as ADC clock

Bits 27:26 **CLK48SEL[1:0]:** 48 MHz clock source selection

These bits are set and cleared by software to select the 48 MHz clock source used by true RNG. The true RNG clock source selection is furthermore selected by RNGSEL.
00: HSI48 clock selected as 48 MHz clock
01: Reserved
10: PLL “Q” clock (PLLQCLK) selected as 48 MHz clock
11: MSI clock selected as 48 MHz clock

Bits 25:22 Reserved, must be kept at reset value.

Bits 21:20 **LPTIM2SEL[1:0]:** Low power timer 2 clock source selection

These bits are set and cleared by software to select the LPTIM2 clock source.
00: PCLK selected as LPTIM2 clock
01: LSI clock selected as LPTIM2 clock
10: HSI16 clock selected as LPTIM2 clock
11: LSE clock selected as LPTIM2 clock

Bits 19:18 **LPTIM1SEL[1:0]:** Low power timer 1 clock source selection

These bits are set and cleared by software to select the LPTIM1 clock source.
00: PCLK selected as LPTIM1 clock
01: LSI clock selected as LPTIM1 clock
10: HSI16 clock selected as LPTIM1 clock
11: LSE clock selected as LPTIM1 clock

Bits 17:14 Reserved, must be kept at reset value.

Bits 13:12 **I2C1SEL[1:0]:** I2C1 clock source selection

These bits are set and cleared by software to select the I2C1 clock source.
00: PCLK selected as I2C1 clock
01: System clock (SYSCLK) selected as I2C1 clock
10: HSI16 clock selected as I2C1 clock
11: reserved

Bits 11:2 Reserved, must be kept at reset value.

Bits 1:0 **USART1SEL[1:0]:** USART1 clock source selection
This bit is set and cleared by software to select the USART1 clock source.
00: PCLK selected as USART1 clock
01: System clock (SYSCLK) selected as USART1 clock
10: HSI16 clock selected as USART1 clock
11: LSE clock selected as USART1 clock

### 7.4.28 RCC backup domain control register (RCC_BDCR)

Address offset: 0x090

Reset value: 0x0000 0000, (reset by Backup domain reset, except LSCOSEL, LSCOEN and BDRST, which are reset only by Backup domain power-on reset, not reset by wakeup from Standby and System reset)

Access: 0 ≤wait state ≤3, word, half-word and byte access.
Wait states are inserted in case of successive accesses to this register.

*Note: The bits of the RCC backup domain control register (RCC_BDCR) are outside of the $V_{CORE}$ domain. As a result, after Reset, these bits are write-protected and the DBP bit in the PWR control register 1 (PWR_CR1) has to be set before these can be modified. Refer to Section 6.1.2: Battery backup domain for further information. These bits (except LSCOSEL, LSCOEN and BDRST) are only reset after a Backup domain reset. Any internal or external Reset will not have any effect on these bits.*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | LSCO SEL | LSCO EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BDRST |
| | | | | | | rw | rw | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTCEN | Res. | Res. | Res. | Res. | Res. | RTCSEL[1:0] | | Res. | LSE CSSD | LSE CSSON | LSEDRV[1:0] | | LSE BYP | LSE RDY | LSEON |
| rw | | | | | | rw | rw | | r | rw | rw | rw | rw | r | rw |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **LSCOSEL:** Low speed clock output selection
Set and cleared by software.
0: LSI clock selected
1: LSE clock selected

Bit 24 **LSCOEN:** Low speed clock output enable
Set and cleared by software.
0: Low speed clock output (LSCO) disable
1: Low speed clock output (LSCO) enable

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **BDRST:** Backup domain software reset
Set and cleared by software.
0: Reset not activated
1: Resets the entire Backup domain

Bit 15 **RTCEN:** RTC clock enable

Set and cleared by software.
0: RTC clock disabled
1: RTC clock enabled

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

Bits 9:8 **RTCSEL[1:0]:** RTC clock source selection

Set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset, or unless a failure is detected on LSE (LSECSSD is set). The BDRST bit can be used to reset them.
00: No clock
01: LSE oscillator clock used as RTC clock
10: LSI oscillator clock used as RTC clock
11: HSE oscillator clock divided by 32 used as RTC clock

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LSECSSD** CSS on LSE failure detection

Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator (LSE).
0: No failure detected on LSE (32 kHz oscillator)
1: Failure detected on LSE (32 kHz oscillator)

Bit 5 **LSECSSON** CSS on LSE enable

Set by software to enable the clock security system on LSE (32 kHz oscillator).
LSECSSON must be enabled after the LSE oscillator is enabled (LSEON bit enabled) and ready (LSERDY flag set by hardware), and after the RTCSEL bit is selected.
Once enabled this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case the software MUST disable the LSECSSON bit.
0: CSS on LSE (32 kHz external oscillator) OFF
1: CSS on LSE (32 kHz external oscillator) ON

Bits 4:3 **LSEDRV[1:0]** LSE oscillator drive capability

Set by software to modulate the LSE oscillator drive capability.
00: ‘Xtal mode’ lower driving capability
01: ‘Xtal mode’ medium low driving capability
10: ‘Xtal mode’ medium high driving capability
11: ‘Xtal mode’ higher driving capability
The oscillator is in Xtal mode when it is not in bypass mode.

Bit 2 **LSEBYP:** LSE oscillator bypass

Set and cleared by software to bypass oscillator. This bit can be written only when the external 32 kHz oscillator is disabled (LSEON = 0 and LSERDY = 0).
0: LSE oscillator not bypassed
1: LSE oscillator bypassed

Bit 1 **LSERDY:** LSE oscillator ready

Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after six external low-speed oscillator clock cycles.
0: LSE oscillator not ready
1: LSE oscillator ready

Bit 0 **LSEON:** LSE oscillator enable
Set and cleared by software.
0: LSE oscillator OFF
1: LSE oscillator ON

### 7.4.29 RCC control/status register (RCC_CSR)

Address: 0x094

Reset value: 0x0C00 0000 (reset by System reset, except reset flags by POR only, not reset by wakeup from Standby)

Access: 0 ≤ wait state ≤ 3, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPWR RSTF | WWDG RSTF | IWWG RSTF | SFT RSTF | BOR RSTF | PIN RSTF | OB L RSTF | Res. | RMVF | Res. | Res. | Res. | Res. | Res. | Res. | RF RSTS |
| r | r | r | r | r | r | r | | rw | | | | | | | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RFWKPSEL[1:0] | | Res. | Res. | LSI2TRIM[3:0] | | | | Res. | Res. | Res. | Res. | LSI2 RDY | LSI2 ON | LSI1 RDY | LSI1 ON |
| rw | rw | | | rw | rw | rw | rw | | | | | r | rw | r | rw |

Bit 31 **LPWRRSTF:** Low-power reset flag
Set by hardware when a reset occurs due to illegal Stop, Standby or Shutdown mode entry.
Cleared by writing to the RMVF bit.
0: No illegal mode reset occurred
1: Illegal mode reset occurred

Bit 30 **WWDGRSTF:** Window watchdog reset flag
Set by hardware when a window watchdog reset occurs.
Cleared by writing to the RMVF bit.
0: No window watchdog reset occurred
1: Window watchdog reset occurred

Bit 29 **IWDGRSTF**: Independent window watchdog reset flag
Set by hardware when an independent watchdog reset domain occurs.
Cleared by writing to the RMVF bit.
0: No independent watchdog reset occurred
1: Independent watchdog reset occurred

Bit 28 **SFTRSTF:** Software reset flag
Set by hardware when a software reset occurs.
Cleared by writing to the RMVF bit.
0: No software reset occurred
1: Software reset occurred

Bit 27 **BORRSTF:** BOR flag
Set by hardware when a BOR occurs.
Cleared by writing to the RMVF bit.
0: No BOR occurred
1: BOR occurred

Bit 26 **PINRSTF:** Pin reset flag
Set by hardware when a reset from the NRST pin occurs.
Cleared by writing to the RMVF bit.
0: No reset from NRST pin occurred
1: Reset from NRST pin occurred

Bit 25 **OBLRSTF:** Option byte loader reset flag
Set by hardware when a reset from the Option Byte loading occurs.
Cleared by writing to the RMVF bit.
0: No reset from Option Byte loading occurred
1: Reset from Option Byte loading occurred

Bit 24 Reserved, must be kept at reset value.

Bit 23 **RMVF:** Remove reset flag
Set by software to clear the reset flags.
0: No effect
1: Clear the reset flags

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **RFRSTS:** Radio system BLE and 802.15.4 reset status
Set and cleared by hardware
0: Radio system BLE and 802.15.4 not in reset, radio system can be accessed
1: Radio system BLE and 802.15.4 under reset, radio system cannot be accessed

Bits 15:14 **RFWKPSEL[1:0]**: RF system wakeup clock source selection
Set by software to select the clock source for RF system wakeup logic.
00: No clock
01: LSE oscillator clock used as RF system wakeup clock
10: Reserved
11: HSE oscillator clock divided by 1024 used as RF system wakeup clock

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:8 **LSI2TRIM[3:0]**: LSI2 oscillator trim.
*Note: LSI2TRIM must be changed only when LSI2 is disabled (LSI2ON = 0).*

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

Bit 3 **LSI2RDY:** LSI2 oscillator ready
Set and cleared by hardware to indicate when the LSI2 oscillator is stable. After the LSI2ON bit is cleared, LSI2RDY goes low after three LSI2 oscillator clock cycles.
0: LSI2 oscillator not ready
1: LSI2 oscillator ready

Bit 2 **LSI2ON:** LSI2 oscillator enable and selection
Set and cleared by software.
0: LSI2 oscillator OFF (LSI1 selected on LSI)
1: LSI2 oscillator ON(LSI2 when ready selected on LSI)

Bit 1 **LSI1RDY:** LSI1 oscillator ready

Set and cleared by hardware to indicate when the LSI1 oscillator is stable. After the LSI1ON bit is cleared, LSI1RDY goes low after three LSI1 oscillator clock cycles. This bit can be set even if LSI1ON = 0 if the LSI1 is requested by the clock security system on LSE, by the Independent watchdog or by the RTC.

0: LSI1 oscillator not ready

1: LSI1 oscillator ready

Bit 0 **LSI1ON:** LSI1 oscillator enable

Set and cleared by software.

0: LSI1 oscillator OFF

1: LSI1 oscillator ON

## 7.4.30 RCC clock recovery RC register (RCC_CRRCR)

Address: 0x098

Reset value: 0x0000 XXX0 where X is factory-programmed.

Access: No wait state, word, half-word and byte access

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HSI48CAL[8:0] | | | | | | | | | Res. | Res. | Res. | Res. | Res. | HSI48 RDY | HSI48 ON |
| r | r | r | r | r | r | r | r | r | | | | | | r | rw |

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

Bits 15:7 **HSI48CAL[8:0]:** HSI48 clock calibration

These bits are initialized at startup with the factory-programmed HSI48 calibration trim value. They are ready only.

Bits 6:2 Reserved, must be kept at reset value

Bit 1 **HSI48RDY:** HSI48 clock ready flag

Set by hardware to indicate that HSI48 oscillator is stable. This bit is set only when HSI48 is enabled by software by setting HSI48ON.

0: HSI48 oscillator not ready

1: HSI48 oscillator ready

Bit 0 **HSI48ON:** HSI48 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSI48 when entering in Stop, Standby or Shutdown modes.

0: HSI48 oscillator OFF

1: HSI48 oscillator ON

## 7.4.31 RCC clock HSE register (RCC_HSECR)

Address: 0x09C

Reset value: 0x0000 0030 (reset by System reset, not reset by wakeup from Standby)

Access: Write access is protected and software must write a KEY (0xCAFE CAFE) as a word access in this register to unlock the register. Once unlocked a single write access (word, half-word or byte) can be performed to the register. It is then locked again. 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. | HSETUNE[5:0] | | | | | | Res. | HSEGMC[2:0] | | | HSES | Res. | Res. | UNLOCKED |
| | | r | r | r | r | r | r | | rw | rw | rw | rw | | | r |

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

Bits 13:8 **HSETUNE[5:0]:** HSE capacitor tuning

Can be changed by software. Do not change when HSE is on.
0x00: minimum load capacitance
0x3F maximum load capacitance

Bits 7 Reserved, must be kept at reset value

Bits 6:4 **HSEGMC[2:0]:** HSE current control

Can be changed by software. Do not change when HSE is on.
000: current max limit 0.18 mA/V
001: current max limit 0.57 mA/V
010: current max limit 0.78 mA/V
011: current max limit 1.13 mA/V
100: current max limit 0.61 mA/V
101: current max limit 1.65 mA/V
110: current max limit 2.12 mA/V
111: current max limit 2.84 mA/V

*Note: The HSEGMC value must be greater than $g_{mcrit}$, whose value must be calculated according to AN2867 "Oscillator design guide for STM8S, STM8A and STM32 microcontrollers", available on www.st.com.*

Bit 3 **HSES:** HSE Sense amplifier threshold

Can be changed by software. Do not change when HSE is on.
0: HSE bias current factor 1/2
1: HSE bias current factor 3/4

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

Bit 0 **UNLOCKED:** HSE clock control register unlocked

Set and cleared by hardware.
0: register locked, the key has not been programmed, or a write access has been performed to the register
1: Register unlocked. Key 0xCAFE CAFE has been written to unlock this register, enabling a single data write

### 7.4.32 RCC extended clock recovery register (RCC_EXTCFGR)

Address: 0x108

Reset value: 0x0003 0000.

Access: No wait state, word, half-word and byte access

| 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. | RFCSS | Res. | Res. | C2HPREF | SHDHPREF |
| | | | | | | | | | | | r | | | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | C2HPRE[3:0] | | | | SHDHPRE[3:0] | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bit 20 **RFCSS:** Radio system HCLK5 and APB3 selected clock source indication

Set and reset by hardware to indicate which clock source is selected for the Radio system HCLK5 and APB3 clock.

0: HSI16 used for Radio system HCLK5 and APB3 clock

1: HSE divided by 2 used for Radio system HCLK5 and APB3 clock

Bits 19:18 Reserved, must be kept at reset value

Bit 17 **C2HPREF:** HCLK2 prescaler flag (CPU2)

Set and reset by hardware to acknowledge HCLK2 prescaler programming

Reset when a new prescaler value is programmed in C2HPRE. set when the programmed value is actually applied.

0: HCLK2 prescaler value not yet applied

1: HCLK2 prescaler value applied

Bit 16 **SHDHPREF:** HCLK4 shared prescaler flag (AHB4, Flash memory and SRAM2)

Set and reset by hardware to acknowledge Shared HCLK4 prescaler programming

Reset when a new prescaler value is programmed in SHDHPRE. set when the programmed value is actually applied.

0: HCLK4 prescaler value not yet applied

1: HCLK4 prescaler value applied

Bits 15:8 Reserved, must be kept at reset value

Bits 7:4 **C2HPRE[3:0]:** HCLK2 prescaler (CPU2)

Set and cleared by software to control the division factor of the HCLK2 clock (CPU2).
The C2HPREF flag can be checked to know if the programmed C2HPRE prescaler value is applied.

**Caution:** The software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details refer to *Section 7.2.10: Clock source frequency*). After a write operation to these bits the register bit C2HPREF must be read to be sure that the new value has been taken into account.

0001: SYSCLK divided by 3
0010: SYSCLK divided by 5
0101: SYSCLK divided by 6
0110: SYSCLK divided by 10
0111: SYSCLK divided by 32
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512
Others: SYSCLK not divided

Bit 3:0 **SHDHPRE[3:0]:** HCLK4 shared prescaler (AHB4, Flash memory and SRAM2)

Set and cleared by software to control the division factor of the Shared HCLK4 clock (AHB4, Flash memory and SRAM2).
The SHDHPREF flag can be checked to know if the programmed SHDHPRE prescaler value is applied.

**Caution:** The software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details refer to *Section 7.2.10: Clock source frequency*). After a write operation to these bits the register bit SHDHPRE must be read to be sure that the new value has been taken into account.

0001: SYSCLK divided by 3
0010: SYSCLK divided by 5
0101: SYSCLK divided by 6
0110: SYSCLK divided by 10
0111: SYSCLK divided by 32
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512
Others: SYSCLK not divided

### 7.4.33 RCC CPU2 AHB1 peripheral clock enable register (RCC_C2AHB1ENR)

Address offset: 0x148

Reset value: 0x0000 0000

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not 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. | CRCEN | Res. | Res. | SRAM1EN | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1EN | Res. | DMA1EN |
| | | | rw | | | rw | | | | | | | rw | | rw |

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

Bit 12 **CRCEN:** CPU2 CRC clock enable
Set and cleared by software.
0: CRC clock disable for CPU2
1: CRC clock enable for CPU2

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAM1EN:** CPU2 SRAM1 clock enable
Set and cleared by software.
0: SRAM1 clock disabled for CPU2
1: SRAM1 clock enabled for CPU2

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1:** CPU2 DMAMUX1 clock enable
Set and cleared by software.
0: DMAMUX1 clock disable for CPU2
1: DMAMUX1 clock enable for CPU2

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMA1EN:** CPU2 DMA1 clock enable
Set and cleared by software.
0: DMA1 clock disable for CPU2
1: DMA1 clock enable for CPU2

### 7.4.34 RCC CPU2 AHB2 peripheral clock enable register (RCC_C2AHB2ENR)

Address offset: 0x14C

Reset value: 0x0000 0000

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not 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. | ADC EN | Res. | Res. | Res. | Res. | Res. | GPIOH EN | Res. | Res. | GPIOE EN | Res. | GPIOC EN | GPIOB EN | GPIOA EN |
| | | rw | | | | | | rw | | | rw | | rw | rw | rw |

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

Bit 13 **ADCEN:** ADC clocks enable
Set and cleared by software.
0: ADC bus and kernel clocks disabled for CPU2
1: ADC bus and kernel clocks enabled for CPU2

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHEN:** CPU2 IO port H clock enable
Set and cleared by software.
0: IO port H clock disabled for CPU2
1: IO port H clock enabled for CPU2

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOEEN:** CPU2 IO port E clock enable
Set and cleared by software.
0: IO port E clock disabled for CPU2
1: IO port E clock enabled for CPU2

Bit 3 Reserved, must be kept at reset value.

Bit 2 **GPIOCEN:** CPU2 IO port C clock enable
Set and cleared by software.
0: IO port C clock disabled for CPU2
1: IO port C clock enabled for CPU2

Bit 1 **GPIOBEN:** CPU2 IO port B clock enable
Set and cleared by software.
0: IO port B clock disabled for CPU2
1: IO port B clock enabled for CPU2

Bit 0 **GPIOAEN:** CPU2 IO port A clock enable
Set and cleared by software.
0: IO port A clock disabled for CPU2
1: IO port A clock enabled for CPU2

### 7.4.35 RCC CPU2 AHB3 and AHB4 peripheral clock enable register (RCC_C2AHB3ENR)

Address offset: 0x150

Reset value: 0x0208 0000

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | FLASH EN | Res. | Res. | Res. | Res. | IPCC EN | HSEM EN | RNG EN | AES2 EN | PKA EN |
| | | | | | | 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHEN** CPU2 Flash interface clock enable
This bit can only be cleared when the Flash is in Power down. Set and cleared by software.
0: Flash interface clock disable for CPU2
1: Flash interface clock enable for CPU2

Bits 24:21 Reserved, must be kept at reset value.

Bit 20 **IPCCEN** CPU2 IPCC interface clock enable
Set and cleared by software.
0: IPCC clock disable for CPU2
1: IPCC clock enable for CPU2

Bit 19 **HSEMEN** CPU2 HSEM clock enable
Set and cleared by software.
0: HSEM clock disable for CPU2
1: HSEM clock enable for CPU2

Bit 18 **RNGEN** CPU2 true RNG clocks enable
Set and cleared by software.
0: True RNG bus and kernel clocks disable for CPU2
1: True RNG bus and kernel clocks enable for CPU2

Bit 17 **AES2EN** CPU2 AES2 accelerator clock enable
Set and cleared by software.
0: AES2 clock disable for CPU2
1: AES2 clock enable for CPU2

Bit 16 **PKAEN** CPU2 PKA accelerator clock enable
Set and cleared by software.
0: PKA clock disable for CPU2
1: PKA clock enable for CPU2

Bits 15:0 Reserved, must be kept at reset value.

### 7.4.36 RCC CPU2 APB1 peripheral clock enable register 1 (RCC_C2APB1ENR1)

Address: 0x158

Reset value: 0x0000 0400

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPTIM1 EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | I2C1 EN | 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. | RTCAPB EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM2 EN |
| | | | | | rw | | | | | | | | | | rw |

Bit 31 **LPTIM1EN:** CPU2 Low power timer 1 clocks enable
Set and cleared by software.
0: LPTIM1 bus and kernel clocks disabled for CPU2
1: LPTIM1 bus and kernel clocks enabled for CPU2

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN:** CPU2 I2C1 clocks enable
Set and cleared by software.
0: I2C1 bus and kernel clocks disabled for CPU2
1: I2C1 bus and kernel clocks enabled for CPU2

Bits 20:11 Reserved, must be kept at reset value.

Bit 10 **RTCAPBEN:** CPU2 RTC APB clock enable
Set and cleared by software
0: RTC APB clock disabled for CPU2
1: RTC APB clock enabled for CPU2

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **TIM2EN:** CPU2 TIM2 timer clock enable
Set and cleared by software.
0: TIM2 clock disabled for CPU2
1: TIM2 clock enabled for CPU2

### 7.4.37 RCC CPU2 APB1 peripheral clock enable register 2 (RCC_C2APB1ENR2)

Address offset: 0x15C

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not 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. | LPTIM2EN | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | rw | | | | | |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2EN** CPU2 Low power timer 2 clocks enable
Set and cleared by software.
0: LPTIM2 bus and kernel clocks disable for CPU2
1: LPTIM2 bus and kernel clocks enable for CPU2

Bits 4:0 Reserved, must be kept at reset value.

### 7.4.38 RCC CPU2 APB2 peripheral clock enable register (RCC_C2APB2ENR)

Address: 0x160

Reset value: 0x0000 0000

Access: word, half-word and byte access

*Note: When the peripheral clock is not active the peripheral registers read or write access from CPU2 is not 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. | TIM17 EN | TIM16 EN | Res. |
| | | | | | | | | | | | | | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | USART1 EN | Res. | SPI1 EN | TIM1 EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | rw | | rw | rw | | | | | | | | | | | |

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

Bit 18 **TIM17EN:** CPU2 TIM17 timer clock enable
Set and cleared by software.
0: TIM17 timer clock disabled for CPU2
1: TIM17 timer clock enabled for CPU2

Bit 17 **TIM16EN:** CPU2 TIM16 timer clock enable
Set and cleared by software.
0: TIM16 timer clock disabled for CPU2
1: TIM16 timer clock enabled for CPU2

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1EN:** CPU2 UFSART1 clock enable
Set and cleared by software.
0: USART1 bus and kernel clocks disabled for CPU2
1: USART1 bus and kernel clocks enabled for CPU2

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN:** CPU2 SPI1 clock enable
Set and cleared by software.
0: SPI1 clock disabled for CPU2
1: SPI1 clock enabled for CPU2

Bit 11 **TIM1EN:** CPU2 TIM1 timer clock enable
Set and cleared by software.
0: TIM1 timer clock disabled for CPU2
1: TIM1P timer clock enabled for CPU2

Bits 10:0 Reserved, must be kept at reset value.

### 7.4.39 RCC CPU2 APB3 peripheral clock enable register (RCC_C2APB3ENR)

Address offset: 0x164

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access

*Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not 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. | 802EN | BLEEN |
| | | | | | | | | | | | | | | rw | rw |

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

Bit 1 **802EN:** CPU2 802.15.4 interface clock enable
Set and cleared by software.
0: 802.15.4 clock disable for CPU2
1: 802.15.4 clock enable for CPU2

Bit 0 **BLEEN:** CPU2 BLE interface clock enable
Set and cleared by software.
0: BLE clock disable for CPU2
1: BLE clock enable for CPU2

### 7.4.40 RCC CPU2 AHB1 peripheral clocks enable in Sleep modes register (RCC_C2AHB1SMENR)

Address offset: 0x168

Reset value: 0x0001 1007

Access: No wait state, word, half-word and byte access

| 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. | CRC SMEN | Res. | Res. | SRAM1 SMEN | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1 SMEN | Res. | DMA1 SMEN |
| | | | rw | | | rw | | | | | | | rw | | rw |

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

Bit 12 **CRCSMEN:** CRC clock enable during CPU2 CSleep mode
Set and cleared by software.
0: CRC clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: CRC clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAM1SMEN**: SRAM1 interface clock enabled during CPU2 CSleep mode
Set and cleared by software
0: SRAM1 interface clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: SRAM1 interface clock enabled by the clock gating during CPU2 CSleep mode, disabled during CStop mode

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1SMEN:** DMAMUX1 clock enable during CPU2 CSleep mode
Set and cleared by software during Sleep mode.
0: DMAMUX1 clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: DMAMUX1 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMA1SMEN:** DMA1 clock enable during CPU2 CSleep mode
Set and cleared by software.
0: DMA1 clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: DMA1 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

### 7.4.41 RCC CPU2 AHB2 peripheral clocks enable in Sleep modes register (RCC_C2AHB2SMENR)

Address offset: 0x16C

Reset value: 0x0001 209F

Access: No wait state, word, half-word and byte access

| 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. | ADC SMEN | Res. | Res. | Res. | Res. | Res. | GPIOH SMEN | Res. | Res. | GPIOE SMEN | Res. | GPIOC SMEN | GPIOB SMEN | GPIOA SMEN |
| | | **rw** | | | | | | rw | | | rw | | rw | rw | rw |

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

Bit 13 **ADCSMEN:** ADC clock enable during CPU2 CSleep and CStop modes
Set and cleared by software.
0: ADC bus clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: ADC bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHSMEN:** IO port H clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port H clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: IO port H clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOESMEN:** IO port E clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port E clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: IO port E clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 3 Reserved, must be kept at reset value.

Bit 2 **GPIOCSMEN:** IO port C clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port C clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: IO port C clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 1 **GPIOBSMEN:** IO port B clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port B clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: IO port B clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 0 **GPIOASMEN:** IO port A clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port A clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: IO port A clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

### 7.4.42 RCC CPU2 AHB3 and AHB4 peripheral clocks enable in Sleep mode register (RCC_C2AHB3SMENR)

Address offset: 0x170

Reset value: 0x0307 0000

Access: No wait state, word, half-word and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | FLASH SMEN | SRAM2 SMEN | Res. | Res. | Res. | Res. | Res. | RNG SMEN | AES2 SMEN | PKA SMEN |
| | | | | | | 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHSMEN** Flash memory interface clock enable during CPU2 CSleep mode
Set and cleared by software.
0: Flash memory interface clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: Flash memory interface clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 24 **SRAM2SMEN:** SRAM2a and SRAM2b interface clock enable during CPU2 CSleep mode
Set and cleared by software.
0: SRAM2 clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: SRAM2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **RNGSMEN** True RNG clock enable during CPU2 CSleep and CStop mode
Set and cleared by software.
0: True RNG bus clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: True RNG bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 17 **AES2SMEN** AES2 accelerator clock enable during CPU2 CSleep mode
Set and cleared by software.
0: AES2 clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: AES2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 16 **PKASMEN:** PKA accelerator clock enable during CPU2 CSleep mode
Set and cleared by software.
0: PKA clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: PKA clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 15:0 Reserved, must be kept at reset value.

### 7.4.43 RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 1 (RCC_C2APB1SMENR1)

Address: 0x178

Reset value: 0x85A0 4601

Access: No wait state, word, half-word and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPTIM1 SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | I2C1 SMEN | 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. | RTCAPB SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM2 SMEN |
| | | | | | rw | | | | | | | | | | rw |

Bit 31 **LPTIM1SMEN:** Low power timer 1 clock enable during CPU2 CSleep and CStop mode
Set and cleared by software.
0: LPTIM1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: LPTIM1 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **I2C1SMEN:** I2C1 clock enable during CPU2 CSleep and CStop modes
Set and cleared by software.
0: I2C1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: I2C1 bus clock enabled by the clock gating during CPU2 CSleep modes disabled during CPU2 CStop mode

Bits 20:11 Reserved, must be kept at reset value.

Bit 10 **RTCAPBSMEN:** RTC bus clock enable during CPU2 CSleep mode
Set and cleared by software
0: RTC bus clock disabled during by the clock gating during CPU2 CSleep and CStop modes
1: RTC bus clock enabled during by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **TIM2SMEN:** TIM2 timer clock enable during CPU2 CSleep mode
Set and cleared by software.
0: TIM2 clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: TIM2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

### 7.4.44 RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 2 (RCC_C2APB1SMENR2)

Address offset: 0x17C

Reset value: 0x0000 0021

Access: No wait state, word, half-word and byte access

| 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. | LPTIM2 SMEN | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | rw | | | | | |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2SMEN** Low power timer 2 clock enable during CPU2 CSleep and CStop modes.
Set and cleared by software.
0: LPTIM2 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: LPTIM2 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 4:0 Reserved, must be kept at reset value.

### 7.4.45 RCC CPU2 APB2 peripheral clocks enable in Sleep mode register (RCC_C2APB2SMENR)

Address: 0x180

Reset value: 0x0026 5800

Access: word, half-word and byte access

| 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. | TIM17 SMEN | TIM16 SMEN | Res. |
| | | | | | | | | | | | | | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | USART1 SMEN | Res. | SPI1 SMEN | TIM1SM EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | rw | | rw | rw | | | | | | | | | | | |

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

Bit 18 **TIM17SMEN:** TIM17 timer clock enable during CPU2 CSleep mode

Set and cleared by software.

0: TIM17 timer clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: TIM17 timer clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 17 **TIM16SMEN:** TIM16 timer clock enable during CPU2 CSleep mode

Set and cleared by software.

0: TIM16 timer clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: TIM16 timer clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN:** USART1 clock enable during CPU2 CSleep and CStop mode

Set and cleared by software.

0: USART1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: USART1 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SMEN:** SPI1 clock enable during CPU2 CSleep mode

Set and cleared by software.

0: SPI1 clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: SPI1 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 11 **TIM1SMEN:** TIM1 timer clock enable during CPU2 CSleep mode

Set and cleared by software.

0: TIM1 timer clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: TIM1 timer clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 10:0 Reserved, must be kept at reset value.

### 7.4.46 RCC CPU2 APB3 peripheral clock enable in Sleep mode register (RCC_C2APB3SMENR)

Address offset: 0x184

Reset value: 0x00000 0003

Access: word, half-word and byte access

| 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. | 802SMEN | BLESMEN |
| | | | | | | | | | | | | | | rw | rw |

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

Bit 1 **802SMEN:** 802.15.4 interface bus clock enable during CPU2 CSleep mode

Set and cleared by software.

0: 802.15.4 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: 802.15.4 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 0 **BLESMEN:** BLE interface bus clock enable during CPU2 CSleep mode

Set and cleared by software.

0: BLE bus clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: BLE bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

*Note: The BLE interface bus clock is also enabled by the BLE IP itself.*

## 7.4.47 RCC register map

The following table gives the RCC register map and the reset values.

**Table 37. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000 | RCC_CR | Res. | Res. | Res. | Res. | Res. | Res. | PLLRDY | PLLON | Res. | Res. | Res. | HSEPRE | CSSON | Res. | HSERDY | HSEON | Res. | Res. | Res. | HSIKERDY | HSIASFS | HSIRDY | HSIKERON | HSION | MSIRANGE [3:0] | | | | Res. | MSIPLLEN | MSIRDY | MSION |
| | Reset value | | | | | | | 0 | 0 | | | | 0 | 0 | | 0 | 0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | | 0 | 0 | 1 |
| 0x004 | RCC_ICSCR | Res. | HSITRIM[6:0] | | | | | | | HSICAL[7:0] | | | | | | | | MSITRIM[7:0] | | | | | | | | MSICAL[7:0] | | | | | | | |
| | Reset value | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x |
| 0x008 | RCC_CFGR | Res. | MCOPRE [2:0] | | | MCOSEL[3:0] | | | | Res. | Res. | Res. | Res. | Res. | PPRE2F | PPRE1F | HPREF | STOPWUCK | Res. | PPRE2 [2:0] | | | PPRE1 [2:0] | | | HPRE[3:0] | | | | SWS [1:0] | | SW [1:0] | |
| | Reset value | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00C | RCC_PLLCFGR | PLLR[2:0] | | | PLLREN | PLLQ[2:0] | | | PLLQEN | Res. | Res. | PLLP[4:0] | | | | | PLLPEN | Res. | PLLN[6:0] | | | | | | | Res. | PLLM[2:0] | | | Res. | Res. | PLLSRC [1:0] | |
| | Reset value | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | | | 0 | 0 | 0 | 1 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | 0 | 0 | 0 | | | 0 | 0 |
| 0x010 | 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. |
| 0x018 | RCC_CIER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LSI2RDYIE | HSI48RDYIE | LSECSSIE | Res. | Res. | Res. | PLLRDYIE | HSERDYIE | HSIRDYIE | MSIRDYIE | LSERDYIE | LSI1RDYIE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x01C | RCC_CIFR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LSI2RDYF | HSI48RDYF | LSECSSF | CSSF | Res. | Res. . | PLLRDYF | HSERDYF | HSIRDYF | MSIRDYF | LSERDYF | LSI1RDYF |
| | Reset value | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x020 | RCC_CICR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LSI2RDYC | HSI48RDYC | LSECSSC | CSSC | Res. | Res. | PLLRDYC | HSERDYC | HSIRDYC | MSIRDYC | LSERDYC | LSI1RDYC |
| | Reset value | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x024 | 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. |
| 0x028 | RCC_AHB1RSTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CRCRST | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1RST | Res. | DMA1RST |
| | Reset value | | | | | | | | | | | | | | | | | | | | 0 | | | | | | | | | | 0 | | 0 |
| 0x02C | RCC_AHB2RSTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res.. | Res. | Res. | ADCRST | Res. | Res. | Res. | Res. | Res. | GPIOHRST | Res. | Res. | GPIOERST | Res. | GPIOCRST | GPIOBRST | GPIOARST |
| | Reset value | | | | | | | | | | | | | | | | | | | 0 | | | | | | 0 | | | 0 | | 0 | 0 | 0 |

**Table 37. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x030 | RCC_AHB3RSTR | Res. | Res. | Res. | Res. | Res. | Res. | FLASHRST | Res. | Res. | Res. | Res. | IPCCRST | HSEMRST | RNGRST | AES2RST | PKARST | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | |
| 0x034 | 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. |
| 0x038 | RCC_ APB1RSTR1 | LPTIM1RST | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | I2C1RST | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM2RST |
| | Reset value | 0 | | | | | | | | | | 0 | | | | | | | | | | | | | | | | | | | | | 0 |
| 0x03C | RCC_ APB1RSTR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LPTIM2RST | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | |
| 0x040 | RCC_ APB2RSTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM17RST | TIM16RST | Res. | Res. | USART1RST | Res. | SPI1RST | TIM1RST | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | 0 | 0 | | | 0 | | 0 | 0 | | | | | | | | | | | |
| 0x044 | RCC_APB3RSTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | RFRST |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 0x048 | RCC_AHB1ENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CRCEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1EN | Res. | DMA1EN |
| | Reset value | | | | | | | | | | | | | | | | | | | | 0 | | | | | | | | | | 0 | | 0 |
| 0x04C | RCC_AHB2ENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ADCEN | Res. | Res. | Res. | Res. | Res. | GPIOHEN | Res. | Res. | GPIOEEN | Res. | GPIOCEN | GPIOBEN | GPIOAEN |
| | Reset value | | | | | | | | | | | | | | | | | | | 0 | | | | | | 0 | | | 0 | | 0 | 0 | 0 |
| 0x050 | RCC_AHB3ENR | Res. | Res. | Res. | Res. | Res. | Res. | FLASHEN | Res. | Res. | Res. | Res. | IPCCEN | HSEMEN | RNGEN | AES2EN | PKAEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | 1 | | | | | 0 | 1 | 0 | 0 | 0 | | | | | | | | | | | | | | | | |
| 0x054 | 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. |
| 0x058 | RCC_APB1ENR1 | LPTIM1EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | I2C1EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WWDGEN | RTCAPBEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM2EN |
| | Reset value | 0 | | | | | | | | | | 0 | | | | | | | | | | 0 | 1 | | | | | | | | | | 0 |
| 0x05C | RCC_APB1ENR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LPTIM2EN | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | |

**Table 37. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x060 | RCC_APB2ENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM17EN | TIM16EN | Res. | Res. | USART1EN | Res. | SPI1EN | TIM1EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | 0 | 0 | | | 0 | | 0 | 0 | | | | | | | | | | | |
| 0x064 | 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. |
| 0x068 | RCC_ AHB1SMENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CRCSMEN | Res. | Res. | SRAM1SMEN | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1SMEN | Res. | DMA1SMEN |
| | Reset value | | | | | | | | | | | | | | | | | | | | 1 | | | 1 | | | | | | | 1 | | 1 |
| 0x06C | RCC_ AHB2SMENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ADCFSSMEN | Res. | Res. | Res. | Res. | Res. | GPIOHSMEN | Res. | Res. | GPIOESMEN | Res. | GPIOCSMEN | GPIOBSMEN | GPIOASMEN |
| | Reset value | | | | | | | | | | | | | | | | | | | 1 | | | | | | 1 | | | 1 | | 1 | 1 | 1 |
| 0x070 | RCC _AHB3SMENR | Res. | Res. | Res. | Res. | Res. | Res. | FLASHSMEN | SRAM2SMEN | Res. | Res. | Res. | Res. | Res. | RNGSMEN | AES2SMEN | PKASMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | 1 | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | | | | | | | |
| 0x074 | 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. |
| 0x078 | RCC _APB1SMENR1 | LPTIM1SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | I2C1SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WWDGSMEN | RTCAPBSMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM2SMEN |
| | Reset value | 1 | | | | | | | | | | 1 | | | | | | | | | | 1 | 1 | | | | | | | | | | 1 |
| 0x07C | RCC _APB1SMENR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LPTIM2SMEN | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | |
| 0x080 | RCC_ APB2SMENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM17SMEN | TIM16SMEN | Res. | Res. | USART1SMEN | Res. | SPI1SMEN | TIM1SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | 1 | 1 | | | 1 | | 1 | 1 | | | | | | | | | | | |
| 0x084 | 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. |
| 0x088 | RCC_CCIPR | RNGSEL [1:0] | | ADCSEL [1:0] | | CLK48SEL [1:0] | | Res. | Res. | Res. | Res. | LPTIM2SEL [1:0] | | LPTIM1SEL [1:0] | | Res. | Res. | Res. | Res. | I2C1SEL [1:0] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | USART1SEL [1:0] | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 0 | | | | | 0 | 0 | | | | | | | | | | | 0 | 0 |
| 0x08C | 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. |

**Table 37. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x090 | RCC_BDCR | Res. | Res. | Res. | Res. | Res. | Res. | LSCOSEL | LSCOEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BDRST | RTCEN | Res. | Res. | Res. | Res. | Res. | RTCSEL [1:0] | | Res. | LSECSSD | LSECSSON | LSEDRV [1:0] | | LSEBYP | LSERDY | LSEON |
| | Reset value | | | | | | | 0 | 0 | | | | | | | | 0 | 0 | | | | | | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x094 | RCC_CSR | LPWRRSTF | WWDGRSTF | IWDGRSTF | SFTRSTF | BORRSTF | PINRSTF | OBLRSTF | Res. | RMVF | Res. | Res. | Res. | Res. | Res. | Res. | RFRSTS | RFWKPSEL [1:0]. | | Res. | Res. | LSI2TRIM [3:0] | | | | Res. | Res. | Res. | Res. | LSI2RDY | LSI2ON | LSI1RDY | LSI1ON |
| | Reset value | 0 | 0 | 0 | 0 | 1 | 1 | 0 | | 0 | | | | | | | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 0 |
| 0x098 | RCC_CRRCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | HSI48CAL[8:0] | | | | | | | | | Res. | Res. | Res. | Res. | Res. | HSI48RDY | HSI48ON |
| | Reset value | | | | | | | | | | | | | | | | | x | x | x | x | x | x | x | x | x | | | | | | 0 | 0 |
| 0x09C | RCC_HSECR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | HSETUNE[5:0] | | | | | | Res. | HSEGMC [2:0] | | | HSES | Res. | Res. | UNLOCKED |
| | Reset value | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0x0A0 to 0x104 | 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. |
| 0x108 | RCC_EXTCFGR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | RFCSS | Res. | Res. | C2HPREF | SHDHPREF | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | C2HPRE[3:0] | | | | SHDHPRE [3:0] | | | |
| | Reset value | | | | | | | | | | | | 0 | | | 1 | 1 | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10C to 0x144 | 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. |
| 0x148 | RCC _C2AHB1ENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CRCEN | Res. | Res. | SRAM1EN | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1EN | Res. | DMA1EN |
| | Reset value | | | | | | | | | | | | | | | | | | | | 0 | | | 0 | | | | | | | 0 | | 0 |
| 0x14C | RCC _C2AHB2ENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ADCEN | Res. | Res. | Res. | Res. | Res. | GPIOHEN | Res. | Res. | GPIOEEN | Res. | GPIOCEN | GPIOBEN | GPIOAEN |
| | Reset value | | | | | | | | | | | | | | | | | | | 0 | | | | | | 0 | | | 0 | | 0 | 0 | 0 |
| 0x150 | RCC _C2AHB3ENR | Res. | Res. | Res. | Res. | Res. | Res. | FLASHEN | Res. | Res. | Res. | Res. | IPCCEN | HSEMEN | RNGEN | AES2EN | PKAEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | 1 | | | | | 0 | 1 | 0 | 0 | 0 | | | | | | | | | | | | | | | | |
| 0x154 | 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. |
| 0x158 | RCC _C2APB1ENR1 | LPTIM1EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | I2C1EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | RTCAPBEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM2EN |
| | Reset value | 0 | | | | | | | | | | 0 | | | | | | | | | | | 1 | | | | | | | | | | 0 |

**Table 37. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x15C | RCC _C2APB1ENR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LPTIM2EN | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | |
| 0x160 | RCC _C2APB2ENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM17EN | TIM16EN | Res. | Res. | USART1EN | Res. | SPI1EN | TIM1EN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | 0 | 0 | | | 0 | | 0 | 0 | | | | | | | | | | | |
| 0x164 | RCC _C2APB3ENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 802EN | BLEEN |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 |
| 0x168 | RCC _C2AHB1SMENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CRCSMEN | Res. | Res. | SRAM1SMEN | Res. | Res. | Res. | Res. | Res. | Res. | DMAMUX1SMEN | Res. | DMA1SMEN |
| | Reset value | | | | | | | | | | | | | | | | | | | | 1 | | | 0 | | | | | | | 1 | | 1 |
| 0x16C | RCC _C2AHB2SMENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ADCSMEN | Res. | Res. | Res. | Res. | Res. | GPIOHSMEN | Res. | Res. | GPIOESMEN | Res. | GPIOCSMEN | GPIOBSMEN | GPIOASMEN |
| | Reset value | | | | | | | | | | | | | | | | | | | 1 | | | | | | 1 | | | 1 | | 1 | 1 | 1 |
| 0x170 | RCC _C2AHB33MENR | Res. | Res. | Res. | Res. | Res. | Res. | FLASHSMEN | SRAM2SMEN | Res. | Res. | Res. | Res. | Res. | RNGSMEN | AES2SMEN | PKASMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | 1 | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | | | | | | | |
| 0x174 | 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. |
| 0x178 | RCC_C2 APB1SMENR1 | LPTIM1SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | I2C1SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | RTCAPBSMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res | Res. | Res. | TIM2SMEN |
| | Reset value | 1 | | | | | | | | | | 1 | | | | | | | | | | | 1 | | | | | | | | | | 1 |
| 0x17C | RCC_C2 APB1SMENR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LPTIM2SMEN | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | |
| 0x180 | RCC_C2 APB2SMENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TIM17SMEN | TIM16SMEN | Res. | Res. | USART1SMEN | Res. | SPI1SMEN | TIM1SMEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | 1 | 1 | | | 1 | | 1 | 1 | | | | | | | | | | | |

**Table 37. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x184 | RCC_C2 APB3SMENR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 802SMEN | BLESMEN |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 |
| 0x188 to 03FC | 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 on page 56* for the register boundary addresses.

# 8 General-purpose I/Os (GPIO)

## 8.1 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).

## 8.2 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 two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

## 8.3 GPIO functional description

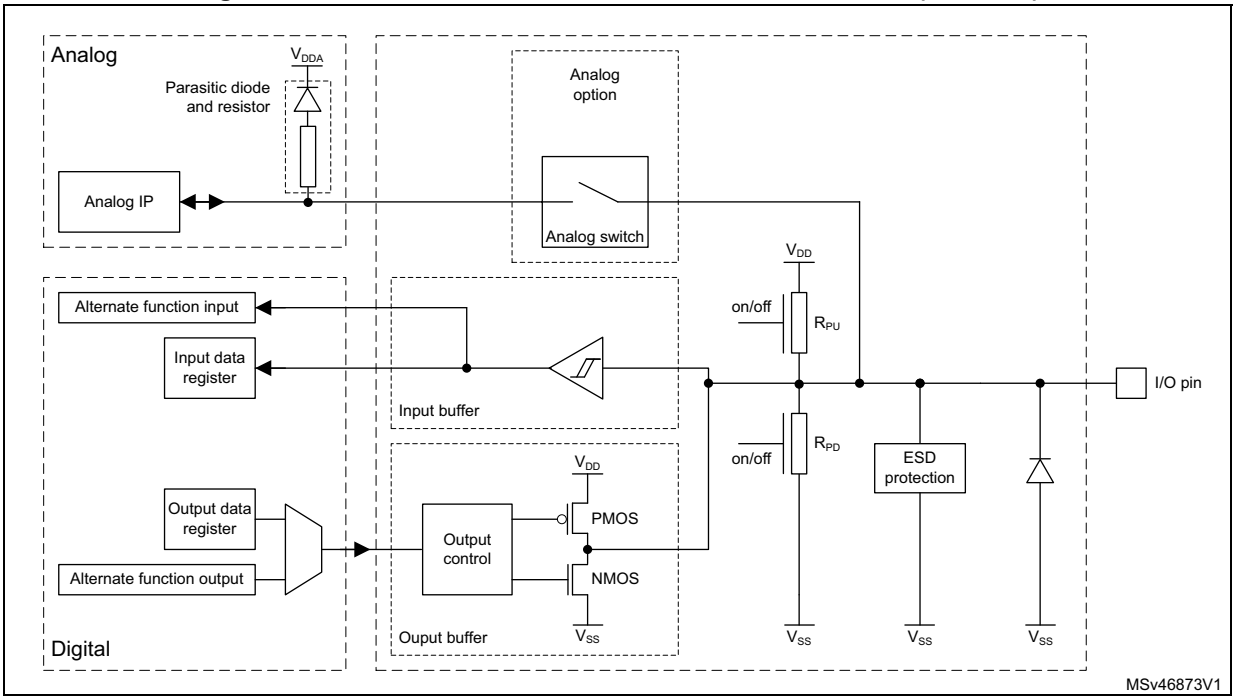
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- 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 register 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.

*Figure 19* shows the basic structure of a Three-volt tolerant (TT) and Five-volt tolerant (FT) I/O port bit.

**Figure 19. Three-volt or Five-volt tolerant GPIO structure (TT or FT)**




*Note: The parasitic diode in the analog domain is connected to $V_{DDA}$ and cannot be used as a protection diode.*

*The voltage level called $V_{DD_FT}$ in some datasheets and reference manuals is within the ESD protection block.*

*When the analog option is selected, the FT I/O is not five-volt tolerant anymore since the pin is supplied with $V_{DDA}$.*

A TT or FT GPIO pin has no internal protection diode connected to supply ($V_{DD}$). There is no physical limitation against over-voltage. Therefore, for applications requiring a limited voltage threshold, it is recommended to connect an external diode to $V_{DD}$.

*Table 38* gives the possible port bit configurations.

**Table 38. 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 | |
| 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.

### 8.3.1 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 analog mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in input mode with pull-up
- PA14: JTCK/SWCLK in input mode with pull-down
- PA13: JTMS/SWDAT in input mode with pull-up
- PB4: NJTRST in input mode with pull-up
- PB3: JTDO in Hi-Z mode no pulls

PH3/BOOT0 is in input mode during the reset until at least the end of the option byte loading phase. See *Section 8.3.15: Using PH3 as GPIO*.

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.

### 8.3.2 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 sixteen alternate function inputs (AF0 to AF15) 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 device datasheet.

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
- **GPIO:** configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral 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.
- **Additional functions:**
  - For the ADC and PVD, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC and COMP registers.
  - For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

### 8.3.3 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.

### 8.3.4 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 *Section 8.4.5: GPIO port input data register (GPIOx_IDR) (x = A to C and E and H)* and *Section 8.4.6: GPIO port output data register (GPIOx_ODR) (x = A to C and E and H)* for the register descriptions.

### 8.3.5 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.

### 8.3.6 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 *Section 8.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to C and E and H)*) 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 *Section 8.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to C and E and H)*.

### 8.3.7 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, the user can connect an alternate function to some other pin as required by the 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 device datasheet.

### 8.3.8 External interrupt/wake-up lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode.

Refer to *Section 14: Extended interrupt and event controller (EXTI)* and to *Section 14.3: EXTI functional description.*

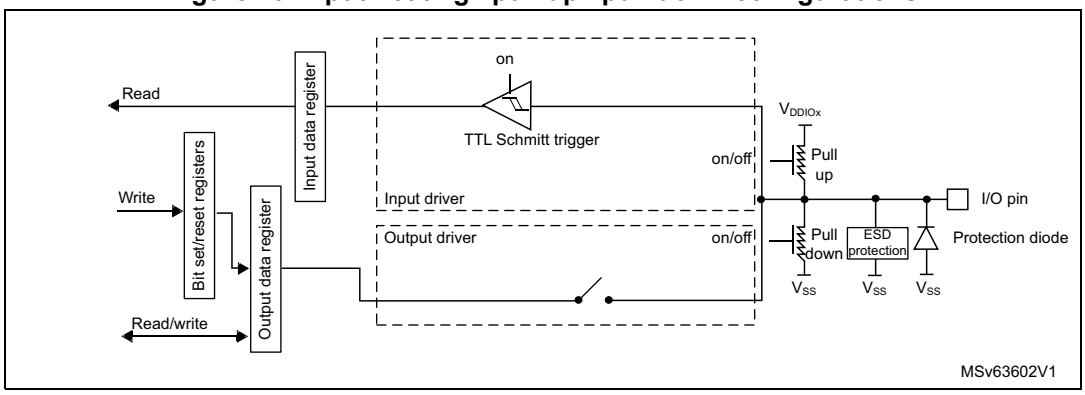
### 8.3.9 Input configuration

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 20* shows the input configuration of the I/O port bit.

**Figure 20. Input floating / pull up / pull down configurations**



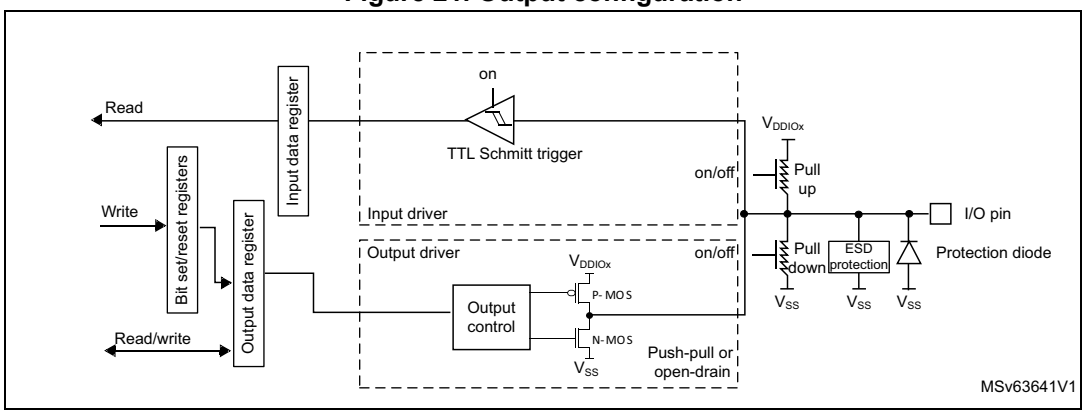

## 8.3.10 Output configuration

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 21* shows the output configuration of the I/O port bit.

**Figure 21. Output configuration**



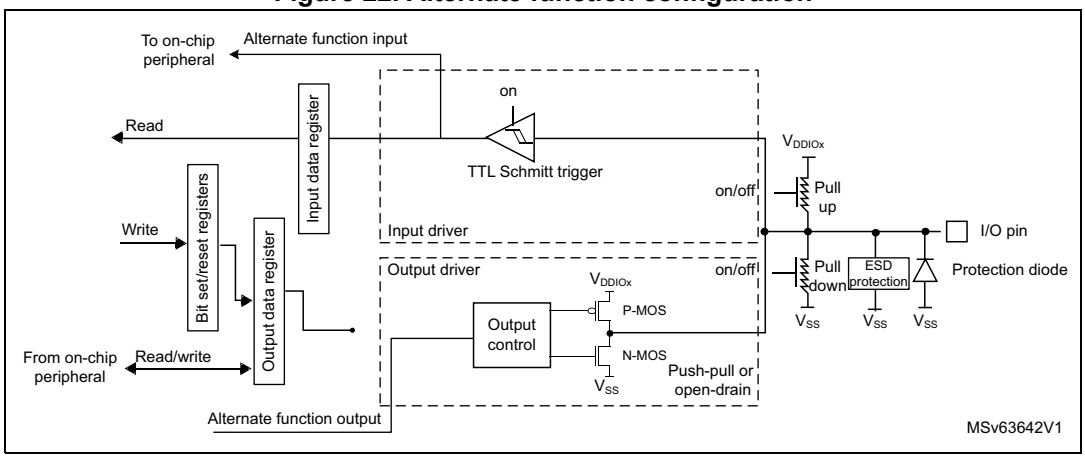
### 8.3.11 Alternate function configuration

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 22* shows the alternate function configuration of the I/O port bit.

**Figure 22. Alternate function configuration**



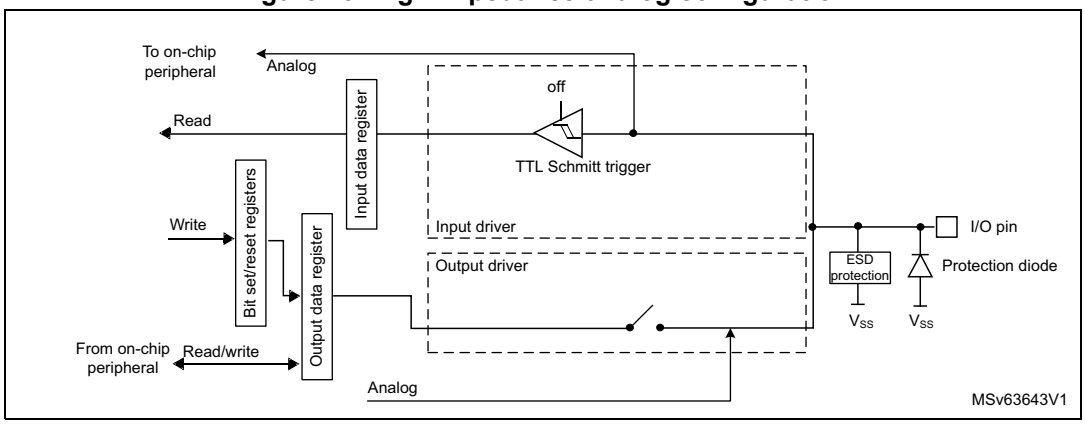

### 8.3.12 Analog configuration

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 weak pull-up and pull-down resistors are disabled by hardware
- Read access to the input data register gets the value “0”

*Figure 23* shows the high-impedance, analog-input configuration of the I/O port bits.

**Figure 23. High impedance-analog configuration**




## 8.3.13 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 LSE oscillator is switched ON (by setting the LSEON bit in the RCC_CSR register) the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

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.

*Note: The HSE OSC_IN and OSC_OUT pins are dedicated oscillator pins and cannot be used as GPIO.*

## 8.3.14 Using the GPIO pins in the RTC supply domain

The PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to *Section 24.3: RTC functional description*.

## 8.3.15 Using PH3 as GPIO

PH3 may be used as boot pin (BOOT0) or as a GPIO. Depending on the nSWBOOT0 bit in the user option byte, it switches from the input mode to the analog input mode:

- After the option byte loading phase if nSWBOOT0 = 1.
- After reset if nSWBOOT0 = 0.

## 8.4 GPIO registers

For a summary of register bits, register address offsets and reset values, refer to *Table 39*.

The peripheral registers can be written in word, half word or byte mode.

### 8.4.1 GPIO port mode register (GPIOx_MODER) (x = A to C and E and H)

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFFFF FEBF (for port B)

Reset value: 0xFFFF FFFF for port C, D

Reset value: 0x0000 03FF for port E

Reset value: 0x0000 00CF for port H

Port B[31:20] are reserved

Port C[27:0] are reserved

Port E[31:10, 7:0] are reserved

Port H[31:8, 5:0] are reserved

| 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 31:0 **MODE[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

### 8.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to C and E and H)

Address offset: 0x04

Reset value: 0x0000 0000

Port B[31:10] are reserved

Port C[31:16, 13:0] are reserved

Port E[31:5, 3:0] are reserved

Port H[31:4, 2:0] are reserved

| 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 **OT[15:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

### 8.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to C and E and H)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for other ports)

Port B[31:20] are reserved

Port C[27:0] are reserved

Port E[31:10, 7:0] are reserved

Port H[31:8, 5:0] are reserved

| 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 31:0 **OSPEED[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed
01: Medium speed
10: Fast speed
11: High speed

*Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed..*

### 8.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to C and E and H)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for other ports)

Port B[31:20] are reserved

Port C[27:0] are reserved

Port E[31:10, 7:0] are reserved

Port H[31:8, 5:0] are reserved

| 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 31:0 **PUPD[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)

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

### 8.4.5 GPIO port input data register (GPIOx_IDR) (x = A to C and E and H)

Address offset: 0x10

Reset value: 0x0000 XXXX

Port B[31:10] are reserved

Port C[31:16, 13:0] are reserved

Port E[31:5, 3:0] are reserved

Port H[31:4, 2:0] are reserved

| 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 **ID[15:0]:** Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

### 8.4.6 GPIO port output data register (GPIOx_ODR) (x = A to C and E and H)

Address offset: 0x14

Reset value: 0x0000 0000

Port B[31:10] are reserved

Port C[31:16, 13:0] are reserved

Port E[31:5, 3:0] are reserved

Port H[31:4, 2:0] are reserved

| 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 **OD[15:0]:** Port output data I/O pin y (y = 15 to 0)
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 register (x = A..F).*

### 8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to C and E and H)

Address offset: 0x18

Reset value: 0x0000 0000

Port B[31:26, 15:10] are reserved

Port C[29:16, 13:0] are reserved

Port E[31:21, 19:5, 3:0] are reserved

Port H[31:20, 18:4, 2:0] are reserved

| 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 **BR[15:0]:** Port x reset I/O pin y (y = 15 to 0)

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 **BS[15:0]:** Port x set I/O pin y (y = 15 to 0)

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

### 8.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to C and E and H)

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

Port B[31:17, 15:10] are reserved

Port C[31:17, 13:0] are reserved

Port E[31:17, 15:5, 3:0] are reserved

Port H[31:17, 15:4, 2:0] are reserved

| 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 **LCK[15:0]:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0.

0: Port configuration not locked

1: Port configuration locked

### 8.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to C and E and H)

Address offset: 0x20

Reset value: 0x0000 0000

Port C[31:0] are reserved

Port E[31:20, 15:0] are reserved

Port H[31:16, 11:0] are reserved

| 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 **AFSEL[7:0][3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

### 8.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to C and E and H)

Address offset: 0x24

Reset value: 0x0000 0000

Port B[31:8] are reserved

Port C[23:0] are reserved

Port E[31:0] are reserved

Port H[31:0] are reserved

| 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 **AFSEL[15:8][3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.

0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

### 8.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to C and E and H)

Address offset: 0x28

Reset value: 0x0000 0000

Port B[31:10] are reserved

Port C[31:16, 13:0] are reserved

Port E[31:5, 3:0] are reserved

Port H[31:4, 2:0] are reserved

| 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 Reserved, must be kept at reset value.

Bits 15:0 **BR[15:0]:** Port x reset IO pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODx bit
1: Reset the corresponding ODx bit

### 8.4.12 GPIO register map

The following table gives the GPIO register map and reset values.

**Table 39. GPIO register map and reset values**

| Offset | Register name | 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 | 1 | 0 | 1 | 0 | 1 | 0 | 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 |
| 0x00 | **GPIOB_MODER** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0x00 | **GPIOC_MODER** | MODE15[1:0] | | MODE14[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. |
| | Reset value | 1 | 1 | 1 | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 0x00 | **GPIOE_MODER** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MODE4[1:0] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | - | - | - | - | - | - | - | - |
| 0x00 | **GPIOH_MODER** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MODE3[1:0] | | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | - | - | - | - | - | - |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | OT9 | OT8 | OT7 | OT6 | OT5 | OT4 | OT3 | OT2 | OT1 | OT0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x04 | **GPIOC_OTYPER** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OT15 | OT14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | | | | | | | | |
| 0x04 | **GPIOE_OTYPER** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OT4 | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - | - |
| 0x04 | **GPIOH_OTYPER** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OT3 | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - |

**Table 39. GPIO register map and reset values (continued)**

| Offset | Register name | 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 | **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 | 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 | 0 |
| 0x08 | **GPIOB_OSPEEDR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x08 | **GPIOC_OSPEEDR** | OSPEED15[1:0] | | OSPEED14[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. |
| | Reset value | 0 | 0 | 0 | 0 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 0x08 | **GPIOE_OSPEEDR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OSPEED4[1:0] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | - | - | - | - | - | - | - | - |
| 0x08 | **GPIOH_OSPEEDR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OSPEED3[1:0] | | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | - | - | - | - | - | - |
| 0x0C | **GPIOA_PUPDR** | PUPD15[1:0] | | PUPD14[1:0] | | PUPD13[1:0] | | PUPD12[1:0] | | PUPD11[1:0] | | PUPD10[1:0] | | PUPD9[1:0] | | PUPD8[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] | |
| | Reset value | 0 | 1 | 1 | 0 | 0 | 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 | 0 |
| 0x0C | **GPIOB_PUPDR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PUPD9[1:0] | | PUPD8[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] | |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0C | **GPIOx_PUPDR** (where x = C, D | PUPD15[1:0] | | PUPD14[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. |
| | Reset value | 0 | 0 | 0 | 0 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 0x0C | **GPIOE_PUPDR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PUPD4[1:0] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | - | - | - | - | - | - | - | - |
| 0x0C | **GPIOH_PUPDR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PUPD3[1:0] | | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | - | - | - | - | - | - |

**Table 39. GPIO register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | x | x | x | x | x | x | x | x | x | x |
| 0x10 | **GPIOC_IDR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ID15 | ID14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | x | x | | | | | | | | | | | | | | |
| 0x10 | **GPIOE_IDR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ID4 | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | x | - | - | - | - |
| 0x10 | **GPIOH_IDR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ID3 | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 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 |
| 0x14 | **GPIOB_ODR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OD9 | OD8 | OD7 | OD6 | OD5 | OD4 | OD3 | OD2 | OD1 | OD0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x14 | **GPIOC_ODR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OD15 | OD14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | | | | | | | | |
| 0x14 | **GPIOE_ODR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OD4 | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - | - |
| 0x14 | **GPIOH_ODR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OD3 | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - |
| 0x18 | **GPIOA_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 |
| 0x18 | **GPIOB_BSRR** | Res. | Res. | Res. | Res. | Res. | Res. | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 | Res. | Res. | Res. | Res. | Res. | Res. | 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 |
| 0x18 | **GPIOC_BSRR** | BR15 | BR14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BS15 | BS14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | 0 | 0 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 0x18 | **GPIOE_BSRR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BS4 | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - | - |
| 0x18 | **GPIOH_BSRR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BS3 | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | 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 | Res. | Res. | Res. | Res. | Res. | Res. | LCK9 | LCK8 | LCK7 | LCK6 | LCK5 | LCK4 | LCK3 | LCK2 | LCK1 | LCK0 |
| | Reset value | | | | | | | | | | | | | | | | 0 | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 39. GPIO register map and reset values (continued)**

| Offset | Register name | 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 | **GPIOC_LCKR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LCKK | LCK15 | LCK14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | 0 | 0 | 0 | | | | | | | | | | | | | | |
| 0x1C | **GPIOE_LCKR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LCKK | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LCK4 | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - | - |
| 0x1C | **GPIOH_LCKR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LCKK | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LCK3 | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - | - | - | - | - | - | - | - | - | - | 0 | - | - | - |
| 0x20 | **GPIOx_AFRL** (where x = **A,B**) | 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 | **GPIOE_AFRL** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | AFSEL4 [3:0] | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | 0 | 0 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 0x20 | **GPIOH_AFRL** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | AFSEL3 [3:0] | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | 0 | 0 | - | - | - | - | - | - | - | - | - | - | - | - |
| 0x24 | **GPIOA_AFRH** | 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** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | AFSEL9 [3:0] | | | | AFSEL8 [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x24 | **GPIOC_AFRH** | AFSEL15 [3:0] | | | | AFSEL14 [3:0] | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | 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. | Res. | Res. | Res. | Res. | Res. | Res. | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x28 | **GPIOC_BRR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BR15 | BR14 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | | | | | | | | |
| 0x28 | **GPIOE_BRR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BR4 | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | |
| 0x28 | **GPIOH_BRR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BR3 | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 9 System configuration controller (SYSCFG)

## 9.1 SYSCFG main features

The devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Remapping memory areas
- Managing the external interrupt line connection to the GPIOs
- Managing robustness feature
- Setting SRAM2 and PKA RAM write protection and software erase
- Configuring FPU interrupts
- Enabling /disabling $I^2C$ Fast-mode Plus driving capability on some I/Os and voltage booster for I/Os analog switches
- Interrupt pre-masking

## 9.2 SYSCFG registers

### 9.2.1 SYSCFG memory remap register (SYSCFG_MEMRMP)

This register is used for specific configurations on memory remap.

Address offset: 0x000

Reset value: 0x0000 000X (X is the memory mode selected by the BOOT0 pin and BOOT1 option bit)

| 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. | MEM_MODE[2:0] | | |
| | | | | | | | | | | | | | rw | rw | rw |

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

Bits 2:0 **MEM_MODE[2:0]:** Memory mapping selection

These bits control the memory internal mapping at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT mode setting. After reset these bits take the value selected by BOOT0 (pin or option bit depending on nSWBOOT0 option bit) and BOOT1 option bit.

000: Main Flash memory mapped at CPU1 0x00000000
001: System Flash memory mapped at CPU1 0x00000000
010: Reserved
011: SRAM1 mapped at CPU1 0x00000000
100: Reserved
101: Reserved
110: Reserved
111: Reserved

### 9.2.2 SYSCFG configuration register 1 (SYSCFG_CFGR1)

Address offset: 0x004

Reset value: 0x7C00 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FPU_IE[5:0] | | | | | | Res. | Res. | Res. | Res. | Res. | I2C1_FMP | I2C_PB9_FMP | I2C_PB8_FMP | I2C_PB7_FMP | I2C_PB6_FMP |
| 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. | Res. | Res. | Res. | Res. | BOOSTEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | rw | | | | | | | | |

Bits 31:26 **FPU_IE[5:0]**: CPU1 FPU interrupts enable bits
FPU_IE[5]: Inexact interrupt enable
FPU_IE[4]: Input denormal interrupt enable
FPU_IE[3]: Overflow interrupt enable
FPU_IE[2]: underflow interrupt enable
FPU_IE[1]: Divide-by-zero interrupt enable
FPU_IE[0]: Invalid operation interrupt enable

Bits 25:21 Reserved, must be kept at reset value.

Bit 20 **I2C1_FMP:** I2C1 Fast-mode Plus driving capability activation
This bit enables the Fm+ driving mode on I2C1 pins selected through AF selection bits.
0: Fm+ mode is not enabled on I2C1 pins selected through AF selection bits
1: Fm+ mode is enabled on I2C1 pins selected through AF selection bits.

Bit 19 **I2C_PB9_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB9
This bit enables the Fm+ driving mode for PB9.
0: PB9 pin operates in standard mode.
1: Fm+ mode enabled on PB9 pin, and the Speed control is bypassed.

Bit 18 **I2C_PB8_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB8
This bit enables the Fm+ driving mode for PB8.
0: PB8 pin operates in standard mode.
1: Fm+ mode enabled on PB8 pin, and the Speed control is bypassed.

Bit 17 **I2C_PB7_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB7
This bit enables the Fm+ driving mode for PB7.
0: PB7 pin operates in standard mode.
1: Fm+ mode enabled on PB7 pin, and the Speed control is bypassed.

Bit 16 **I2C_PB6_FMP**: Fast-mode Plus (Fm+) driving capability activation on PB6
This bit enables the Fm+ driving mode for PB6.
0: PB6 pin operates in standard mode.
1: Fm+ mode enabled on PB6 pin, and the Speed control is bypassed.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BOOSTEN:** I/O analog switch voltage booster enable

0: I/O analog switches are supplied by $V_{DDA}$ voltage. This is the recommended configuration when using the ADC in high $V_{DDA}$ voltage operation.

1: I/O analog switches are supplied by a dedicated voltage booster (supplied by $V_{DD}$). This is the recommended configuration when using the ADC in low $V_{DDA}$ voltage operation.

Bits 7:0 Reserved, must be kept at reset value.

### 9.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x008

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. | EXTI3[2:0] | | | Res. | EXTI2[2:0] | | | Res. | EXTI1[2:0] | | | Res. | EXTI0[2:0] | | |
| | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | | rw | rw | rw |

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

Bits 14:12 **EXTI3[2:0]**: EXTI 3 configuration bits

These bits are written by software to select the source input for the EXTI3 external interrupt.

000: PA[3] pin
001: PB[3] pin
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: PH[3] pin

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI2[2:0]**: EXTI 2 configuration bits

These bits are written by software to select the source input for the EXTI2 external interrupt.

000: PA[2] pin
001: PB[2] pin
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI1[2:0]**: EXTI 1 configuration bits

These bits are written by software to select the source input for the EXTI1 external interrupt.

000: PA[1] pin
001: PB[1] pin
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI0[2:0]**: EXTI 0 configuration bits

These bits are written by software to select the source input for the EXTI0 external interrupt.

000: PA[0] pin
001: PB[0] pin
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

*Note: Some of the I/O pins mentioned in this register may be not available on small packages.*

### 9.2.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

Address offset: 0x00C

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. | EXTI7[2:0] | | | Res. | EXTI6[2:0] | | | Res. | EXTI5[2:0] | | | Res. | EXTI4[2:0] | | |
| | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | | rw | rw | rw |

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

Bits 14:12 **EXTI7[2:0]**: EXTI 7 configuration bits

These bits are written by software to select the source input for the EXTI7 external interrupt.

000: PA[7] pin
001: PB[7] pin
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI6[2:0]**: EXTI 6 configuration bits

These bits are written by software to select the source input for the EXTI6 external interrupt.
000: PA[6] pin
001: PB[6] pin
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI5[2:0]**: EXTI 5 configuration bits

These bits are written by software to select the source input for the EXTI5 external interrupt.
000: PA[5] pin
001: PB[5] pin
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI4[2:0]**: EXTI 4 configuration bits

These bits are written by software to select the source input for the EXTI4 external interrupt.
000: PA[4] pin
001: PB[4] pin
010: Reserved
011: Reserved
100: PE[4] pin
101: Reserved
110: Reserved
111: Reserved

*Note: Some of the I/O pins mentioned in this register may be not available on small packages.*

### 9.2.5 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)

Address offset: 0x010

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. | EXTI11[2:0] | | | Res. | EXTI10[2:0] | | | Res. | EXTI9[2:0] | | | Res. | EXTI8[2:0] | | |
| | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | | rw | rw | rw |

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

Bits 14:12 **EXTI11[2:0]**: EXTI 11 configuration bits

These bits are written by software to select the source input for the EXTI11 external interrupt.

000: PA[11] pin
001: Reserved
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI10[2:0]**: EXTI 10 configuration bits

These bits are written by software to select the source input for the EXTI10 external interrupt.

000: PA[10] pin
001: Reserved
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI9[2:0]**: EXTI 9 configuration bits

These bits are written by software to select the source input for the EXTI9 external interrupt.

000: PA[9] pin
001: PB[9] pin
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI8[2:0]**: EXTI 8 configuration bits

These bits are written by software to select the source input for the EXTI8 external interrupt.

000: PA[8] pin
001: PB[8] pin
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

*Note: Some of the I/O pins mentioned in this register may be not available on small packages.*

### 9.2.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)

Address offset: 0x014

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. | EXTI15[2:0] | | | Res. | EXTI14[2:0] | | | Res. | EXTI13[2:0] | | | Res. | EXTI12[2:0] | | |
| | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | | rw | rw | rw |

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

Bits 14:12 **EXTI15[2:0]**: EXTI15 configuration bits

These bits are written by software to select the source input for the EXTI15 external interrupt.

000: PA[15] pin
001: Reserved
010: PC[15] pin
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI14[2:0]**: EXTI14 configuration bits

These bits are written by software to select the source input for the EXTI14 external interrupt.

000: PA[14] pin
001: Reserved
010: PC[14] pin
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI13[2:0]**: EXTI13 configuration bits

These bits are written by software to select the source input for the EXTI13 external interrupt.

000: PA[13] pin
001: Reserved
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI12[2:0]**: EXTI12 configuration bits

These bits are written by software to select the source input for the EXTI12 external interrupt.

000: PA[12] pin
001: Reserved
010: Reserved
011: Reserved
100: Reserved
101: Reserved
110: Reserved
111: Reserved

*Note: Some of the I/O pins mentioned in this register may be not available on small packages.*

### 9.2.7 SYSCFG SRAM2 control and status register (SYSCFG_SCSR)

Address offset: 0x18

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C2RFD | 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. | Res. | Res. | SRAM2 BSY | SRAM2 ER |
| | | | | | | | | | | | | | | r | rw |

Bit 31 **C2RFD**: CPU2 SRAM fetch (execution) disable.

This bit can be set by software and be reset by a hardware reset, including a reset after Standby. Software writing 'b0 has no effect.

0: CPU2 fetch from SRAM1, SRAM2a and SRAM2b enabled, allows CPU2 to fetch and execute code from SRAMs

1: CPU2 fetch from SRAM1, SRAM2a and SRAM2b disabled, Any CPU2 fetch from SRAMs generates a bus error.

Bits 30:2 Reserved, must be kept at reset value.

Bit 1 **SRAM2BSY:** SRAM2 and PKA RAM busy by erase operation

0: Nor SRAM2 neither PKA RAM erase operation is on going.

1: SRAM2 and/or PKA RAM erase operation is on going.

Bit 0 **SRAM2ER:** SRAM2 and PKA RAM Erase

Setting this bit starts a hardware SRAM2 and PKA RAM erase operation. This bit is automatically cleared at the end of the SRAM2 and PKA RAM erase operation.

*Note: This bit is write-protected: setting this bit is possible only after the correct key sequence is written in the SYSCFG_SKR register.*

### 9.2.8 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x01C

System 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. | SPF | Res. | Res. | Res. | Res. | ECCL | PVDL | SPL | CLL |
| | | | | | | | rc_w1 | | | | | rs | rs | rs | rs |

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

Bit 8 **SPF**: SRAM2 parity error flag

This bit is set by hardware when an SRAM2 parity error is detected. It is cleared by software by writing ‘1’.

0: No SRAM2 parity error detected

1: SRAM2 parity error detected

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

Bit 3 **ECCL**: ECC Lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Flash ECC error connection to TIM1/16/17 Break input.

0: ECC error disconnected from TIM1/16/17 Break input.

1: ECC error connected to TIM1/16/17 Break input.

Bit 2 **PVDL**: PVD lock enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/16/17 Break input, as well as the PVDE and PLS[2:0] in the PWR_CR2 register.

0: PVD interrupt disconnected from TIM1/16/17 Break input. PVDE and PLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/16/17 Break input, PVDE and PLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM2 parity lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM2 parity error signal connection to TIM1/16/17 Break inputs.

0: SRAM2 parity error signal disconnected from TIM1/16/17 Break inputs

1: SRAM2 parity error signal connected to TIM1/16/17 Break inputs

Bit 0 **CLL:** CPU1 LOCKUP (Hardfault) output enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the connection of CPU1 LOCKUP (Hardfault) output to TIM1/16/17 Break input

0: CPU1 LOCKUP output disconnected from TIM1/16/17 Break inputs

1: CPU1 LOCKUP output connected to TIM1/16/17 Break inputs

### 9.2.9 SYSCFG SRAM2 write protection register (SYSCFG_SWPR1)

Address offset: 0x020

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P31WP | P30WP | P29WP | P28WP | P27WP | P26WP | P25WP | P24WP | P23WP | P22WP | P21WP | P20WP | P19WP | P18WP | P17WP | P16WP |
| rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P15WP | P14WP | P13WP | P12WP | P11WP | P10WP | P9WP | P8WP | P7WP | P6WP | P5WP | P4WP | P3WP | P2WP | P1WP | P0WP |
| rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs |

Bits 31:0 **PxWP** (x= 0 to 31): SRAM2 1Kbyte page x write protection

These bits are set by software and cleared only by a system reset.

0: Write protection of SRAM2 1Kbyte page x is disabled.

1: Write protection of SRAM2 1Kbyte page x is enabled.

### 9.2.10 SYSCFG SRAM2 key register (SYSCFG_SKR)

Address offset: 0x024

System 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]**: SRAM2 write protection key for software erase

The following steps are required to unlock the write protection of the SRAM2ER bit in the SYSCFG_CFGR2 register.

1. Write 0xCA into Key[7:0]
2. Write 0x53 into Key[7:0]

Writing a wrong key reactivates the write protection.

### 9.2.11 SYSCFG SRAM2 write protection register 2 (SYSCFG_SWPR2)

Address offset: 0x028

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P63WP | P62WP | P61WP | P60WP | P59WP | P58WP | P57WP | P56WP | P55WP | P54WP | P53WP | P52WP | P51WP | P50WP | P49WP | P48WP |
| rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P47WP | P46WP | P45WP | P44WP | P43WP | P42WP | P41WP | P40WP | P39WP | P38WP | P37WP | P36WP | P35WP | P34WP | P33WP | P32WP |
| rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs |

Bits 31:0 **PxWP** (x= 32 to 63): SRAM2 1 Kbyte page x write protection
These bits are set by software and cleared only by a system reset.
0: Write protection of SRAM2 1 Kbyte page x is disabled.
1: Write protection of SRAM2 1 Kbyte page x is enabled.

## 9.2.12 SYSCFG CPU1 interrupt mask register 1 (SYSCFG_IMR1)

Address offset: 0x100

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTI15 IM | EXTI14 IM | EXTI13 IM | EXTI12 IM | EXTI11I M | EXTI10 IM | EXTI9 IM | EXTI8 IM | EXTI7 IM | EXTI6 IM | EXTI5 IM | Res. | Res. | Res. | Res. | Res. |
| 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 |
| TIM17 IM | TIM16 IM | TIM1 IM | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| rw | rw | rw | | | | | | | | | | | | | |

Bits 31:21 **xxxIM**: Peripheral xxx interrupt mask to CPU1
0: Peripheral xxx interrupt forwarded to CPU1
1. Peripheral xxx interrupt to CPU1 masked.

Bits 20:16 Reserved, must be kept at reset value.

Bits 15:13 **xxxIM**: Peripheral xxx interrupt mask to CPU1
0: Peripheral xxx interrupt forwarded to CPU1
1. Peripheral xxx interrupt to CPU1 masked.

Bits 12:0 Reserved, must be kept at reset value.

## 9.2.13 SYSCFG CPU1 interrupt mask register 2 (SYSCFG_IMR2)

Address offset: 0x104

System 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. | PVDIM | 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. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |

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

Bit 20 **xxxIM**: Peripheral xxx interrupt mask to CPU1
0: Peripheral xxx interrupt forwarded to CPU1
1. Peripheral xxx interrupt to CPU1 masked.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:0 Reserved, must be kept at reset value.

### 9.2.14 SYSCFG CPU2 interrupt mask register 1 (SYSCFG_C2IMR1)

Address offset: 0x108

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTI15 IM | EXTI14 IM | EXTI13 IM | EXTI12 IM | EXTI11 IM | EXTI10 IM | EXTI9 IM | EXTI8 IM | EXTI7 IM | EXTI6 IM | EXTI5 IM | EXTI4 IM | EXTI3 IM | EXTI2 IM | EXTI1 IM | EXTI0 IM |
| 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 |
| Res. | Res. | Res. | ADC IM | Res. | Res. | RNG IM | PKA IM | Res. | FLASH IM | RCC IM | RTC ALARM IM | RTC WKUP IM | Res. | Res. | RTC STAMP TAMP LSECSS IM |
| | | | rw | | | rw | rw | | rw | rw | rw | rw | | | rw |

Bits 31:16 **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1. Peripheral xxx interrupt to CPU2 masked.

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

Bit 12 **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1. Peripheral xxx interrupt to CPU2 masked.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1. Peripheral xxx interrupt to CPU2 masked.

Bit 7 Reserved, must be kept at reset value.

Bits 6:3 **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1. Peripheral xxx interrupt to CPU2 masked.

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

Bit 0 **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1. Peripheral xxx interrupt to CPU2 masked.

### 9.2.15 SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2)

Address offset: 0x10C

System 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. | PVD IM | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMA MUX1 IM | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DMA1 CH7 IM | DMA1 CH6 IM | DMA1 CH5 IM | DMA1 CH4 IM | DMA1 CH3 IM | DMA1 CH2 IM | DMA1 CH1 IM |
| rw | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

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

Bit:20 **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1. Peripheral xxx interrupt to CPU2 masked.

Bits 19:16 Reserved, must be kept at reset value.

Bit15 **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1. Peripheral xxx interrupt to CPU2 masked.

Bits 14:7 Reserved, must be kept at reset value.

Bits 6:0 **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1. Peripheral xxx interrupt to CPU2 masked.

### 9.2.16 SYSCFG secure IP control register (SYSCFG_SIPCR)

Address offset: 0x110

System reset value: 0x0000 0000

This register provides write access security and can only be written by the CPU2. A write access from the CPU1 is ignored and a bus error is generated. On any read access the register value is returned.

| 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. | SRNG | SPKA | SAES2 | Res. |
| | | | | | | | | | | | | rw | rw | rw | |

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

Bit 3 **SRNG**: Enable true RNG security
0: True RNG security disabled (RNG registers can be accessed by both CPU1 and CPU2)
1: True RNG security enabled (RNG registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return 0 value).

Bit 2 **SPKA**: Enable PKA security

0: PKA security disabled (PKA registers can be accessed by both CPU1 and CPU2)

1: PKA security enabled. PKA registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return 0 value. When this bis is set and a system reset occurs (including a reset due to wakeup from Standby) the PKA RAM is erased.

Bit 1 **SAES2**: Enable AES2 security

0: AES2 security disabled (AES2 registers can be accessed by both CPU1 and CPU2)

1: AES2 security enabled. (AES2 registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return 0 value).

Bit 0 Reserved, must be kept at reset value.

## 9.2.17 SYSCFG register map

The following table summarizes the SYSCFG register map and the reset values.

**Table 40. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000 | **SYSCFG_ MEMRMP** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MEM_MODE [2:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | x | x | x |
| 0x004 | **SYSCFG_CFGR1** | FPU_IE[5..0] | | | | | | Res. | Res. | Res. | Res. | Res. | I2C1_FMP | I2C_PB9_FMP | I2C_PB8_FMP | I2C_PB7_FMP | I2C_PB6_FMP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BOOSTEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | 0 | 1 | 1 | 1 | 1 | 1 | | | | | | 0 | 0 | 0 | 0 | 0 | | | | | | | | 0 | | | | | | | | |
| 0x008 | **SYSCFG_EXTICR1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EXTI3 [2:0] | | | Res. | EXTI2 [2:0] | | | Res. | EXTI1 [2:0] | | | Res. | EXTI0 [2:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 |
| 0x00C | **SYSCFG_EXTICR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EXTI7 [2:0] | | | Res. | EXTI6 [2:0] | | | Res. | EXTI5 [2:0] | | | Res. | EXTI4 [2:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 |
| 0x010 | **SYSCFG_EXTICR3** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EXTI11 [2:0] | | | Res. | EXTI10 [2:0] | | | Res. | EXTI9 [2:0] | | | Res. | EXTI8 [2:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 |
| 0x014 | **SYSCFG_EXTICR4** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EXTI15 [2:0] | | | Res. | EXTI14 [2:0] | | | Res. | EXTI13 [2:0] | | | Res. | EXTI12 [2:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 |
| 0x018 | **SYSCFG_SCSR** | C2RFD | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SRAM2BS | SRAM2ER |
| | Reset value | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 |
| 0x01C | **SYSCFG_CFGR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SPF | Res. | Res. | Res. | Res. | ECCL | PVDL | SPL | CLL |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 |
| 0x020 | **SYSCFG_SWPR** | P31WP | P30WP | P29WP | P28WP | P27WP | P26WP | P25WP | P24WP | P23WP | P22WP | P21WP | P20WP | P19WP | P18WP | P17WP | P16WP | P15WP | P14WP | P13WP | P12WP | P11WP | P10WP | P9WP | P8WP | P7WP | P6WP | P5WP | P4WP | P3WP | P2WP | P1WP | P0WP |
| | 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 | **SYSCFG_SKR** | 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 | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x028 | **SYSCFG_SWPR2** | P63WP | P62WP | P61WP | P60WP | P59WP | P58WP | P57WP | P56WP | P55WP | P54WP | P53WP | P52WP | P51WP | P50WP | P49WP | P48WP | P47WP | P46WP | P45WP | P44WP | P43WP | P42WP | P41WP | P40WP | P39WP | P38WP | P37WP | P36WP | P35WP | P34WP | P33WP | P32WP |
| | 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 40. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x100 | **SYSCFG_IMR1** | EXTI15IM | EXTI14IM | EXTI13IM | EXTI12IM | EXTI11IM | EXTI10IM | EXTI9IM | EXTI8IM | EXTI7IM | EXTI6IM | EXTI5IM | Res. | Res. | Res. | Res. | Res. | TIM17IM | TIM16IM | TIM1IM | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | 0 | 0 | 0 | | | | | | | | | | | | | |
| 0x104 | **SYSCFG_IMR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PVDIM | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | 0 | | | | | | | | | | | | | | | | | | | | |
| 0x108 | **SYSCFG_C2IMR1** | EXTI15IM | EXTI14IM | EXTI13IM | EXTI12IM | EXTI11IM | EXTI10IM | EXTI9IM | EXTI8IM | EXTI7IM | EXTI6IM | EXTI5IM | EXTI4IM | EXTI3IM | EXTI2IM | EXTI1IM | EXTI0IM | Res. | Res. | Res. | ADCIM | Res. | Res. | RNGIM | PKAIM | Res. | FLASHIM | RCCIM | RTCALARMIM | RTCWKUPIM | Res. | Res. | RTCSTAMPTAMPLSECSSIM |
| | 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 |
| 0x10C | **SYSCFG_C2IMR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | Res. | PVDIM | Res. | Res. | Res. | Res. | DMAMUX1IM | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DMA1CH7IM | DMA1CH6IM | DMA1CH5IM | DMA1CH4IM | DMA1CH3IM | DMA1CH2IM | DMA1CH1IM |
| | Reset value | | | | | | | | | | | | 0 | | | | | 0 | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x110 | **SYSCFG_SIPCR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SRNG | SPKA | SAES2 | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 10 Peripherals interconnect matrix

## 10.1 Introduction

Several peripherals have direct connections, enabling autonomous communication and/or synchronization between them. This saves CPU resources and, consequently, reduces power consumption. In addition, these hardware connections remove software latency and result in more predictable system design.

Depending on peripherals, these interconnections can operate in Run, Sleep, Low-power run and sleep, Stop 0, Stop 1 and Stop 2 modes.

## 10.2 Connection summary

**Table 41. Peripherals interconnect matrix(1) (2)**

| Source | Destination | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | TIM1 | TIM2 | TIM16 | TIM17 | LPTIM1 | LPTIM2 | ADC1 | IRTIM |
| TIM1 | - | 1 | - | - | - | - | 2 | - |
| TIM2 | 1 | - | - | - | - | - | 2 | - |
| TIM16 | - | - | - | - | - | - | - | 8 |
| TIM17 | 1 | - | - | - | - | - | - | 8 |
| LPTIM1 | - | - | - | - | - | - | - | - |
| LPTIM2 | - | - | - | - | - | - | - | - |
| ADC1 | 3 | - | - | - | - | - | - | - |
| Temp. sensor | - | - | - | - | - | - | 6 | - |
| VBAT | - | - | - | - | - | - | 6 | - |
| VREFINT | - | - | - | - | - | - | 6 | - |
| HSE | - | - | - | 4 | - | - | - | - |
| LSE | - | 4 | 4 | - | - | - | - | - |
| MSI | - | - | - | 4 | - | - | - | - |
| LSI | - | - | 4 | - | - | - | - | - |
| MCO | - | - | - | 4 | - | - | - | - |
| EXTI | - | - | - | - | - | - | 2 | - |
| RTC | - | - | 4 | - | 5 | 5 | - | - |
| SYST ERR | 7 | - | 7 | 7 | - | - | - | - |

1. Numbers in table are links to corresponding subsections of *Section 10.3: Interconnection details*.
2. The "-" symbol in grayed cells means no interconnect.

## 10.3 Interconnection details

### 10.3.1 From timer (TIM1/TIM2/TIM17) to timer (TIM1/TIM2)

#### Purpose

Some of the timers are linked together internally for 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. A description of the feature is provided in *Section 19.3.26: Timer synchronization*.

The modes of synchronization are detailed in:

- *Section 19.3.26: Timer synchronization* for advanced-control timers (TIM1)
- *Section 20.3.18: Timers and external trigger synchronization* for general-purpose timers (TIM2)

#### Triggering signals

The output (from Master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1) following a configurable timer event. The input (to slave) is on signals TIMx_ITR0/ITR1/ITR2/ITR3.

The input and output signals for TIM1 are shown in *Figure 115: Advanced-control timer block diagram*.

The possible master/slave connections are given in:

- *Table 128: TIM1 internal trigger connection*
- *Table 132: TIM2 internal trigger connection*

#### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.2 From timer (TIM1/TIM2) and EXTI to ADC (ADC1)

#### Purpose

General-purpose timer TIM2, advanced-control timer TIM1 and EXTI can be used to generate an ADC triggering event.

TIMx synchronization is described in *Section 19.3.27: ADC synchronization*.

ADC synchronization is described in *Section 16.4: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)*.

#### Triggering signals

The output (from timer) is on signal TIMx_TRGO, TIMx_TRGO2 or TIMx_CCx event.

The input (to ADC) is on signals EXT[15:0] and JEXT[15:0].

The connection between timers and ADC is provided in:

- *Table 67: ADC1 - External triggers for regular channels*
- *Table 68: ADC1 - External trigger for injected channels*

#### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.3 From ADC (ADC1) to timer (TIM1)

#### Purpose

ADC1 can provide trigger event through watchdog signals to advanced-control timers (TIM1).

A description of the ADC analog watchdog setting is provided in *Section 15.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*.

Trigger settings on the timer are provided in *Section 19.3.4: External trigger input*.

#### Triggering signals

The output (from ADC) is on signals ADC_AWDx_OUT n = 1, 2, 3 (for ADC1) x = 1, 2, 3 (three watchdogs on ADC) and the input (to timer) on signal TIMx_ETR (external trigger).

#### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.4 From HSE, LSE, LSI, MSI, MCO, RTC to timers (TIM2/TIM16/TIM17)

#### Purpose

External clocks (HSE, LSE), internal clocks (LSI, MSI), microcontroller output clock (MCO), GPIO and RTC wakeup interrupt can be used as input to general-purpose timers (TIM16/17) channel 1.

This makes possible calibration of the HSI16/MSI system clocks (with TIM16 and LSE) or of the LSI (with TIM16 and HSE). This is also used to precisely measure LSI (with TIM16 and HSI16) or MSI (with TIM17 and HSI16) oscillator frequency.

When Low Speed External (LSE) oscillator is used, no additional hardware connections are required.

This feature is described in *Section 7.2.20: Internal/external clock measurement with TIM16/TIM17*.

External clock LSE can be used as input to general-purpose timer (TIM2) on TIM2_ETR pin, see *Section 7.2.20: Internal/external clock measurement with TIM16/TIM17*..

#### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.5 From RTC to low-power timers (LPTIM1/LPTIM2)

#### Purpose

RTC alarm A/B, RTC_TAMP2 input detection can be used as trigger to start LPTIM counters (LPTIM1/2).

#### Triggering signals

This trigger feature is described in *Section 22.4.7: Trigger multiplexer* (and following sections).

The input selection is described in *Table 141: LPTIM1 external trigger connection* and *Table 142: LPTIM2 external trigger connection*.

#### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep, Stop 0, Stop 1, Stop 2 (LPTIM1 only).

### 10.3.6 From internal analog to ADC1

#### Purpose

Internal temperature sensor (VTS), internal reference voltage (VREFINT) and VBAT monitoring channel are connected to ADC1 input channel.

This is according to:

- *Section 16.2: ADC main features*
- *Section 16.3.8: Channel selection (CHSEL, SCANDIR, CHSELRMOD)*
- *Section 16.9: Temperature sensor and internal reference voltage*
- *Section 16.10: Battery voltage monitoring*

#### Active power mode(s)

Run, Sleep, low-power run, Low-power sleep.

### 10.3.7 From system errors to timers (TIM1/TIM16/TIM17)

#### Purpose

CSS, CPU hard fault, RAM parity error, FLASH ECC double error detection, PVD can generate system errors in the form of timer break toward timers TIM1/TIM16/TIM17.

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

List(s) of possible break source(s) are described in:

- *Section 19.3.16: Using the break function* (TIM1)
- *Section 21.3.11: Using the break function* (TIM16/TIM17)
- *Figure 224: TIM16/TIM17 block diagram*

#### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

### 10.3.8 From timers (TIM16/TIM17) to IRTIM

#### Purpose

General-purpose timer (TIM16/TIM17) output channel TIMx_OC1 are used to generate the waveform of infrared signal output.

The functionality is described in *Section 23: Infrared interface (IRTIM)*.

#### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

# 11 Direct memory access controller (DMA)

## 11.1 Introduction

The direct memory access (DMA) controller is a bus master and system peripheral.

The DMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.

The DMA controller features a single AHB master architecture.

There is one instance of DMA with 7 channels.

Each channel is dedicated to managing memory access requests from one or more peripherals. The DMA includes an arbiter for handling the priority between DMA requests.

## 11.2 DMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access, as source and destination, to on-chip memory-mapped devices such as flash memory, SRAM, and AHB and APB peripherals
- All DMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (four levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to $2^{16}$ - 1
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.

## 11.3 DMA implementation

### 11.3.1 DMA1

DMA1 is implemented with the hardware configuration parameters shown in the table below.

**Table 42. DMA1 implementation**

| Feature | DMA1 |
|---|---|
| Number of channels | 7 |

### 11.3.2 DMA request mapping

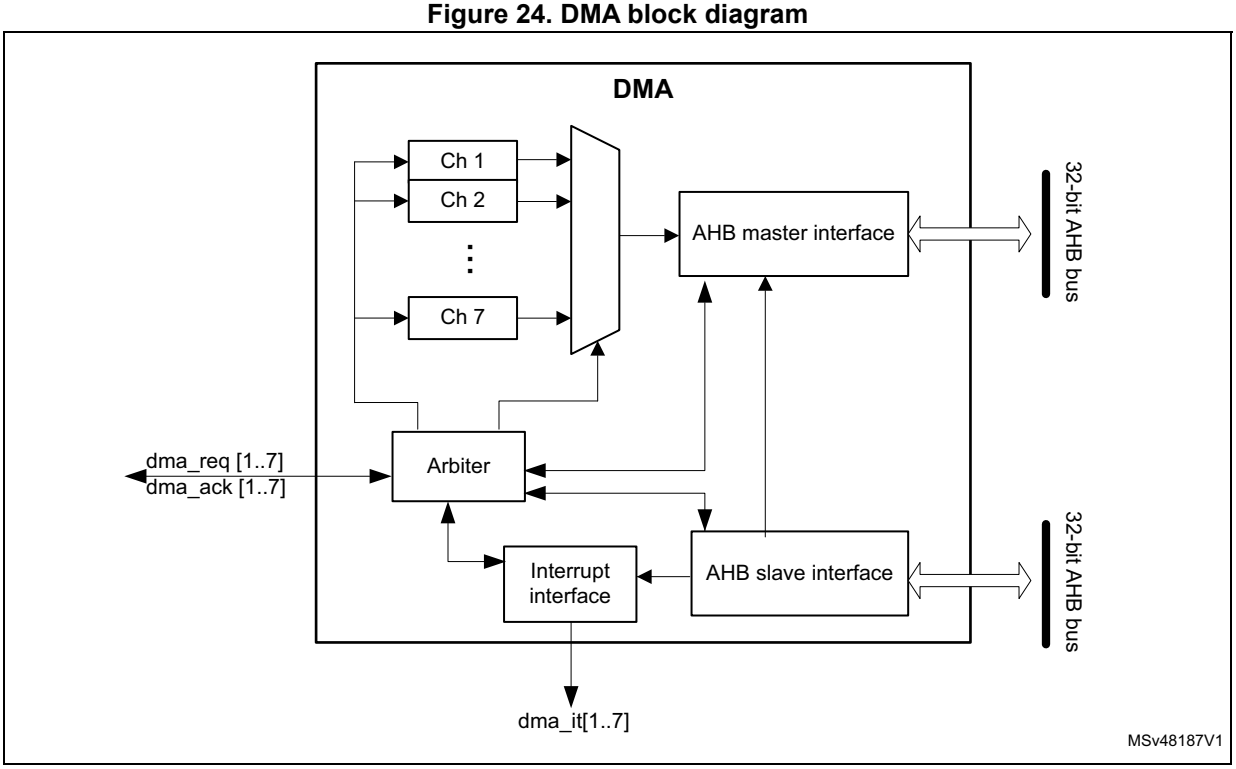
For the mapping of the different requests, refer to the DMAMUX section.

## 11.4 DMA functional description

### 11.4.1 DMA block diagram

The DMA block diagram is shown in the figure below.

**Figure 24. DMA block diagram**




The DMA controller performs direct memory transfer by sharing the AHB system bus with other system masters. The bus matrix implements round-robin scheduling. DMA requests

may stop the CPU access to the system bus for a number of bus cycles, when CPU and DMA target the same destination (memory or peripheral).

According to its configuration through the AHB slave interface, the DMA controller arbitrates between the DMA channels and their associated received requests. The DMA controller also schedules the DMA data transfers over the single AHB port master.

The DMA controller generates an interrupt per channel to the interrupt controller.

### 11.4.2 DMA pins and internal signals

**Table 43. DMA internal input/output signals**

| Signal name | Signal type | Description |
|---|---|---|
| dma_req[x] | Input | DMA channel x request |
| dma_ack[x] | Output | DMA channel x acknowledge |
| dma_it[x] | Output | DMA channel x interrupt |

### 11.4.3 DMA transfers

The software configures the DMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB bus transfers.

A DMA block transfer may be requested from a peripheral, or triggered by the software in case of memory-to-memory transfer.

After an event, the following steps of a single DMA transfer occur:

1. The peripheral sends a single DMA request signal to the DMA controller.
2. The DMA controller serves the request, depending on the priority of the channel associated to this peripheral request.
3. As soon as the DMA controller grants the peripheral, an acknowledge is sent to the peripheral by the DMA controller.
4. The peripheral releases its request as soon as it gets the acknowledge from the DMA controller.
5. Once the request is de-asserted by the peripheral, the DMA controller releases the acknowledge.

The peripheral may order a further single request and initiate another single DMA transfer.

The request/acknowledge protocol is used when a peripheral is either the source or the destination of the transfer. For example, in case of memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the DMA controller. The DMA controller reads then a single data in the memory and writes this data to the peripheral.

For a given channel x, a DMA block transfer consists of a repeated sequence of:

- a single DMA transfer, encapsulating two AHB transfers of a single data, over the DMA AHB bus master:
  - a single data read (byte, half-word, or word) from the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
    The start address used for the first single transfer is the base address of the

peripheral or memory, and is programmed in the DMA_CPARx or DMA_CMARx register.

  – a single data write (byte, half-word, or word) to the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
    The start address used for the first transfer is the base address of the peripheral or memory, and is programmed in the DMA_CPARx or DMA_CMARx register.

- post-decrementing of the programmed DMA_CNDTRx register
  This register contains the remaining number of data items to transfer (number of AHB 'read followed by write' transfers).

This sequence is repeated until DMA_CNDTRx is null.

*Note: The AHB master bus source/destination address must be aligned with the programmed size of the transferred single data to the source/destination.*

### 11.4.4 DMA arbitration

The DMA arbiter manages the priority between the different channels.

When an active channel x is granted by the arbiter (hardware requested or software triggered), a single DMA transfer is issued (such as a AHB 'read followed by write' transfer of a single data). Then, the arbiter considers again the set of active channels and selects the one with the highest priority.

The priorities are managed in two stages:

- software: priority of each channel is configured in the DMA_CCRx register, to one of the four different levels:
  – very high
  – high
  – medium
  – low
- hardware: if two requests have the same software priority level, the channel with the lowest index gets priority. For example, channel 2 gets priority over channel 4.

When a channel x is programmed for a block transfer in memory-to-memory mode, re arbitration is considered between each single DMA transfer of this channel x. Whenever there is another concurrent active requested channel, the DMA arbiter automatically alternates and grants the other highest-priority requested channel, which may be of lower priority than the memory-to-memory channel.

### 11.4.5 DMA channels

Each channel may handle a DMA transfer between a peripheral register located at a fixed address, and a memory address. The amount of data items to transfer is programmable. The register that contains the amount of data items to transfer is decremented after each transfer.

A DMA channel is programmed at block transfer level.

### Programmable data sizes

The transfer sizes of a single data (byte, half-word, or word) to the peripheral and memory are programmable through, respectively, the PSIZE[1:0] and MSIZE[1:0] fields of the DMA_CCRx register.

### Pointer incrementation

The peripheral and memory pointers may be automatically incremented after each transfer, depending on the PINC and MINC bits of the DMA_CCRx register.

If the **incremented mode** is enabled (PINC or MINC set to 1), the address of the next transfer is the address of the previous one incremented by 1, 2 or 4, depending on the data size defined in PSIZE[1:0] or MSIZE[1:0]. The first transfer address is the one programmed in the DMA_CPARx or DMA_CMARx register. During transfers, 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 x is configured in **non-circular mode**, no DMA request is served after the last data transfer (once the number of single data to transfer reaches zero). The DMA channel must be disabled in order to reload a new number of data items into the DMA_CNDTRx register.

*Note: If the channel x 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 data transfer, the DMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the DMA_CPARx and DMA_CMARx registers.

### Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

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, or after the channel is enabled in memory-to-memory mode.
2. Set the memory address in the DMA_CMARx register.
   The data is written to/read from the memory after the peripheral event or after the channel is enabled in memory-to-memory mode.
3. Configure the total number of data to transfer in the DMA_CNDTRx register.
   After each data transfer, this value is decremented.
4. Configure the parameters listed below in the DMA_CCRx register:
   - the channel priority
   - the data transfer direction
   - the circular mode
   - the peripheral and memory incremented mode
   - the peripheral and memory data size
   - the interrupt enable at half and/or full transfer and/or transfer error
5. Activate the channel by setting the EN bit in the DMA_CCRx register.

A channel, as soon as enabled, may serve any DMA request from the peripheral connected to this channel, or may start a memory-to-memory block transfer.

*Note: The two last steps of the channel configuration procedure may be merged into a single access to the DMA_CCRx register, to configure and enable the channel.*

### Channel state and disabling a channel

A channel x in the active state is an enabled channel (read DMA_CCRx.EN = 1). An active channel x is a channel that must have been enabled by the software (DMA_CCRx.EN set to 1) and afterwards with no occurred transfer error (DMA_ISR.TEIFx = 0). In case there is a transfer error, the channel is automatically disabled by hardware (DMA_CCRx.EN = 0).

The three following use cases may happen:

- Suspend and resume a channel

  This corresponds to the two following actions:

  – An active channel is disabled by software (writing DMA_CCRx.EN = 0 whereas DMA_CCRx.EN = 1).

  – The software enables the channel again (DMA_CCRx.EN set to 1) without reconfiguring the other channel registers (such as DMA_CNDTRx, DMA_CPARx and DMA_CMARx).

  This case is not supported by the DMA hardware, that does not guarantee that the remaining data transfers are performed correctly.

- Stop and abort a channel

  If the application does not need any more the channel, this active channel can be disabled by software. The channel is stopped and aborted but the DMA_CNDTRx register content may not correctly reflect the remaining data transfers versus the aborted source and destination buffer/register.

- Abort and restart a channel

  This corresponds to the software sequence: disable an active channel, then reconfigure the channel and enable it again.

  This is supported by the hardware if the following conditions are met:

  – The application guarantees that, when the software is disabling the channel, a DMA data transfer is not occurring at the same time over its master port. For example, the application can first disable the peripheral in DMA mode, in order to ensure that there is no pending hardware DMA request from this peripheral.

  – The software must operate separated write accesses to the same DMA_CCRx register: First disable the channel. Second reconfigure the channel for a next block transfer including the DMA_CCRx if a configuration change is needed. There are read-only DMA_CCRx register fields when DMA_CCRx.EN=1. Finally enable again the channel.

When a channel transfer error occurs, the EN bit of the DMA_CCRx register is cleared by hardware. This EN bit cannot be set again by software to reactivate the channel x, until the TEIFx bit of the DMA_ISR register is set.

### Circular mode (in memory-to-peripheral/peripheral-to-memory transfers)

The circular mode is available to handle circular buffers and continuous data flows (such as ADC scan mode). This feature is enabled using the CIRC bit in the DMA_CCRx register.

*Note: The circular mode must not be used in memory-to-memory mode. Before enabling a channel in circular mode (CIRC = 1), the software must clear the MEM2MEM bit of the DMA_CCRx register. When the circular mode is activated, the amount of data to transfer is*

*automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.*

*In order to stop a circular transfer, the software needs to stop the peripheral from generating DMA requests (such as quit the ADC scan mode), before disabling the DMA channel. The software must explicitly program the DMA_CNDTRx value before starting/enabling a transfer, and after having stopped a circular transfer.*

### Memory-to-memory mode

The DMA channels may operate without being triggered by a request from a peripheral. This mode is called memory-to-memory mode, and is initiated by software.

If the MEM2MEM bit in the DMA_CCRx register is set, the channel, if enabled, initiates transfers. The transfer stops once the DMA_CNDTRx register reaches zero.

*Note: The memory-to-memory mode must not be used in circular mode. Before enabling a channel in memory-to-memory mode (MEM2MEM = 1), the software must clear the CIRC bit of the DMA_CCRx register.*

### Peripheral-to-peripheral mode

Any DMA channel can operate in peripheral-to-peripheral mode:

- when the hardware request from a peripheral is selected to trigger the DMA channel

  This peripheral is the DMA initiator and paces the data transfer from/to this peripheral to/from a register belonging to another memory-mapped peripheral (this one being not configured in DMA mode).
- when no peripheral request is selected and connected to the DMA channel

  The software configures a register-to-register transfer by setting the MEM2MEM bit of the DMA_CCRx register.

### Programming transfer direction, assigning source/destination

The value of the DIR bit of the DMA_CCRx register sets the direction of the transfer, and consequently, it identifies the source and the destination, regardless the source/destination type (peripheral or memory):

- **DIR = 1** defines typically a memory-to-peripheral transfer. More generally, if DIR = 1:
  - The **source** attributes are defined by the DMA_MARx register, the MSIZE[1:0] field, and MINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘memory’ register, field, and bit are used to define the source peripheral in peripheral-to-peripheral mode.
  - The **destination** attributes are defined by the DMA_PARx register, the PSIZE[1:0] field and the PINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field, and bit are used to define the destination memory in memory-to-memory mode.
- **DIR = 0** defines typically a peripheral-to-memory transfer. More generally, if DIR = 0:
  - The **source** attributes are defined by the DMA_PARx register, the PSIZE[1:0] field and the PINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field, and bit are used to define the source memory in memory-to-memory mode
  - The **destination** attributes are defined by the DMA_MARx register, the MSIZE[1:0] field and the MINC bit of the DMA_CCRx register.

Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the destination peripheral in peripheral-to-peripheral mode.

### 11.4.6 DMA data width, alignment, and endianness

When PSIZE[1:0] and MSIZE[1:0] are not equal, the DMA controller performs some data alignments as described in the table below.

**Table 44. Programmable data width and endian behavior (when PINC = MINC = 1)**

| Source port width (MSIZE if DIR = 1, else PSIZE) | Destination port width (PSIZE if DIR = 1, else MSIZE) | Number of data items to transfer (NDT) | Source content: address / data (DMA_CMARx if DIR = 1, else DMA_CPARx) | DMA transfers | Destination content: address / data (DMA_CPARx if DIR = 1, else DMA_CMARx) |
|---|---|---|---|---|---|
| 8 | 8 | 4 | @0x0 / B0<br>@0x1 / B1<br>@0x2 / B2<br>@0x3 / B3 | 1: read B0[7:0] @0x0 then write B0[7:0] @0x0<br>2: read B1[7:0] @0x1 then write B1[7:0] @0x1<br>3: read B2[7:0] @0x2 then write B2[7:0] @0x2<br>4: read B3[7:0] @0x3 then write B3[7:0] @0x3 | @0x0 / B0<br>@0x1 / B1<br>@0x2 / B2<br>@0x3 / B3 |
| 8 | 16 | 4 | @0x0 / B0<br>@0x1 / B1<br>@0x2 / B2<br>@0x3 / B3 | 1: read B0[7:0] @0x0 then write 00B0[15:0] @0x0<br>2: read B1[7:0] @0x1 then write 00B1[15:0] @0x2<br>3: read B2[7:0] @0x2 then write 00B2[15:0] @0x4<br>4: read B3[7:0] @0x3 then write 00B3[15:0] @0x6 | @0x0 / 00B0<br>@0x2 / 00B1<br>@0x4 / 00B2<br>@0x6 / 00B3 |
| 8 | 32 | 4 | @0x0 / B0<br>@0x1 / B1<br>@0x2 / B2<br>@0x3 / B3 | 1: read B0[7:0] @0x0 then write 000000B0[31:0] @0x0<br>2: read B1[7:0] @0x1 then write 000000B1[31:0] @0x4<br>3: read B2[7:0] @0x2 then write 000000B2[31:0] @0x8<br>4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC | @0x0 / 000000B0<br>@0x4 / 000000B1<br>@0x8 / 000000B2<br>@0xC / 000000B3 |
| 16 | 8 | 4 | @0x0 / B1B0<br>@0x2 / B3B2<br>@0x4 / B5B4<br>@0x6 / B7B6 | 1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0<br>2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1<br>3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2<br>4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3 | @0x0 / B0<br>@0x1 / B2<br>@0x2 / B4<br>@0x3 / B6 |
| 16 | 16 | 4 | @0x0 / B1B0<br>@0x2 / B3B2<br>@0x4 / B5B4<br>@0x6 / B7B6 | 1: read B1B0[15:0] @0x0 then write B1B0[15:0] @0x0<br>2: read B3B2[15:0] @0x2 then write B3B2[15:0] @0x2<br>3: read B5B4[15:0] @0x4 then write B5B4[15:0] @0x4<br>4: read B7B6[15:0] @0x6 then write B7B6[15:0] @0x6 | @0x0 / B1B0<br>@0x2 / B3B2<br>@0x4 / B5B4<br>@0x6 / B7B6 |
| 16 | 32 | 4 | @0x0 / B1B0<br>@0x2 / B3B2<br>@0x4 / B5B4<br>@0x6 / B7B6 | 1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0<br>2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4<br>3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8<br>4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC | @0x0 / 0000B1B0<br>@0x4 / 0000B3B2<br>@0x8 / 0000B5B4<br>@0xC / 0000B7B6 |
| 32 | 8 | 4 | @0x0 / B3B2B1B0<br>@0x4 / B7B6B5B4<br>@0x8 / BBBAB9B8<br>@0xC / BFBEBDBC | 1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0<br>2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x1<br>3: read BBBAB9B8[31:0] @0x8 then write B8[7:0] @0x2<br>4: read BFBEBDBC[31:0] @0xC then write BC[7:0] @0x3 | @0x0 / B0<br>@0x1 / B4<br>@0x2 / B8<br>@0x3 / BC |
| 32 | 16 | 4 | @0x0 / B3B2B1B0<br>@0x4 / B7B6B5B4<br>@0x8 / BBBAB9B8<br>@0xC / BFBEBDBC | 1: read B3B2B1B0[31:0] @0x0 then write B1B0[15:0] @0x0<br>2: read B7B6B5B4[31:0] @0x4 then write B5B4[15:0] @0x2<br>3: read BBBAB9B8[31:0] @0x8 then write B9B8[15:0] @0x4<br>4: read BFBEBDBC[31:0] @0xC then write BDBC[15:0] @0x6 | @0x0 / B1B0<br>@0x2 / B5B4<br>@0x4 / B9B8<br>@0x6 / BDBC |
| 32 | 32 | 4 | @0x0 / B3B2B1B0<br>@0x4 / B7B6B5B4<br>@0x8 / BBBAB9B8<br>@0xC / BFBEBDBC | 1: read B3B2B1B0[31:0] @0x0 then write B3B2B1B0[31:0] @0x0<br>2: read B7B6B5B4[31:0] @0x4 then write B7B6B5B4[31:0] @0x4<br>3: read BBBAB9B8[31:0] @0x8 then write BBBAB9B8[31:0] @0x8<br>4: read BFBEBDBC[31:0] @0xC then write BFBEBDBC[31:0] @0xC | @0x0 / B3B2B1B0<br>@0x4 / B7B6B5B4<br>@0x8 / BBBAB9B8<br>@0xC / BFBEBDBC |

### Addressing AHB peripherals not supporting byte/half-word write transfers

When the DMA controller initiates an AHB byte or half-word write transfer, the data are duplicated on the unused lanes of the AHB master 32-bit data bus (HWDATA[31:0]).

When the AHB slave peripheral does not support byte or half-word write transfers and does not generate any error, the DMA controller writes the 32 HWDATA bits as shown in the two examples below:

- To write the half-word 0xABCD, the DMA controller sets the HWDATA bus to 0xABCDABCD with a half-word data size (HSIZE = HalfWord in the AHB master bus).
- To write the byte 0xAB, the DMA controller sets the HWDATA bus to 0xABABABAB with a byte data size (HSIZE = Byte in the AHB master bus).

Assuming the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take into account the HSIZE data, any AHB byte or half-word transfer is changed into a 32-bit APB transfer as described below:

- An AHB byte write transfer of 0xB0 to one of the 0x0, 0x1, 0x2, or 0x3 addresses, is converted to an APB word write transfer of 0xB0B0B0B0 to the 0x0 address.
- An AHB half-word write transfer of 0xB1B0 to the 0x0 or 0x2 addresses is converted to an APB word write transfer of 0xB1B0B1B0 to the 0x0 address.

## 11.4.7 DMA error management

A DMA transfer error is generated when reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or write access, the faulty channel x is automatically disabled through a hardware clear of its EN bit in the corresponding DMA_CCRx register.

The TEIFx bit of the DMA_ISR register is set. An interrupt is then generated if the TEIE bit of the DMA_CCRx register is set.

The EN bit of the DMA_CCRx register cannot be set again by software (channel x reactivated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

When the software is notified with a transfer error over a channel, which involves a peripheral, the software has first to stop this peripheral in DMA mode, in order to disable any pending or future DMA request. Then software may normally reconfigure both DMA and the peripheral in DMA mode for a new transfer.

## 11.5 DMA interrupts

An interrupt can be generated on a half transfer, transfer complete or transfer error for each DMA channel x. Separate interrupt enable bits are available for flexibility.

**Table 45. DMA interrupt requests**

| Interrupt request | Interrupt event | Event flag | Interrupt enable bit |
|---|---|---|---|
| Channel x interrupt | Half transfer on channel x | HTIFx | HTIEx |
| | Transfer complete on channel x | TCIFx | TCIEx |
| | Transfer error on channel x | TEIFx | TEIEx |
| | Half transfer or transfer complete or transfer error on channel x | GIFx | - |

## 11.6 DMA registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The DMA registers have to be accessed by words (32-bit).

### 11.6.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

Every status bit is cleared by hardware when the software sets the corresponding clear bit or the corresponding global clear bit CGIFx, in the DMA_IFCR register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | TEIF7 | HTIF7 | TCIF7 | GIF7 | TEIF6 | HTIF6 | TCIF6 | GIF6 | TEIF5 | HTIF5 | TCIF5 | GIF5 |
| | | | | 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 31:28 Reserved, must be kept at reset value.

Bit 27 **TEIF7**: Transfer error (TE) flag for channel 7
0: No TE event
1: A TE event occurred.

Bit 26 **HTIF7**: Half transfer (HT) flag for channel 7
0: No HT event
1: An HT event occurred.

Bit 25 **TCIF7**: Transfer complete (TC) flag for channel 7
0: No TC event
1: A TC event occurred.

Bit 24 **GIF7**: Global interrupt flag for channel 7
0: No TE, HT, or TC event
1: A TE, HT, or TC event occurred.

Bit 23 **TEIF6**: Transfer error (TE) flag for channel 6
0: No TE event
1: A TE event occurred.

Bit 22 **HTIF6**: Half transfer (HT) flag for channel 6
0: No HT event
1:An HT event occurred.

Bit 21 **TCIF6**: Transfer complete (TC) flag for channel 6
0: No TC event
1: A TC event occurred.

Bit 20 **GIF6**: Global interrupt flag for channel 6
0: No TE, HT, or TC event
1: A TE, HT, or TC event occurred.

Bit 19 **TEIF5**: Transfer error (TE) flag for channel 5
0: No TE event
1: A TE event occurred.

Bit 18 **HTIF5**: Half transfer (HT) flag for channel 5
0: No HT event
1: An HT event occurred.

Bit 17 **TCIF5**: Transfer complete (TC) flag for channel 5
0: No TC event
1: A TC event occurred.

Bit 16 **GIF5**: global interrupt flag for channel 5
0: No TE, HT, or TC event
1: A TE, HT, or TC event occurred.

Bit 15 **TEIF4**: Transfer error (TE) flag for channel 4
0: No TE event
1: A TE event occurred.

Bit 14 **HTIF4**: Half transfer (HT) flag for channel 4
0: No HT event
1: An HT event occurred.

Bit 13 **TCIF4**: Transfer complete (TC) flag for channel 4
0: No TC event
1: A TC event occurred.

Bit 12 **GIF4**: global interrupt flag for channel 4
0: No TE, HT, or TC event
1: A TE, HT, or TC event occurred.

Bit 11 **TEIF3**: Transfer error (TE) flag for channel 3
0: No TE event
1: A TE event occurred.

Bit 10 **HTIF3**: Half transfer (HT) flag for channel 3
0: No HT event
1: An HT event occurred.

Bit 9 **TCIF3**: Transfer complete (TC) flag for channel 3
0: No TC event
1: A TC event occurred.

Bit 8 **GIF3**: Global interrupt flag for channel 3
0: No TE, HT, or TC event
1: A TE, HT, or TC event occurred.

Bit 7 **TEIF2**: Transfer error (TE) flag for channel 2
0: No TE event
1: A TE event occurred.

Bit 6 **HTIF2**: Half transfer (HT) flag for channel 2
0: No HT event
1: An HT event occurred.

Bit 5 **TCIF2**: Transfer complete (TC) flag for channel 2
0: No TC event
1: A TC event occurred.

Bit 4 **GIF2**: Global interrupt flag for channel 2
0: No TE, HT, or TC event
1: A TE, HT, or TC event occurred.

Bit 3 **TEIF1**: Transfer error (TE) flag for channel 1
0: No TE event
1: A TE event occurred.

Bit 2 **HTIF1**: Half transfer (HT) flag for channel 1
0: No HT event
1: An HT event occurred.

Bit 1 **TCIF1**: Transfer complete (TC) flag for channel 1
0: No TC event
1: A TC event occurred.

Bit 0 **GIF1**: Global interrupt flag for channel 1
0: No TE, HT, or TC event
1: A TE, HT, or TC event occurred.

### 11.6.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

Setting the global clear bit CGIFx of the channel x in this DMA_IFCR register, causes the DMA hardware to clear the corresponding GIFx bit and any individual flag among TEIFx, HTIFx, TCIFx, in the DMA_ISR register.

Setting any individual clear bit among CTEIFx, CHTIFx, CTCIFx in this DMA_IFCR register, causes the DMA hardware to clear the corresponding individual flag and the global flag GIFx in the DMA_ISR register, provided that none of the two other individual flags is set.

Writing 0 into any flag clear bit has no effect.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | CTEIF7 | CHTIF7 | CTCIF7 | CGIF7 | CTEIF6 | CHTIF6 | CTCIF6 | CGIF6 | CTEIF5 | CHTIF5 | CTCIF5 | CGIF5 |
| | | | | 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 |
| CTEIF4 | CHTIF4 | CTCIF4 | CGIF4 | CTEIF3 | CHTIF3 | 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.

Bit 27 **CTEIF7**: Transfer error flag clear for channel 7

Bit 26 **CHTIF7**: Half transfer flag clear for channel 7

Bit 25 **CTCIF7**: Transfer complete flag clear for channel 7

Bit 24 **CGIF7**: Global interrupt flag clear for channel 7

Bit 23 **CTEIF6**: Transfer error flag clear for channel 6

Bit 22 **CHTIF6**: Half transfer flag clear for channel 6

Bit 21 **CTCIF6**: Transfer complete flag clear for channel 6

Bit 20 **CGIF6**: Global interrupt flag clear for channel 6

Bit 19 **CTEIF5**: Transfer error flag clear for channel 5

Bit 18 **CHTIF5**: Half transfer flag clear for channel 5

Bit 17 **CTCIF5**: Transfer complete flag clear for channel 5

Bit 16 **CGIF5**: Global interrupt flag clear for channel 5

Bit 15 **CTEIF4**: Transfer error flag clear for channel 4

Bit 14 **CHTIF4**: Half transfer flag clear for channel 4

Bit 13 **CTCIF4**: Transfer complete flag clear for channel 4

Bit 12 **CGIF4**: Global interrupt flag clear for channel 4

Bit 11 **CTEIF3**: Transfer error flag clear for channel 3

Bit 10 **CHTIF3**: Half transfer flag clear for channel 3

Bit 9 **CTCIF3**: Transfer complete flag clear for channel 3

Bit 8 **CGIF3**: Global interrupt flag clear for channel 3

Bit 7 **CTEIF2**: Transfer error flag clear for channel 2

Bit 6 **CHTIF2**: Half transfer flag clear for channel 2

Bit 5 **CTCIF2**: Transfer complete flag clear for channel 2

Bit 4 **CGIF2**: Global interrupt flag clear for channel 2

Bit 3 **CTEIF1**: Transfer error flag clear for channel 1

Bit 2 **CHTIF1**: Half transfer flag clear for channel 1

Bit 1 **CTCIF1**: Transfer complete flag clear for channel 1

Bit 0 **CGIF1**: Global interrupt flag clear for channel 1

### 11.6.3 DMA channel x configuration register (DMA_CCRx)

Address offset: 0x08 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

The register fields/bits MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

| 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

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

Bits 13:12 **PL[1:0]**: Priority level

00: Low

01: Medium

10: High

11: Very high

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

Bits 11:10 **MSIZE[1:0]**: Memory size

Defines the data size of each DMA transfer to the identified memory.

In memory-to-memory mode, this bitfield identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: Reserved

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

Bits 9:8 **PSIZE[1:0]**: Peripheral size

Defines the data size of each DMA transfer to the identified peripheral.

In memory-to-memory mode, this bitfield identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: Reserved

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

Bit 7 **MINC**: Memory increment mode

Defines the increment mode for each DMA transfer to the identified memory.

In memory-to-memory mode, this bit identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this bit identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

Bit 6 **PINC**: Peripheral increment mode

Defines the increment mode for each DMA transfer to the identified peripheral.

n memory-to-memory mode, this bit identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this bit identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

Bit 5 **CIRC**: Circular mode

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

Bit 4 **DIR**: Data transfer direction

This bit must be set only in memory-to-peripheral and peripheral-to-memory modes.

0: Read from peripheral

- Source attributes are defined by PSIZE and PINC, plus the DMA_CPARx register. This is still valid in a memory-to-memory mode.
- Destination attributes are defined by MSIZE and MINC, plus the DMA_CMARx register. This is still valid in a peripheral-to-peripheral mode.

1: Read from memory

- Destination attributes are defined by PSIZE and PINC, plus the DMA_CPARx register. This is still valid in a memory-to-memory mode.
- Source attributes are defined by MSIZE and MINC, plus the DMA_CMARx register. This is still valid in a peripheral-to-peripheral mode.

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

Bit 3 **TEIE**: Transfer error interrupt enable

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

Bit 2 **HTIE**: Half transfer interrupt enable

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

Bit 1 **TCIE**: Transfer complete interrupt enable

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

Bit 0 **EN**: Channel enable

When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software.*

### 11.6.4 DMA channel x number of data to transfer register (DMA_CNDTRx)

Address offset: 0x0C + 0x14 * (x - 1), (x = 1 to 7)

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 (0 to $2^{16}$ - 1)

This bitfield is updated by hardware when the channel is enabled:

- It is decremented after each single DMA ‘read followed by write’ transfer, indicating the remaining amount of data items to transfer.
- It is kept at zero when the programmed amount of data to transfer is reached, if the channel is not in circular mode (CIRC = 0 in the DMA_CCRx register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this bitfield is zero, no transfer can be served whatever the channel status (enabled or not).

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

### 11.6.5 DMA channel x peripheral address register (DMA_CPARx)

Address offset: 0x10 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

| 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

It contains the base address of the peripheral data register from/to which the data is read/written.

When PSIZE[1:0] = 01 (16 bits), bit 0 of PA[31:0] is ignored. Access is automatically aligned to a half-word address.

When PSIZE[1:0] = 10 (32 bits), bits 1 and 0 of PA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this bitfield identifies the memory destination address if DIR = 1 and the memory source address if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral destination address if DIR = 1 and the peripheral source address if DIR = 0.

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

## 11.6.6 DMA channel x memory address register (DMA_CMARx)

Address offset: 0x14 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

| 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]**: Peripheral address

It contains the base address of the memory from/to which the data is read/written.

When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.

When MSIZE[1:0] = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this bitfield identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral source address if DIR = 1 and the peripheral destination address if DIR = 0.

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

## 11.6.7 DMA register map

**Table 46. DMA register map and reset values**

| Offset | Register name | 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 | Res. | Res. | Res. | Res. | 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 |
| 0x004 | DMA_IFCR | Res. | Res. | Res. | Res. | 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 |

**Table 46. DMA register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x008 | DMA_CCR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MEM2MEM | PL[1:0] | | MSIZE[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. | NDTR[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 |
| 0x018 | Reserved | Reserved. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x01C | DMA_CCR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MEM2MEM | PL[1:0] | | MSIZE[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. | NDTR[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 |
| 0x02C | Reserved | Reserved. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x030 | DMA_CCR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MEM2MEM | PL[1:0] | | MSIZE[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. | NDTR[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 |
| 0x040 | Reserved | Reserved. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x044 | DMA_CCR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MEM2MEM | PL[1:0] | | MSIZE[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. | NDTR[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 | 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] | | MSIZE[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 46. DMA register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x05C | DMA_CNDTR5 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | NDTR[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 | 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] | | MSIZE[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. | NDTR[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 | 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] | | MSIZE[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 |
| 0x084 | DMA_CNDTR7 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | NDTR[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 |

Refer to *Section 2.2* for the register boundary addresses.

# 12 DMA request multiplexer (DMAMUX)

## 12.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is deasserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables 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, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in *Section 12.3.1*.

The assignment of DMAMUX request multiplexer inputs to the DMA request lines from peripherals and to the DMAMUX request generator outputs, the assignment of DMAMUX request multiplexer outputs to DMA controller channels, and the assignment of DMAMUX synchronizations and trigger inputs to internal and external signals depend upon product implementation. They are detailed in *Section 12.3.2*.

## 12.2 DMAMUX main features

- 7-channel programmable DMA request line multiplexer output
- 4-channel DMA request generator
- 20 trigger inputs to DMA request generator
- 20 synchronization inputs
- Per DMA request generator channel:
  - DMA request trigger input selector
  - DMA request counter
  - Event overrun flag for selected DMA request trigger input
- Per DMA request line multiplexer channel output:
  - 36 input DMA request lines from peripherals
  - One DMA request line output
  - Synchronization input selector
  - DMA request counter
  - Event overrun flag for selected synchronization input
  - One event output, for DMA request chaining

## 12.3 DMAMUX implementation

### 12.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

**Table 47. DMAMUX instantiation**

| Feature | DMAMUX |
|---|---|
| Number of DMAMUX output request channels | 7 |
| Number of DMAMUX request generator channels | 4 |
| Number of DMAMUX request trigger inputs | 20 |
| Number of DMAMUX synchronization inputs | 20 |
| Number of DMAMUX peripheral request inputs | 36 |

### 12.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

DMAMUX is used with DMA1

- DMAMUX channels 0 to 6 are connected to DMA1 channels 1 to 7

**Table 48. DMAMUX: assignment of multiplexer inputs to resources**

| DMA request MUX input | Resource | DMA request MUX input | Resource | DMA request MUX input | Resource |
|---|---|---|---|---|---|
| 1 | dmamux_req_gen0 | 22 | TIM1_CH2 | 43 | Reserved |
| 2 | dmamux_req_gen1 | 23 | TIM1_CH3 | 44 | Reserved |
| 3 | dmamux_req_gen2 | 24 | TIM1_CH4 | 45 | Reserved |
| 4 | dmamux_req_gen3 | 25 | TIM1_UP | 46 | Reserved |
| 5 | ADC1 | 26 | TIM1_TRIG | 47 | Reserved |
| 6 | SPI1_RX | 27 | TIM1_COM | 48 | Reserved |
| 7 | SPI1_TX | 28 | TIM2_CH1 | 49 | Reserved |
| 8 | Reserved | 29 | TIM2_CH2 | 50 | Reserved |
| 9 | Reserved | 30 | TIM2_CH3 | 51 | Reserved |
| 10 | I2C1_RX | 31 | TIM2_CH4 | 52 | Reserved |
| 11 | I2C1_TX | 32 | TIM2_UP | 53 | Reserved |
| 12 | Reserved | 33 | TIM16_CH1 | 54 | Reserved |
| 13 | Reserved | 34 | TIM16_UP | 55 | Reserved |
| 14 | USART1_RX | 35 | TIM17_CH1 | 56 | Reserved |
| 15 | USART1_TX | 36 | TIM17_UP | 57 | Reserved |
| 16 | Reserved | 37 | Reserved | 58 | Reserved |
| 17 | Reserved | 38 | Reserved | 59 | Reserved |
| 18 | Reserved | 39 | AES2_IN | 60 | Reserved |
| 19 | Reserved | 40 | AES2_OUT | 61 | Reserved |
| 20 | Reserved | 41 | Reserved | 62 | Reserved |
| 21 | TIM1_CH1 | 42 | Reserved | 63 | Reserved |

**Table 49. DMAMUX: assignment of trigger inputs to resources**

| Trigger input | Resource | Trigger input | Resource |
|---|---|---|---|
| 0 | EXTI LINE0 | 16 | dmamux_evt0 |
| 1 | EXTI LINE1 | 17 | dmamux_evt1 |
| 2 | EXTI LINE2 | 18 | LPTIM1_OUT |
| 3 | EXTI LINE3 | 19 | LPTIM2_OUT |
| 4 | EXTI LINE4 | 20 | Reserved |
| 5 | EXTI LINE5 | 21 | Reserved |
| 6 | EXTI LINE6 | 22 | Reserved |
| 7 | EXTI LINE7 | 23 | Reserved |
| 8 | EXTI LINE8 | 24 | Reserved |
| 9 | EXTI LINE9 | 25 | Reserved |
| 10 | EXTI LINE10 | 26 | Reserved |

**Table 49. DMAMUX: assignment of trigger inputs to resources (continued)**

| Trigger input | Resource | Trigger input | Resource |
|---|---|---|---|
| 11 | EXTI LINE11 | 27 | Reserved |
| 12 | EXTI LINE12 | 28 | Reserved |
| 13 | EXTI LINE13 | 29 | Reserved |
| 14 | EXTI LINE14 | 30 | Reserved |
| 15 | EXTI LINE15 | 31 | Reserved |

**Table 50. DMAMUX: assignment of synchronization inputs to resources**

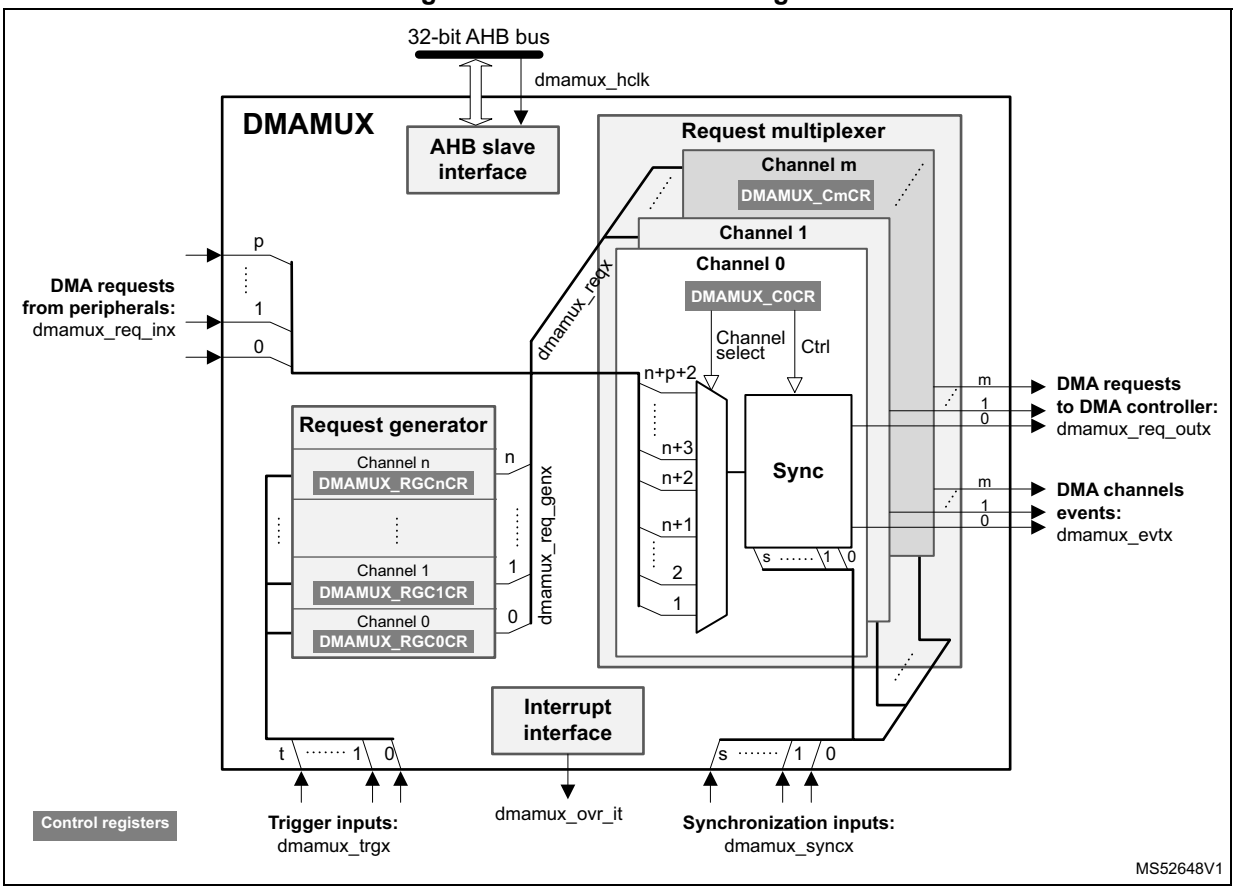
| Sync. input | Resource | Sync. input | Resource |
|---|---|---|---|
| 0 | EXTI LINE0 | 16 | dmamux_evt0 |
| 1 | EXTI LINE1 | 17 | dmamux_evt1 |
| 2 | EXTI LINE2 | 18 | LPTIM1_OUT |
| 3 | EXTI LINE3 | 19 | LPTIM2_OUT |
| 4 | EXTI LINE4 | 20 | Reserved |
| 5 | EXTI LINE5 | 21 | Reserved |
| 6 | EXTI LINE6 | 22 | Reserved |
| 7 | EXTI LINE7 | 23 | Reserved |
| 8 | EXTI LINE8 | 24 | Reserved |
| 9 | EXTI LINE9 | 25 | Reserved |
| 10 | EXTI LINE10 | 26 | Reserved |
| 11 | EXTI LINE11 | 27 | Reserved |
| 12 | EXTI LINE12 | 28 | Reserved |
| 13 | EXTI LINE13 | 29 | Reserved |
| 14 | EXTI LINE14 | 30 | Reserved |
| 15 | EXTI LINE15 | 31 | Reserved |

## 12.4 DMAMUX functional description

### 12.4.1 DMAMUX block diagram

*Figure 25* shows the DMAMUX block diagram.

**Figure 25. DMAMUX block diagram**




DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (dmamux_reqx) from peripherals (dmamux_req_inx) and from channels of the DMAMUX request generator sub-block (dmamux_req_genx)
- DMAMUX request outputs to channels of DMA controllers (dmamux_req_outx)
- Internal or external signals to DMA request trigger inputs (dmamux_trgx)
- Internal or external signals to synchronization inputs (dmamux_syncx)

### 12.4.2 DMAMUX signals

*Table 51* lists the DMAMUX signals.

**Table 51. DMAMUX signals**

| Signal name | Description |
|---|---|
| dmamux_hclk | DMAMUX AHB clock |
| dmamux_req_inx | DMAMUX DMA request line inputs from peripherals |
| dmamux_trgx | DMAMUX DMA request triggers inputs (to request generator sub-block) |
| dmamux_req_genx | DMAMUX request generator sub-block channels outputs |
| dmamux_reqx | DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels) |
| dmamux_syncx | DMAMUX synchronization inputs (to request multiplexer sub-block) |
| dmamux_req_outx | DMAMUX requests outputs (to DMA controller) |
| dmamux_evtx | DMAMUX events outputs |
| dmamux_ovr_it | DMAMUX overrun interrupts |

### 12.4.3 DMAMUX channels

A DMAMUX channel is a request multiplexer channel that can include, depending upon the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to a single channel of DMA controller.

#### Channel configuration procedure

Follow the sequence below to configure a DMAMUX x channel and the related DMA channel y:

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

### 12.4.4 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named 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 either from the peripherals, or from the DMAMUX request generator.

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the DMAREQ_ID field in the DMAMUX_CxCR register.

*Note: The null value in the field DMAREQ_ID corresponds to no DMA request line selected.*

**Caution:** A same non-null DMAREQ_ID cannot be programmed to different x and y DMAMUX request multiplexer channels (via DMAMUX_CxCR and DMAMUX_CyCR), except when the application guarantees that the two connected DMA channels are not simultaneously active.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

### Synchronization mode and channel event generation

Each DMAMUX request line multiplexer channel x can be individually synchronized by setting the synchronization enable (SE) bit in the DMAMUX_CxCR register.

DMAMUX has multiple synchronization inputs. The synchronization inputs are connected in parallel to all the channels of the request multiplexer.

The synchronization input is selected via the SYNC_ID field in the DMAMUX_CxCR register of a given channel x.

When a channel is in this synchronization mode, the selected input DMA request line is propagated to the multiplexer channel output, once a programmable rising/falling edge is detected on the selected input synchronization signal, via the SPOL[1:0] field of the DMAMUX_CxCR register.

Additionally, internally to the DMAMUX request multiplexer, there is a programmable DMA request counter, which can be used for the channel request output generation, and for an event generation. An event generation on the channel x output is enabled through the EGE bit (event generation enable) of the DMAMUX_CxCR register.

As shown in *Figure 27*, upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

*Note: If a synchronization event occurs while there is no pending selected input DMA request line, it is discarded. The following asserted input request lines is not connected to the DMAMUX multiplexer channel output until a synchronization event occurs again.*

From this point on, each time the connected DMAMUX request is served by the DMA controller (a served request is deasserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in the NBREQ field of the DMAMUX_CxCR register and the input DMA request line is disconnected from the multiplexer channel x output.

Thus, the number of DMA requests transferred to the multiplexer channel x output following a detected synchronization event, is equal to the value in the NBREQ field, plus one.

*Note: The NBREQ field value can be written by software only when both synchronization enable bit (SE) and event generation enable bit (EGE) of the corresponding multiplexer channel x are disabled.*

**Figure 26. Synchronization mode of the DMAMUX request line multiplexer channel**

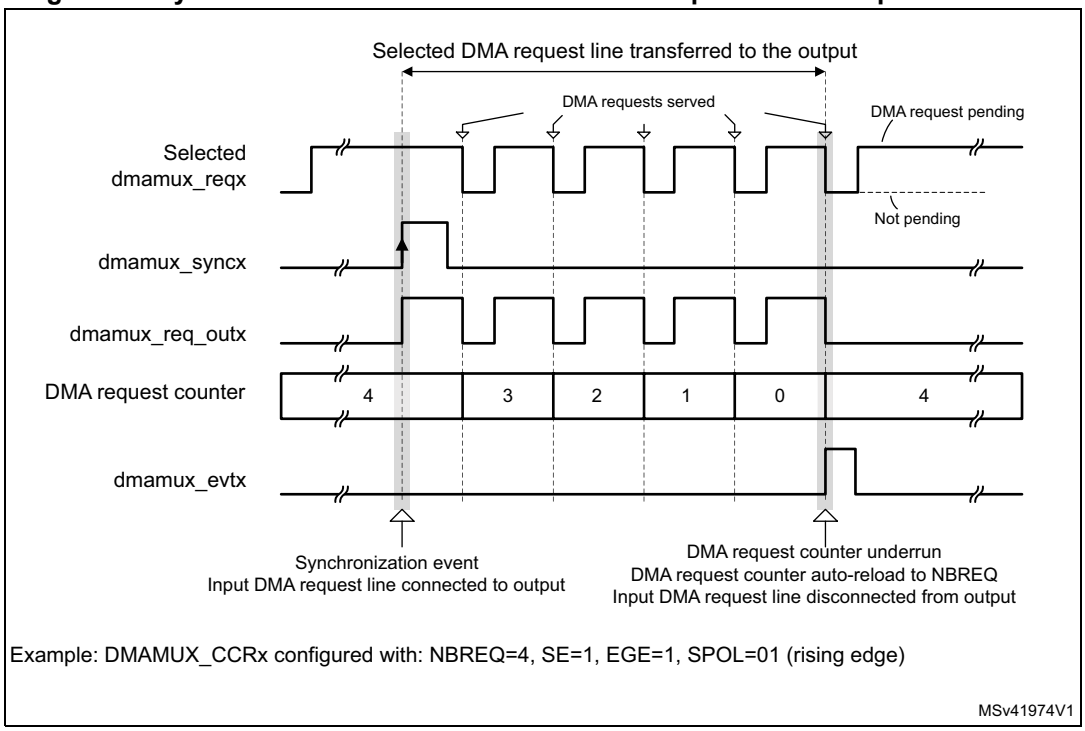
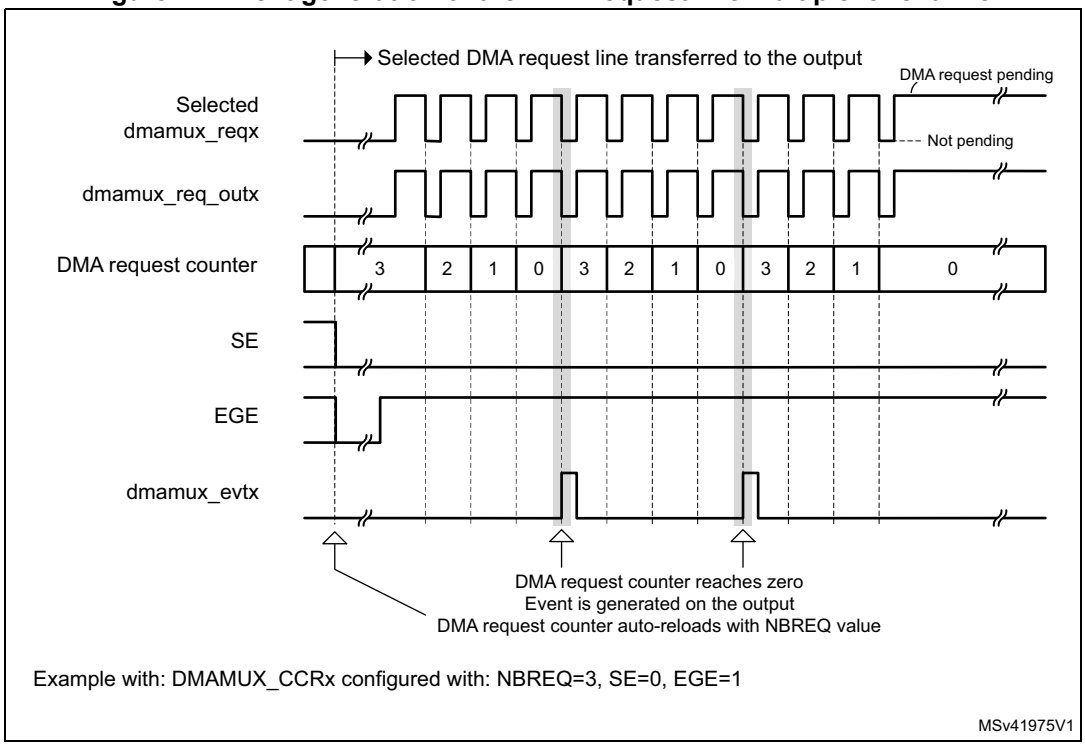



**Figure 27. Event generation of the DMA request line multiplexer channel**




If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in *Figure 26* and *Figure 27*.

*Note: If EGE is enabled and NBREQ = 0, an event is generated after each served DMA request.*

*Note: A synchronization event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.*

*Upon writing into DMAMUX_CxCR register, the synchronization events are masked during three AHB clock cycles.*

#### Synchronization overrun and interrupt

If a new synchronization event occurs before the request counter underrun (the internal request counter programmed via the NBREQ field of the DMAMUX_CxCR register), the synchronization overrun flag bit SOFx is set in the DMAMUX_CSR register.

*Note: The request multiplexer channel x synchronization must be disabled (DMAMUX_CxCR.SE = 0) when the use of the related channel of the DMA controller is completed. Else, upon a new detected synchronization event, there is a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.*

The overrun flag SOFx is reset by setting the associated clear synchronization overrun flag bit CSOFx in the DMAMUX_CFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX_CxCR register.

### 12.4.5 DMAMUX request generator

The DMAMUX request generator produces DMA requests following trigger events on its DMA request trigger inputs.

The DMAMUX request generator has multiple channels. DMA request trigger inputs are connected in parallel to all channels.

The outputs of DMAMUX request generator channels are inputs to the DMAMUX request line multiplexer.

Each DMAMUX request generator channel x has an enable bit GE (generator enable) in the corresponding DMAMUX_RGxCR register.

The DMA request trigger input for the DMAMUX request generator channel x is selected through the SIG_ID (trigger signal ID) field in the corresponding DMAMUX_RGxCR register.

Trigger events on a DMA request trigger input can be rising edge, falling edge or either edge. The active edge is selected through the GPOL (generator polarity) field in the corresponding DMAMUX_RGxCR register.

Upon the trigger event, the corresponding generator channel starts generating DMA requests on its output. Each time the DMAMUX generated request is served by the connected DMA controller (a served request is deasserted), a built-in (inside the DMAMUX request generator) DMA request counter is decremented. At its underrun, the request generator channel stops generating DMA requests and the DMA request counter is automatically reloaded to its programmed value upon the next trigger event.

Thus, the number of DMA requests generated after the trigger event is GNBREQ + 1.

*Note: The GNBREQ field value can be written by software only when the enable GE bit of the corresponding generator channel x is disabled.*

*There is no hardware write protection.*

*A trigger event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.*

*Upon writing into DMAMUX_RGxCR register, the trigger events are masked during three AHB clock cycles.*

### Trigger overrun and interrupt

If a new DMA request trigger event occurs before the DMAMUX request generator counter underrun (the internal counter programmed via the GNBREQ field of the DMAMUX_RGxCR register), and if the request generator channel x was enabled via GE, then the request trigger event overrun flag bit OFx is asserted by the hardware in the DMAMUX_RGSR register.

*Note: The request generator channel x must be disabled (DMAMUX_RGxCR.GE = 0) when the usage of the related channel of the DMA controller is completed. Else, upon a new detected trigger event, there is a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.*

The overrun flag OFx is reset by setting the associated clear overrun flag bit COFx in the DMAMUX_RGCFR register.

Setting the DMAMUX request trigger overrun flag generates an interrupt if the DMA request trigger event overrun interrupt enable bit OIE is set in the DMAMUX_RGxCR register.

## 12.5 DMAMUX interrupts

An interrupt can be generated upon:

- a synchronization event overrun in each DMA request line multiplexer channel
- a trigger event overrun in each DMA request generator channel

For each case, per-channel in.dividual interrupt enable, status, and clear flag register bits are available.

**Table 52. DMAMUX interrupts**

| Interrupt signal | Interrupt event | Event flag | Clear bit | Enable bit |
|---|---|---|---|---|
| dmamuxovr_it | Synchronization event overrun on channel x of the DMAMUX request line multiplexer | SOFx | CSOFx | SOIE |
| | Trigger event overrun on channel x of the DMAMUX request generator | OFx | COFx | OIE |

## 12.6 DMAMUX registers

Refer to the table containing register boundary addresses for the DMAMUX base address.

DMAMUX registers may be accessed per byte (8-bit), half-word (16-bit), or word (32-bit). The address must be aligned with the data size.

### 12.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)

Address offset: 0x000 + 0x04 * x (x = 0 to 6)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | SYNC_ID[4:0] | | | | | NBREQ[4:0] | | | | | SPOL[1:0] | | SE |
| | | | 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. | Res. | Res. | Res. | EGE | SOIE | Res. | Res. | DMAREQ_ID[5:0] | | | | | |
| | | | | | | rw | rw | | | rw | rw | rw | rw | rw | rw |

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SYNC_ID[4:0]**: Synchronization identification

Selects the synchronization input (see *Table 50: DMAMUX: assignment of synchronization inputs to resources*).

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward

Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.
This field must only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity

Defines the edge polarity of the selected synchronization input:
00: No event (no synchronization, no detection).
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 16 **SE**: Synchronization enable

0: Synchronization disabled
1: Synchronization enabled

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

Bit 9 **EGE**: Event generation enable

0: Event generation disabled
1: Event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable

0: Interrupt disabled
1: Interrupt enabled

Bits 76 Reserved, must be kept at reset value.

Bits 5:0 **DMAREQ_ID[5:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

### 12.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR)

Address offset: 0x080

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. | SOF6 | SOF5 | SOF4 | SOF3 | SOF2 | SOF1 | SOF0 |
| | | | | | | | | | r | r | r | r | r | r | r |

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

Bits 6:0 **SOF[6:0]**: Synchronization overrun event flag

The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.

The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX_CFR register.

### 12.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR)

Address offset: 0x084

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. | CSOF 6 | CSOF 5 | CSOF 4 | CSOF 3 | CSOF 2 | CSOF 1 | CSOF 0 |
| | | | | | | | | | w | w | w | w | w | w | w |

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

Bits 6:0 **CSOF[6:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX_CSR register.

### 12.6.4 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)

Address offset: 0x100 + 0x04 * x (x = 0 to 3)

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. | GNBREQ[4:0] | | | | | GPOL[1:0] | | GE |
| | | | | | | | | 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. | Res. | Res. | Res. | OIE | Res. | Res. | Res. | SIG_ID[4:0] | | | | |
| | | | | | | | rw | | | | rw | rw | rw | rw | rw |

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

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ +1.

*Note: This field must be written only when GE bit is disabled.*

Bits 18:17 **GPOL[1:0]**: DMA request generator trigger polarity

Defines the edge polarity of the selected trigger input
00: No event, i.e. no trigger detection nor generation.
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 16 **GE**: DMA request generator channel x enable

0: DMA request generator channel x disabled
1: DMA request generator channel x enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **OIE**: Trigger overrun interrupt enable

0: Interrupt on a trigger overrun event occurrence is disabled
1: Interrupt on a trigger overrun event occurrence is enabled

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

Bits 4:0 **SIG_ID[:0]**: Signal identification

Selects the DMA request trigger input used for the channel x of the DMA request generator

### 12.6.5 DMAMUX request generator interrupt status register (DMAMUX_RGSR)

Address offset: 0x140

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. | OF3 | OF2 | OF1 | OF0 |
| | | | | | | | | | | | | r | r | r | r |

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

Bits 3:0 **OF[3:0]**: Trigger overrun event flag

The flag is set when a new trigger event occurs on DMA request generator channel x, before the request counter underrun (the internal request counter programmed via the GNBREQ field of the DMAMUX_RGxCR register).

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX_RGCFR register.

### 12.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)

Address offset: 0x144

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. | COF3 | COF2 | COF1 | COF0 |
| | | | | | | | | | | | | w | w | w | w |

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

Bits 3:0 **COF[3:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX_RGSR register.

### 12.6.7 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 53. 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. | SYNC_ID[4:0] | | | | | NBREQ[4:0] | | | | | SPOL | | SE | Res. | Res. | Res. | Res. | Res. | Res. | EGE | SOIE | Res. | Res. | DMAREQ_ID[5:0] | | | | | |
| | Reset value | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x004 | DMAMUX_C1CR | Res. | Res. | Res. | SYNC_ID[4:0] | | | | | NBREQ[4:0] | | | | | SPOL | | SE | Res. | Res. | Res. | Res. | Res. | Res. | EGE | SOIE | Res. | Res. | DMAREQ_ID[5:0] | | | | | |
| | Reset value | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x008 | DMAMUX_C2CR | Res. | Res. | Res. | SYNC_ID[4:0] | | | | | NBREQ[4:0] | | | | | SPOL | | SE | Res. | Res. | Res. | Res. | Res. | Res. | EGE | SOIE | Res. | Res. | DMAREQ_ID[5:0] | | | | | |
| | Reset value | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00C | DMAMUX_C3CR | Res. | Res. | Res. | SYNC_ID[4:0] | | | | | NBREQ[4:0] | | | | | SPOL | | SE | Res. | Res. | Res. | Res. | Res. | Res. | EGE | SOIE | Res. | Res. | DMAREQ_ID[5:0] | | | | | |
| | Reset value | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x010 | DMAMUX_C4CR | Res. | Res. | Res. | SYNC_ID[4:0] | | | | | NBREQ[4:0] | | | | | SPOL | | SE | Res. | Res. | Res. | Res. | Res. | Res. | EGE | SOIE | Res. | Res. | DMAREQ_ID[5:0] | | | | | |
| | Reset value | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x014 | DMAMUX_C5CR | Res. | Res. | Res. | SYNC_ID[4:0] | | | | | NBREQ[4:0] | | | | | SPOL | | SE | Res. | Res. | Res. | Res. | Res. | Res. | EGE | SOIE | Res. | Res. | DMAREQ_ID[5:0] | | | | | |
| | Reset value | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x018 | DMAMUX_C6CR | Res. | Res. | Res. | SYNC_ID[4:0] | | | | | NBREQ[4:0] | | | | | SPOL | | SE | Res. | Res. | Res. | Res. | Res. | Res. | EGE | SOIE | Res. | Res. | DMAREQ_ID[5:0] | | | | | |
| | Reset value | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x01C - 0x07C | 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. |
| 0x080 | DMAMUX_CSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SOF6 | SOF5 | SOF4 | SOF3 | SOF2 | SOF1 | SOF0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x084 | DMAMUX_CFR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CSOF6 | CSOF5 | CSOF4 | CSOF3 | CSOF2 | CSOF1 | CSOF0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x088 - 0x0FC | 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. |
| 0x100 | DMAMUX_RG0CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | GNBREQ[4:0] | | | | | GPOL | | GE | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OIE | Res. | Res. | Res. | SIG_ID[4:0] | | | | |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | 0 | | | | 0 | 0 | 0 | 0 | 0 |
| 0x104 | DMAMUX_RG1CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | GNBREQ[4:0] | | | | | GPOL | | GE | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OIE | Res. | Res. | Res. | SIG_ID[4:0] | | | | |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | 0 | | | | 0 | 0 | 0 | 0 | 0 |
| 0x108 | DMAMUX_RG2CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | GNBREQ[4:0] | | | | | GPOL | | GE | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OIE | Res. | Res. | Res. | SIG_ID[4:0] | | | | |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | 0 | | | | 0 | 0 | 0 | 0 | 0 |
| 0x10C | DMAMUX_RG3CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | GNBREQ[4:0] | | | | | GPOL | | GE | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OIE | Res. | Res. | Res. | SIG_ID[4:0] | | | | |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | 0 | | | | 0 | 0 | 0 | 0 | 0 |
| 0x110 - 0x13C | 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. |
| 0x140 | DMAMUX_RGSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OF3 | OF2 | OF1 | OF0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |

**Table 53. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x144 | DMAMUX_RGCFR | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | COF3 | COF2 | COF1 | COF0 |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x148 - 0x3FC | 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 on page 56* for the register boundary addresses.

# 13 Nested vectored interrupt controller (NVIC)

## 13.1 NVIC main features

The CPU1 NVIC features:

- 63 maskable interrupt channels (not including the sixteen Cortex®-M4 with FPU interrupt lines)
- 16 programmable priority levels (four bits of interrupt priority are used)
- Low-latency exception interrupt handling
- Power management control
- Implementation of System control registers

The CPU2 NVIC features:

- 32 maskable interrupt channels (not including the sixteen Cortex®-M0+ interrupt lines)
- Four programmable priority levels (two bits of interrupt priority are used)
- Low-latency exception interrupt handling
- Power management control

The NVICs and the processor cores interfaces are closely coupled, resulting in low latency interrupt processing and efficient processing of late arriving interrupts.

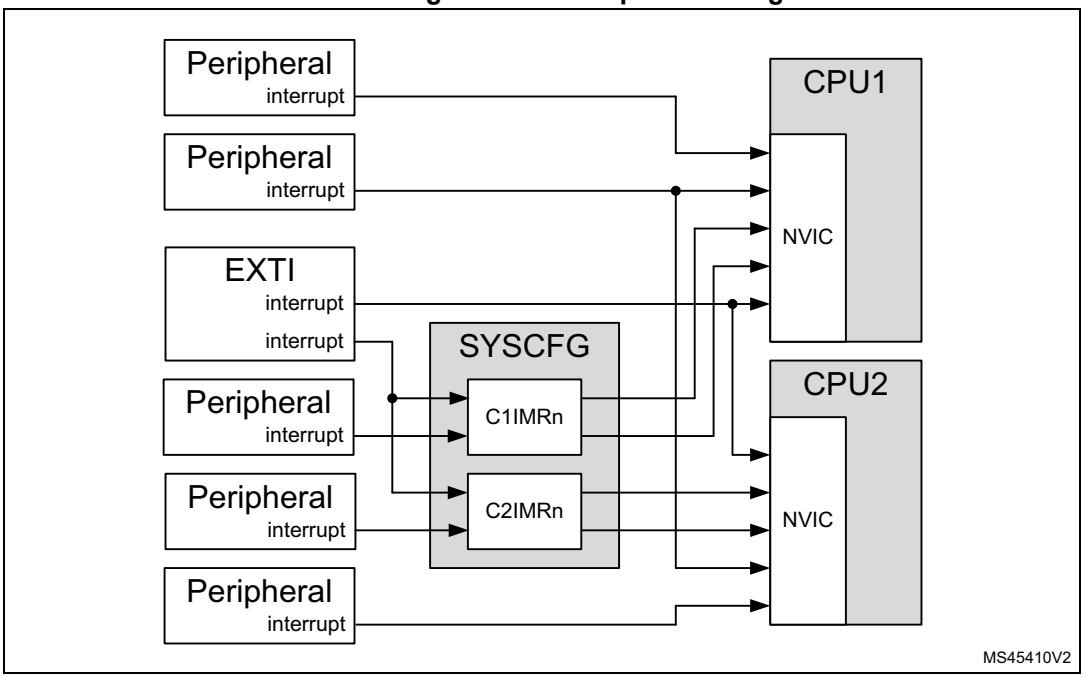
All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to the PM0214 programming manual for Cortex®-M4, and PM0223 programming manual for Cortex®-M0+.

## 13.2 Interrupt block diagram

The different peripheral interrupts are connected in different ways, depending on the sharing between the two CPUs. To prevent a peripheral or EXTI interrupt to trigger both CPUs, they can be masked either in the NVIC, or, for the NVIC vector sharing multiple peripheral interrupts, by a pre-mask in the SYSCFG registers, see *Section 9: System configuration controller (SYSCFG)*.

The interrupt block diagram is shown in *Figure 28*.

Figure 28. Interrupt block diagram




## 13.3 Interrupt and exception vectors

Each of the CPU1 and CPU2 has its own vector table, see, respectively, *Table 54* and *Table 55*, where shaded cells indicate the processor exceptions.

**Table 54. CPU1 vector table**

| Position | Priority | Type of priority | Acronym | Description | Address |
|---|---|---|---|---|---|
| - | - | - | - | Reserved | 0x0000 0000 |
| - | -3 | Fixed | Reset | Reset | 0x0000 0004 |
| - | -2 | Fixed | NMI | Non maskable interrupt HSE CSS, Flash ECC, and SRAM2 parity | 0x0000 0008 |
| - | -1 | Fixed | HardFault | All classes of fault | 0x0000 000C |
| - | 0 | Settable | MemManager | Memory manager | 0x0000 0010 |
| - | 1 | Settable | BusFault | Pre-fetch fault, memory access fault | 0x0000 0014 |
| - | 2 | Settable | UsageFault | Undefined instruction or illegal state | 0x0000 0018 |
| - | - | - | - | Reserved | 0x0000 001C<br>0x0000 0028 |
| - | 3 | Settable | SVCall | System service can via SWI instruction | 0x0000 002C |
| - | 4 | Settable | Debug | Debug monitor | 0x0000 0030 |
| - | - | - | - | Reserved | 0x0000 0034 |
| - | 5 | Settable | PendSV | Pendable request for system service | 0x0000 0038 |
| - | 6 | Settable | Systick | System tick timer | 0x0000 003C |
| 0 | 7 | Settable | WWDG | Window watchdog early wakeup | 0x0000 0040 |
| 1 | 8 | Settable | PVD | PVD through EXTI[16] (C1IMR2[20]) | 0x0000 0044 |
| 2 | 9 | Settable | TAMP, RTC_STAMP, LSE_CSS | Tamper, TimeStamp, LSECSS interrupt through EXTI[18] | 0x0000 0048 |
| 3 | 10 | Settable | RTC_WKUP | RTC wakeup interrupt through EXTI[19] | 0x0000 004C |
| 4 | 11 | Settable | Flash | Flash memory global interrupt and Flash memory ECC single error interrupt | 0x0000 0050 |
| 5 | 12 | Settable | RCC | RCC global interrupt | 0x0000 0054 |
| 6 | 13 | Settable | EXTI0 | EXTI Line 0 interrupt through EXTI[0] | 0x0000 0058 |
| 7 | 14 | Settable | EXTI1 | EXTI Line 1 interrupt through EXTI[1] | 0x0000 005C |
| 8 | 15 | Settable | EXTI2 | EXTI Line 2 interrupt through EXTI[2] | 0x0000 0060 |
| 9 | 16 | Settable | EXTI3 | EXTI Line 3 interrupt through EXTI[3] | 0x0000 0064 |
| 10 | 17 | Settable | EXTI4 | EXTI Line 4 interrupt through EXTI[4] | 0x0000 0068 |
| 11 | 18 | Settable | DMA1_CH1 | DMA1 channel 1 interrupt | 0x0000 006C |
| 12 | 19 | Settable | DMA1_CH2 | DMA1 channel 2 interrupt | 0x0000 0070 |
| 13 | 20 | Settable | DMA1_CH3 | DMA1 channel 3 interrupt | 0x0000 0074 |
| 14 | 21 | Settable | DMA1_CH4 | DMA1 channel 4 interrupt | 0x0000 0078 |
| 15 | 22 | Settable | DMA1_CH5 | DMA1 channel 5 interrupt | 0x0000 007C |
| 16 | 23 | Settable | DMA1_CH6 | DMA1 channel 6 interrupt | 0x0000 0080 |
| 17 | 24 | Settable | DMA1_CH7 | DMA1 channel 7 interrupt | 0x0000 0084 |

**Table 54. CPU1 vector table (continued)**

| Position | Priority | Type of priority | Acronym | Description | Address |
|---|---|---|---|---|---|
| 18 | 25 | Settable | ADC1 | ADC1 global interrupt | 0x0000 0088 |
| 19 | 26 | - | - | Reserved | 0x0000 008C |
| 20 | 27 | - | - | Reserved | 0x0000 0090 |
| 21 | 28 | Settable | C2SEV<br>PWR_C2H | CPU2 SEV through EXTI[40]<br>PWR CPU2 HOLD wakeup interrupt | 0x0000 0094 |
| 22 | 29 | - | - | Reserved | 0x0000 0098 |
| 23 | 30 | Settable | EXTI[9:5] | EXTI Line [9:5] interrupt through EXTI[9:5] (C1IMR1[25:21]) | 0x0000 009C |
| 24 | 31 | Settable | TIM1_BRK | Timer 1 break interrupt | 0x0000 00A0 |
| 25 | 32 | Settable | TIM1_UP,<br>TIM16 | Timer 1 Update (C1IMR1[13])<br>Timer 16 global interrupt (C1IMR1[14]) | 0x0000 00A4 |
| 26 | 33 | Settable | TIM1_TRG_COM,<br>TIM17 | Timer 1 trigger and communication (C1IMR1[13])<br>Timer 17 global interrupt (C1IMR1[15]) | 0x0000 00A8 |
| 27 | 34 | Settable | TIM1_CC | Timer 1 capture compare interrupt | 0x0000 00AC |
| 28 | 35 | Settable | TIM2 | Timer 2 global interrupt | 0x0000 00B0 |
| 29 | 36 | Settable | PKA | Private key accelerator interrupt | 0x0000 00B4 |
| 30 | 37 | Settable | I2C1_EV | I2C1 event interrupt | 0x0000 00B8 |
| 31 | 38 | Settable | I2C1_ER | I2C1 error interrupt | 0x0000 00BC |
| 32 | 39 | - | - | Reserved | 0x0000 00C0 |
| 33 | 40 | - | - | Reserved | 0x0000 00C4 |
| 34 | 41 | Settable | SPI1 | SPI 1 global interrupt | 0x0000 00C8 |
| 35 | 42 | - | - | Reserved | 0x0000 00CC |
| 36 | 43 | Settable | USART1 | USART1 global interrupt | 0x0000 00D0 |
| 37 | 44 | - | - | Reserved | 0x0000 00D4 |
| 38 | 45 | - | - | Reserved | 0x0000 00D8 |
| 39 | 46 | - | - | Reserved | 0x0000 00DC |
| 40 | 47 | Settable | EXTI[15:10] | EXTI Line [15:10] interrupt through EXTI[15:10] (C1IMR1[31:26]) | 0x0000 00E0 |
| 41 | 48 | Settable | RTC_ALARM | RTC alarms (A and B) interrupt through EXTI[17] | 0x0000 00E4 |
| 42 | 49 | - | - | Reserved | 0x0000 00E8 |
| 43 | 50 | Settable | PWR_SOTF<br>PWR_BLEACT<br>PWR_802ACT<br>PWR_RFPHASE | PWR switching on the fly interrupt<br>PWR end of BLE activity interrupt<br>PWR end of 802.15.4 activity interrupt<br>PWR end of critical radio phase interrupt | 0x0000 00EC |
| 44 | 51 | Settable | IPCC_C1_RX_IT | IPCC CPU1 RX occupied interrupt | 0x0000 00F0 |

**Table 54. CPU1 vector table (continued)**

| Position | Priority | Type of priority | Acronym | Description | Address |
|---|---|---|---|---|---|
| 45 | 52 | Settable | IPCC_C1_TX_IT | IPCC CPU1 TX free interrupt | 0x0000 00F4 |
| 46 | 53 | Settable | HSEM | Semaphore interrupt 0 to CPU1 | 0x0000 00F8 |
| 47 | 54 | Settable | LPTIM1 | LPtimer 1 global interrupt | 0x0000 00FC |
| 48 | 55 | Settable | LPTIM2 | LPtimer 2 global interrupt | 0x0000 0100 |
| 49 | 56 | - | - | Reserved | 0x0000 0104 |
| 50 | 57 | - | - | Reserved | 0x0000 0108 |
| 51 | 58 | - | - | Reserved | 0x0000 010C |
| 52 | 59 | Settable | AES2 | AES2 global interrupt | 0x0000 0110 |
| 53 | 60 | Settable | True RNG | True random number generator interrupt | 0x0000 0114 |
| 54 | 61 | Settable | FPU | Floating point unit interrupt | 0x0000 0118 |
| 55 | 62 | - | - | Reserved | 0x0000 011C |
| 56 | 63 | - | - | Reserved | 0x0000 0120 |
| 57 | 64 | - | - | Reserved | 0x0000 0124 |
| 58 | 65 | - | - | Reserved | 0x0000 0128 |
| 59 | 66 | - | - | Reserved | 0x0000 012C |
| 60 | 67 | - | - | Reserved | 0x0000 0130 |
| 61 | 68 | - | - | Reserved | 0x0000 0134 |
| 62 | 69 | Settable | DMAMUX1_OVR | DMAMUX1 overrun interrupt | 0x0000 0138 |

**Table 55. CPU2 vector table**

| Position | Priority | Type of priority | Acronym | Description | Address |
|---|---|---|---|---|---|
| - | - | - | - | Reserved | 0x0000 0000 |
| - | -3 | Fixed | Reset | Reset | 0x0000 0004 |
| -14 | -2 | Fixed | NMI | Non maskable interrupt HSE CSS, Flash ECC, and SRAM2 parity | 0x0000 0008 |
| -13 | -1 | Fixed | HardFault | All classes of fault | 0x0000 000C |
| - | - | - | - | Reserved | 0x0000 0010<br>0x0000 0028 |
| -5 | 0 | Settable | SVCall | System service can via SWI instruction | 0x0000 002C |
| - | - | - | - | Reserved | 0x0000 0030<br>0x0000 0034 |
| -2 | 1 | Settable | PendSV | Pendable request for system service | 0x0000 0038 |
| -1 | 2 | Settable | Systick | System tick timer | 0x0000 003C |
| 0 | 3 | Settable | - | Reserved | 0x0000 0040 |
| 1 | 4 | Settable | PVD | PVD through EXTI[16] (C2IMR2[20]) | 0x0000 0044 |
| 2 | 5 | Settable | RTC_WKUP, TAMP, RTC_STAMP<br>LSE_CSS,<br>RTC_ALARM | RTC wakeup interrupt through EXTI[19] (C2IMR1[3])<br>Tamper, TimeStamp<br>LSECSS interrupt through EXTI[18] (C2IMR1[0])<br>RTC Alarms (A and B) interrupt through EXTI[17] (C2IMR1[4]) | 0x0000 0048 |
| 3 | 6 | - | - | Reserved | 0x0000 004C |
| 4 | 7 | Settable | RCC<br>FLASH<br><br>C1SEV | RCC global interrupt (C2IMR1[5])<br>Flash memory global interrupt and Flash memory ECC single error interrupt (C2IMR1[6])<br>CPU1 SEV through EXTI[41] | 0x0000 0050 |
| 5 | 8 | Settable | EXTI[1:0] | EXTI Line 1:0 interrupt through EXTI[1:0] (C2IMR1[17:16]) | 0x0000 0054 |
| 6 | 9 | Settable | EXTI[3:2] | EXTI Line 3:2 interrupt through EXTI[3:2] (C2IMR1[19:18]) | 0x0000 0058 |
| 7 | 10 | Settable | EXTI[15:4] | EXTI Line 15:4 interrupt through EXTI[15:4] (C2IMR1[31:20]) | 0x0000 005C |
| 8 | 11 | Settable | 802_IT0 | 802.15.4 interrupt 0 | 0x0000 0060 |
| 9 | 12 | Settable | DMA1_CH[3:1] | DMA1 channel 3:1 interrupt (C2IMR2[2:0]) | 0x0000 0064 |
| 10 | 13 | Settable | DMA1_CH[7:4] | DMA1 channel 7:4 interrupt (C2IMR2[6:3]) | 0x0000 0068 |
| 11 | 14 | Settable | DMAMUX1_OVR | DMAMUX1 overrun interrupt (C2IMR2[15]) | 0x0000 006C |
| 12 | 15 | Settable | ADC1 | ADC1 global interrupt (C2IMR1[12]) | 0x0000 0070 |
| 13 | 16 | Settable | LPTIM1 | LPtimer 1 global interrupt | 0x0000 0074 |
| 14 | 17 | Settable | LPTIM2 | LPtimer 2 global interrupt | 0x0000 0078 |

**Table 55. CPU2 vector table (continued)**

| Position | Priority | Type of priority | Acronym | Description | Address |
|---|---|---|---|---|---|
| 15 | 18 | Settable | TIM1_BRK,<br>TIM1_UP,<br>TIM1_TRG_COM,<br>TIM1_CC | Timer 1 break, update, trigger and communication, capture compare interrupt | 0x0000 007C |
| 16 | 19 | Settable | TIM2 | Timer 2 global interrupt | 0x0000 0080 |
| 17 | 20 | Settable | TIM16 | Timer 16 global interrupt | 0x0000 0084 |
| 18 | 21 | Settable | TIM17 | Timer 17 global interrupt | 0x0000 0088 |
| 19 | 22 | Settable | IPCC_C2_RX_IT<br>IPCC_C2_TX_IT<br>HSEM | IPCC CPU2 RX occupied interrupt<br>IPCC CPU2 TX free interrupt<br>Semaphore interrupt 1 o CPU2 | 0x0000 008C |
| 20 | 23 | Settable | PKA<br>True RNG | Private key accelerator interrupt (C2IMR1[8])<br>True random number generator interrupt (C2IMR1[9]) | 0x0000 0090 |
| 21 | 24 | Settable | AES2 | AES2 global interrupt | 0x0000 0094 |
| 22 | 25 | Settable | 802_IT1 | 802.15.4 interrupt 1 | 0x0000 0098 |
| 23 | 26 | Settable | I2C1_EV<br>I2C1_ER | I2C1 event interrupt<br>I2C1 error interrupt | 0x0000 009C |
| 24 | 27 | - | - | Reserved | 0x0000 00A0 |
| 25 | 28 | Settable | SPI1 | SPI1 global interrupt | 0x0000 00A4 |
| 26 | 29 | - | - | Reserved | 0x0000 00A8 |
| 27 | 30 | Settable | USART1 | USART1 global interrupt | 0x0000 00AC |
| 28 | 31 | - | - | Reserved | 0x0000 00B0 |
| 29 | 32 | - | - | Reserved | 0x0000 00B4 |
| 30 | 33 | Settable | BLE_BLUE_IT<br>BLE_RFC_IT<br>BLE_RFFMS_IT<br>BLE_HOST_WKUP | BLE blue controller interrupt<br>BLE radio control interrupt<br>BLE radio states interrupt<br>BLE host wakeup interrupt | 0x0000 00B8 |
| 31 | 34 | Settable | 802_IT2<br>802_HOST_WKUP | 802.15.4 interrupt 2<br>802.15.4 host wakeup interrupt | 0x0000 00BC |

## 13.4 Interrupt list

The different wakeup sources are listed in *Table 56*. Depending on its origin, the wakeup is handled according to different types, see *Section 14.4: EXTI functional behavior* for more information.

Some wakeup sources are able to generate an event to the CPUs. see *Event* column.

The wakeup source capability to wakeup CPU1 and or CPU2 is listed in *Wakeup* column.

For CPUs interrupt handling see *Section 13: Nested vectored interrupt controller (NVIC)*.

For Wakeup handling see *Section 14: Extended interrupt and event controller (EXTI)*.

**Table 56. Wakeup interrupt table**

| EXTI no. | Acronym | Description | EXTI type | Event | Wakeup |
|---|---|---|---|---|---|
| 0 | EXTI[0] | EXTI Line 0 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 1 | EXTI[1] | EXTI Line 1 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 2 | EXTI[2] | EXTI Line 2 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 3 | EXTI[3] | EXTI Line 3 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 4 | EXTI[4] | EXTI Line 4 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 5 | EXTI[5] | EXTI Line 5 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 6 | EXTI[6] | EXTI Line 6 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 7 | EXTI[7] | EXTI Line 7 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 8 | EXTI[8] | EXTI Line 8 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 9 | EXTI[8] | EXTI Line 9 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 10 | EXTI[10] | EXTI Line 10 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 11 | EXTI[11] | EXTI Line 11 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 12 | EXTI[12] | EXTI Line 12 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 13 | EXTI[13] | EXTI Line 13 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 14 | EXTI[14] | EXTI Line 14 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 15 | EXTI[15] | EXTI Line 15 from SYSCFG | Configurable A | Yes | CPU1 and CPU2 |
| 16 | PVD | PVD line | Configurable A | No | CPU1 and CPU2 |
| 17 | RTC_ALARM | RTC Alarms (A and B) interrupt | Configurable A | Yes | CPU1 and CPU2 |
| 18 | TAMP, RTC_STAMP, LSE_CSS | RTC Tamper interrupt<br>RTC TimeStamp interrupt<br>RCC LSECSS interrupt | Configurable A | Yes | CPU1 and CPU2 |
| 19 | RTC_WKUP | RTC wakeup interrupt | Configurable A | Yes | CPU1 and CPU2 |
| 20 | Reserved | - | - | - | - |
| 21 | Reserved | - | - | - | - |
| 22 | I2C1 wakeup | I2C1 wakeup | Direct B | No | CPU1 and CPU2 |
| 23 | Reserved | - | - | - | - |
| 24 | USART1 | USART1 wakeup | Direct B | No | CPU1 and CPU2 |
| 25 | Reserved | - | - | - | - |
| 26 | Reserved | - | - | - | - |
| 27 | Reserved | - | - | - | - |
| 28 | Reserved | - | - | - | - |
| 29 | LPTIM1 wakeup | LPtimer 1 wakeup | Direct B | No | CPU1 and CPU2 |
| 30 | LPTIM2 wakeup | LPtimer 2 wakeup | Direct B | No | CPU1 and CPU2 |
| 31 | Reserved | - | - | - | - |

**Table 56. Wakeup interrupt table (continued)**

| EXTI no. | Acronym | Description | EXTI type | Event | Wakeup |
|---|---|---|---|---|---|
| 32 | Reserved | - | - | - | - |
| 33 | Reserved | - | - | - | - |
| 34 | Reserved | - | - | - | - |
| 35 | Reserved | - | - | - | - |
| 36 | IPCC CPU1 interrupts | IPCC CPU1 RX occupied & TX free interrupts | Direct C | No | CPU1(1) |
| 37 | IPCC CPU2 interrupts | IPCC CPU2 RX occupied & TX free interrupts | Direct C | No | CPU2(2) |
| 38 | HSEM interrupt 0 | Semaphore interrupt 0 for CPU1 | Direct C | No | CPU1(1) |
| 39 | HSEM interrupt 1 | Semaphore interrupt 1 for CPU2 | Direct C | No | CPU2(2) |
| 40 | C2SEV | CPU2 SEV line | Configurable A | Yes | CPU1(3) |
| 41 | C1SEV | CPU1 SEV line | Configurable A | Yes | CPU2(4) |
| 42 | Flash interrupt | Flash ECC and global interrupts | Direct C | No | CPU1 and CPU2 |
| 43 | Reserved | - | - | - | - |
| 44 | HSE CSS interrupt | RCC HSE CSS interrupt | Direct C | No | CPU1 and CPU2 |
| 45 | BLE interrupts | BLE, RADIO, & RF_FSM interrupts | Direct B | No | CPU2(2) |
| 46 | 802 interrupts | 802.15.4 Int0 and Int1 interrupts | Direct B | No | CPU2(2) |
| 47 | Reserved | - | - | - | - |
| 48 | CDBGPWRUPREQ | Debug power up request wakeup | Direct | No | CPU1 and CPU2 |

1. For correct operation the EXTI direct event C2IMRx.IMn bit must be set to 0 before CPU1 uses this direct event.
2. For correct operation the EXTI direct event C1IMRx.IMn bit must be set to 0 before CPU2 uses this direct event.
3. For correct operation the EXTI configurable event both C2IMRx.IMn and C2EMRx.EMn bits must be set to 0 before CPU1 uses this configurable event.
4. For correct operation the EXTI configurable event both C1IMRx.IMn and C1EMRx.EMn bits must be set to 0 before CPU2 uses this configurable event.

# 14 Extended interrupt and event controller (EXTI)

The Extended interrupt and event controller (EXTI) manages the individual CPU and system wakeup through configurable and direct event inputs. It provides wakeup requests to the power control, and generates an interrupt request to the CPUs interrupt controller and events to the CPUs event input. For each CPU an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wakeup requests allow the system to be woken up from STOP modes, and the CPU to be woken up from the CSTOP and CSTANDBY modes.

The interrupt request and event request generation can also be used in RUN modes.

## 14.1 EXTI main features

The EXTI main features are the following:

- 49 input events supported
- Two CPUs supported
- All event inputs allow to wake up the system
- Events which do not have an associated wakeup flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI
- Some events can be used to generate a CPU wakeup event

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse)
  - Configurable events have the following features:
    - Selectable active trigger edge
    - Interrupt pending status register bit.
    - Individual interrupt and event generation mask, used for conditioning the CPU wakeup, interrupt and event generation
    - SW trigger possibility
- Direct events (interrupt and wakeup sources from peripherals having an associated flag which requiring to be cleared in the peripheral)
  - Direct events have the following features:
    - Fixed rising edge active trigger
    - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
    - Individual interrupt and event generation mask, used for conditioning the CPU wakeup and event generation
    - No SW trigger possibility

## 14.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input Trigger block, and the masking block as shown in *Figure 29*.

The register block contains all the EXTI registers.

The event input trigger block provides event input edge trigger logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, and the masking of these.

**Figure 29. EXTI block diagram**

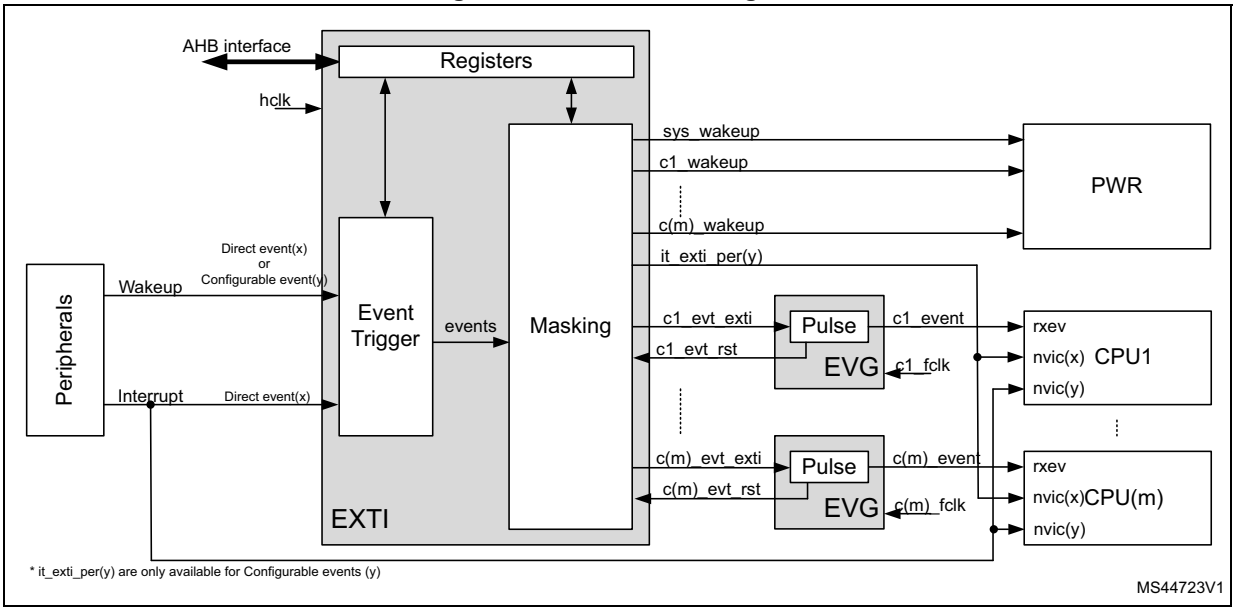



**Table 57. EXTI pin overview**

| Pin name | I/O | Description |
|---|---|---|
| AHB interface | I/O | EXTI register bus interface. When one event is configured to allow security, the AHB interface supports secure accesses. |
| hclk | I | AHB bus clock and EXTI system clock. |
| Configurable event(y) | I | Asynchronous wakeup events from peripherals that do not have an associated interrupt and flag in the peripheral. |
| Direct event(x) | I | Synchronous and asynchronous wakeup events from peripherals having an associated interrupt and flag in the peripheral. |
| it_exti_per (y) | O | Interrupts to the CPU1 to CPU(m) associated with Configurable event (y). |
| c(m)_evt_exti | O | High level sensitive event output for CPU(m) synchronous to hclk. (m= 1 to 2) |
| c(m)_evt_rst | I | Asynchronous reset input to clear c(m)_evt_exti. (m= 1 to 2) |
| sys_wakeup | O | Asynchronous system wakeup request to PWR for ck_sys and hclk. |
| c(m)_wakeup | O | Wakeup request to PWR for CPU(m), synchronous to hclk. (m= 1 to 2) |

**Table 58. EVG pin overview**

| Pin name | I/O | Description |
|---|---|---|
| c_fclk | I | CPU free running clock. |
| c_evt_in | I | High level sensitive Events input from EXTI, asynchronous to CPU clock. |
| c_event | O | Event pulse, synchronous to CPU clock. |
| c_evt_rst | O | Event reset signal, synchronous to CPU clock. |

### 14.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate wakeup or interrupt events when the system is in STOP mode or a CPU is in CSTOP mode are connected to the EXTI.

- Peripheral wakeup signals that generate a pulse or which do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events the EXTI provides a status pending bit that has to be cleared. It is the EXTI interrupt associated with the status bit that will interrupt the CPU.
- Peripheral interrupt and wakeup signals that have a status bit in the peripheral that has to be cleared in the peripheral, are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wakeup is cleared by the CPU in the peripheral. It is the peripheral interrupt that will interrupt the CPU directly.

The EXTI configurable event interrupts are connected to the respective interrupt controller of each CPU(m).

The dedicated EXTI/EVG CPU(m) event is connected to the respective CPU(m) rxev input.

The EXTI CPU(m) wakeup signals are connected to the PWR block, and are used to wake up the system and CPU(m) sub-system bus clocks.

## 14.3 EXTI functional description

Depending on the EXTI event input type and wakeup target(s), different logic implementations are used. The applicable features are controlled from register bits:

- Active trigger edge enable, by rising edge selection
  *EXTI rising trigger selection register (EXTI_RTSR1)*,
  *EXTI rising trigger selection register (EXTI_RTSR2)*,
  and falling edge selection
  *EXTI falling trigger selection register (EXTI_FTSR1)*,
  *EXTI falling trigger selection register (EXTI_FTSR2)*.
- Software trigger, by
  *EXTI software interrupt event register (EXTI_SWIER1)*,
  *EXTI software interrupt event register (EXTI_SWIER2)*.
- Interrupt pending flag, by
  *EXTI pending register (EXTI_PR1)*,
  *EXTI pending register (EXTI_PR2)*.
- CPU wakeup and interrupt enable, by
  *EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)*,
  *EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR1)*,
  *EXTI CPU wakeup with interrupt mask register (EXTI_IMR2)*,
  *EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR2)*.
- CPU wakeup and event enable, by
  *EXTI CPU wakeup with event mask register (EXTI_EMR1)*,
  *EXTI CPU2 wakeup with event mask register (EXTI_C2EMR1)*
  *EXTI CPU wakeup with event mask register (EXTI_EMR2)*,
  *EXTI CPU2 wakeup with event mask register (EXTI_C2EMR2)*.

**Table 59. EXTI event input configurations and register control**

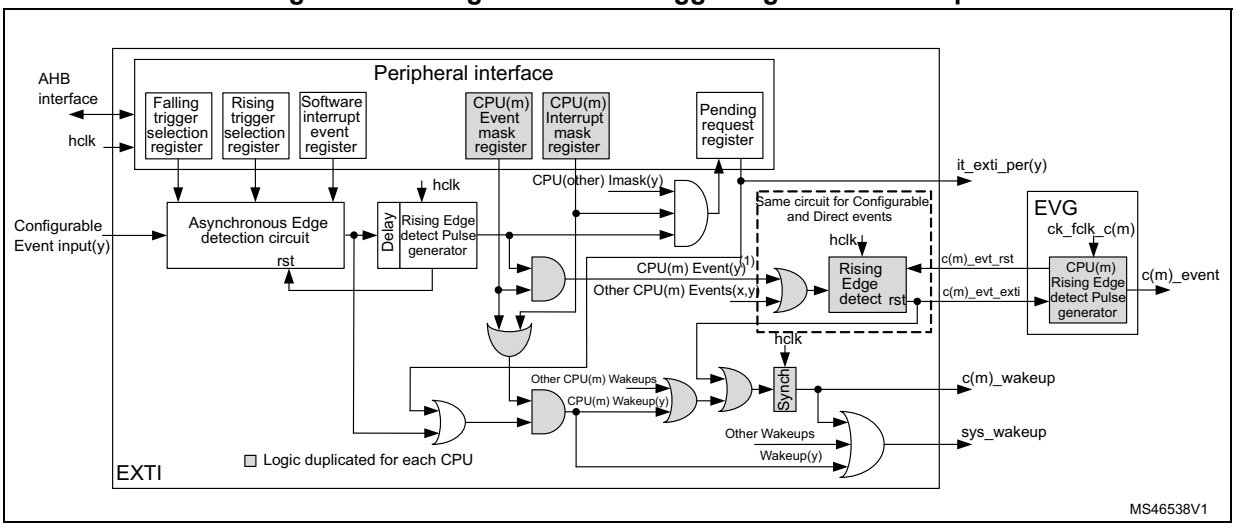
| Event input type | Logic implementation | EXTI_RTSR | EXTI_FTSR | EXTI_SWIER | EXTI_PR | EXTI_CmIMR | EXTI_CmEMR(1) |
|---|---|---|---|---|---|---|---|
| Configurable | Configurable event input wakeup logic | x | x | x | x | x | x |
| Direct | Direct event input wakeup logic | - | - | - | - | x | x |

1. Only for input events with configuration “rxev generation” enabled.

### 14.3.1 EXTI configurable event input wakeup

*Figure 30* is a detailed representation of the logic associated with configurable event inputs which will wake up the CPU(m) sub-system bus clocks and generated an EXTI pending flag and interrupt to the CPU(m) and or a CPU(m) wakeup event.

**Figure 30. Configurable event trigger logic CPU wakeup**




1. Only for the input events that support CPU rxev generation c(n)_event.

The software interrupt event register allows to trigger configurable events by software, writing the corresponding register bit, irrespective of the edge selection setting.

The rising edge and falling edge selection registers allow to enable and select the configurable event active trigger edge or both edges.

Each CPU has its dedicated interrupt mask register and a dedicated event mask registers. The enabled event allows to generate an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The event pending register (EXTI_PR) is not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers, shared by the CPUs. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to all CPUs. The configurable event interrupts need to be acknowledged by software in the EXTI_PR register.

When a CPU(m) interrupt or CPU(m) event is enabled the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

*Note: A detected configurable event interrupt pending request, may be cleared by any CPU. The system is unable to enter into Low-power modes as long as an interrupt pending request is active.*

### 14.3.2 EXTI direct event input wakeup

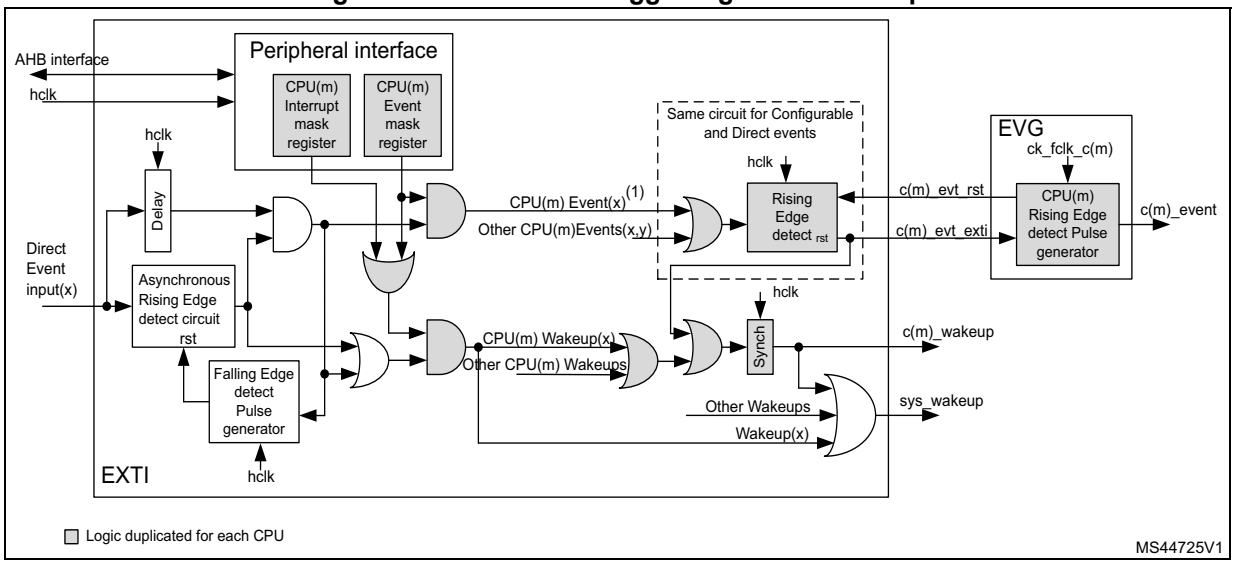
*Figure 31* is a detailed representation of the logic associated with direct event inputs waking up the system.

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU sub-system clocks and may generate a CPU wakeup event. The peripheral synchronous interrupt, associated with the direct wakeup event wakes up the CPU.

The EXTI direct event is able to generate a CPU event. This CPU event wakes up the CPU. The CPU event may occur before the associated Peripheral interrupt flag is set.

*Note: The direct events are cleared in the peripheral generating the event. When a direct event input enabled by CPU(m) is cleared by another CPU before the CPU(m) clock is running, the CPU(m) no longer receives a CPU(m) interrupt nor a CPU(m) event and does not wake up. However the system stays in RUN mode, generating the CPU(m) clock. For this reason CPU(m) direct events must NOT be cleared by the other CPU.*

**Figure 31. Direct event trigger logic CPU wakeup**




1. Only for the input events that support CPU rxev generation c(n)_event.

## 14.4 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wakeup event. The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the generation of a CPU(m) wakeup is conditioned by the CPU(m) interrupt mask and CPU(m) event mask.

**Table 60. Masking functionality**

| CPU interrupt enable EXTI_CmIMR.IMn | CPU event enable EXTI_CmEMR.EMn | Configurable event inputs EXTI_PR.PIFn | exti(n) interrupt(1) | CPU(m) event | CPU(m) wakeup |
|---|---|---|---|---|---|
| 0 for all CPUs | 0 | No | Masked | Masked | Masked |
| | 1 | | Masked | Yes | Yes |
| 1 for any CPU | 0 | Status latched | Yes | Masked | Yes(2) |
| | 1 | | Yes | Yes | Yes |

1. The single exti(n) interrupt goes to all CPUs. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU(m) interrupt controller.
2. Only if CPU(m) interrupt is enabled in EXTI_CmIMR.IMn.

For configurable event inputs, when the enabled edge(s) occur on the event input, an event request is generated. When the associated CPU(m) interrupt is unmasked the corresponding pending bit EXTI_PR.PIFn is set and the CPU(m) sub-system is woken up and CPU interrupt signal is activated. The EXTI_PR.PIFn pending bit must be cleared by software writing, it to ‘1’. This clears the CPU interrupt.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit in the EXTI. When the associated CPU(m) interrupt is unmasked the corresponding CPU sub-system is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU(m) event must be unmasked to generate an event. When the enabled edge(s) occur on the event input a CPU(m) event pulse is generated. There is no event pending bit.

For the configurable event inputs an event request can be generated by software when writing a ‘1’ in the software interrupt/event register EXTI_SWIER, generating a rising edge on the event. The edge event pending bit is set in EXTI_PR, irrespective of the setting in EXTI_RTSR.

## 14.5 EXTI registers

The EXTI register map is divided as detailed in *Table 61*.

**Table 61. EXTI register map sections**

| Address | Description |
|---|---|
| 0x000 - 0x01C | General configurable event [31:0] configuration |
| 0x020 - 0x03C | General configurable event [63:32] configuration |
| 0x040 - 0x05C | General configurable event [95:64] configuration |
| 0x080 - 0x0BC | CPU1 input event configuration |
| 0x0C0 - 0x0FC | CPU2 input event configuration |

All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

### 14.5.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

| 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. | RT19 | RT18 | RT17 | RT16 |
| | | | | | | | | | | | | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RT15 | RT14 | RT13 | RT12 | RT11 | RT10 | RT9 | RT8 | RT7 | RT6 | RT5 | RT4 | RT3 | RT2 | RT1 | RT0 |
| 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 **RT[19:0]:** Rising trigger event configuration bit of configurable event input x (x = 19 to 0)[1].
0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

### 14.5.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

| 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. | FT19 | FT18 | FT17 | FT16 |
| | | | | | | | | | | | | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FT15 | FT14 | FT13 | FT12 | FT11 | FT10 | FT9 | FT8 | FT7 | FT6 | FT5 | FT4 | FT3 | FT2 | FT1 | FT0 |
| 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 **FT[19:0]:** Falling trigger event configuration bit of configurable event input x (x = 19 to 0)[1].
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

### 14.5.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

| 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. | SWI19 | SWI18 | SWI17 | SWI16 |
| | | | | | | | | | | | | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWI15 | SWI14 | SWI13 | SWI12 | SWI11 | SWI10 | SWI9 | SWI8 | SWI7 | SWI6 | SWI5 | SWI4 | SWI3 | SWI2 | SWI1 | SWI0 |
| 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 **SWI[19:0]:** Software interrupt on event x (x = 19 to 0)
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger a rising edge event on event x.
This bit is auto cleared by HW.

### 14.5.4 EXTI pending register (EXTI_PR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

| 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. | PIF19 | PIF18 | PIF17 | PIF16 |
| | | | | | | | | | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PIF15 | PIF14 | PIF13 | PIF12 | PIF11 | PIF10 | PIF9 | PIF8 | PIF7 | PIF6 | PIF5 | PIF4 | PIF3 | PIF2 | PIF1 | PIF0 |
| 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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **PIF[19:0]:** configurable event inputs x (x = 19 to 0) Pending bit.
0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

### 14.5.5 EXTI rising trigger selection register (EXTI_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

Contains only register bits for configurable events.

| 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. | RT41 | RT40 | Res. | Res. | Res. | Res. | Res. | Res. | RT33 | Res. |
| | | | | | | rw | rw | | | | | | | rw | |

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

Bit 9 **RT41:** Rising trigger event configuration bit of configurable event input 41[(1)]
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bit 8 **RT40:** Rising trigger event configuration bit of configurable event input 40[(1)]
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

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

Bit 1 **RT33:** Rising trigger event configuration bit of configurable event input 33[(1)]
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bit 0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable Event input. In this case, both edges generate a trigger.

### 14.5.6 EXTI falling trigger selection register (EXTI_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

Contains only register bits for configurable events.

| 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. | FT41 | FT40 | Res. | Res. | Res. | Res. | Res. | Res. | FT33 | Res. |
| | | | | | | rw | rw | | | | | | | rw | |

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

Bit 9 **FT41:** Falling trigger event configuration bit of configurable event input 41[(1)]
0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bit 8 **FT40:** Falling trigger event configuration bit of configurable event input 40[(1)]
0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

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

Bit 1 **FT33:** Falling trigger event configuration bit of configurable event input 33[(1)]
0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bit 0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

### 14.5.7 EXTI software interrupt event register (EXTI_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

Contains only register bits for configurable events.

| 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. | SWI41 | SWI40 | Res. | Res. | Res. | Res. | Res. | Res. | SWI33 | Res. |
| | | | | | | rw | rw | | | | | | | rw | |

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

Bit 9 **SWI41:** Software interrupt on event 41

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 41.

This bit is auto cleared by HW.

Bit 8 **SWI40:** Software interrupt on event 40

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 40.

This bit is auto cleared by HW.

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

Bit 1 **SWI33:** Software interrupt on event 33

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 33.

This bit is auto cleared by HW.

Bit 0 Reserved, must be kept at reset value.

### 14.5.8 EXTI pending register (EXTI_PR2)

Address offset: 0x02C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

| 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. | PIF41 | PIF40 | Res. | Res. | Res. | Res. | Res. | Res. | PIF33 | Res. |
| | | | | | | rc_w1 | rc_w1 | | | | | | | rc_w1 | |

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

Bit 9 **PIF41:** configurable event inputs 41 pending bit

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 8 **PIF40:** configurable event inputs 40 pending bit

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

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

Bit 1 **PIF33:** configurable event inputs 33 pending bit

0: No trigger request occurred
1: selected trigger request occurred

This bit is set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 0 Reserved, must be kept at reset value.

### 14.5.9 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0x7FC0 0000

Contains register bits for configurable events and direct events.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | IM30 | IM29 | Res. | IM27 | IM26 | Res. | IM24 | Res. | IM22 | Res. | Res. | IM19 | IM18 | IM17 | IM16 |
| | 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 |
| IM15 | IM14 | IM13 | IM12 | IM11 | IM10 | IM9 | IM8 | IM7 | IM6 | IM5 | IM4 | IM3 | IM2 | IM1 | IM0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31, 28, 25, 23, 21:20 Reserved, must be kept at reset value.

Bits 30:29, 27:26, 24, 22, 19:0 **IM[30:29, 27:26, 24, 22, 19:0]:** CPU wakeup with interrupt mask on event input x (x = 30:29, 27:26, 24, 22, 19:0)(1)(2).

0: Wakeup with interrupt request from Line x is masked
1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

### 14.5.10 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR1)

Address offset: 0x0C0

Reset value: 0x7FC0 0000

Contains register bits for configurable events and direct events.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | IM30 | IM29 | Res. | IM27 | IM26 | Res. | IM24 | Res. | IM22 | Res. | Res. | IM19 | IM18 | IM17 | IM16 |
| | 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 |
| IM15 | IM14 | IM13 | IM12 | IM11 | IM10 | IM9 | IM8 | IM7 | IM6 | IM5 | IM4 | IM3 | IM2 | IM1 | IM0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31, 28, 25, 23, 21:20 Reserved, must be kept at reset value.

Bits 30:29, 27:26, 24, 22, 19:0 **IM[30:29, 27:26, 24, 22, 19 :0]:** CPU2 wakeup with interrupt mask on event input x (x = 30:29, 27:26, 24, 22, 19 to 0)[1][2].

0: Wakeup with interrupt request from Line x is masked
1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

### 14.5.11 EXTI CPU wakeup with event mask register (EXTI_EMR1)

Address offset: 0x084

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. | EM19 | EM18 | EM17 | Res. |
| | | | | | | | | | | | | rw | rw | rw | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EM15 | EM14 | EM13 | EM12 | EM11 | EM10 | EM9 | EM8 | EM7 | EM6 | EM5 | EM4 | EM3 | EM2 | EM1 | EM0 |
| 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 19:17 **EM[19:17]:** CPU wakeup with event generation mask on event input x (x = 19 to 17)

0: Wakeup with event generation from Line x is masked
1: Wakeup with event generation from Line x is unmasked

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **EM[15:0]:** CPU wakeup with event generation mask on event input x (x = 15 to 0)

0: Wakeup with event generation from Line x is masked
1: Wakeup with event generation from Line x is unmasked

### 14.5.12 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR1)

Address offset: 0x0C4

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. | EM19 | EM18 | EM17 | Res. |
| | | | | | | | | | | | | rw | rw | rw | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EM15 | EM14 | EM13 | EM12 | EM11 | EM10 | EM9 | EM8 | EM7 | EM6 | EM5 | EM4 | EM3 | EM2 | EM1 | EM0 |
| 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 19:17 **EM[19:17]:** CPU2 wakeup with event generation mask on event input x (x = 19 to 17)

0: Wakeup with event generation from Line x is masked
1: Wakeup with event generation from Line x is unmasked

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **EM[:0]:** CPU2 wakeup with event generation mask on event input x (x = 15 to 0)
0: Wakeup with event generation from Line x is masked
1: Wakeup with event generation from Line x is unmasked

### 14.5.13 EXTI CPU wakeup with interrupt mask register (EXTI_IMR2)

Address offset: 0x090

Reset value: 0x0001 FCFD

Contains register bits for configurable events and direct events.

| 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. | IM48 |
| | | | | | | | | | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | IM46 | IM45 | IM44 | Res. | IM42 | IM41 | IM40 | IM39 | IM38 | IM37 | IM36 | Res. | Res. | IM33 | Res. |
| | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw | | | rw | |

Bits 31:17, 15, 11, 3:2, 0 Reserved, must be kept at reset value.

Bits 16, 14:12, 10:4, 1 **IM[48, 46:44, 42:36, 33]:** CPU wakeup with interrupt mask on event input x (x = 48, 46 to 44, 42 to 36, 33)(1)(2)
0: Wakeup with interrupt request from Line x is masked
1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

### 14.5.14 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR2)

Address offset: 0x0D0

Reset value: 0x0001 FCFD

Contains register bits for configurable events and direct events.

| 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. | IM48 |
| | | | | | | | | | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | IM46 | IM45 | IM44 | Res. | IM42 | IM41 | IM40 | IM39 | IM38 | IM37 | IM36 | Res. | Res. | IM33 | Res. |
| | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw | | | rw | |

Bits 31:17, 15, 11, 3:2, 0 Reserved, must be kept at reset value.

Bits 16, 14:12, 10:4, 1 **IM[48, 46:44, 42:36, 33]:** CPU wakeup with interrupt mask on event input x (x = 48, 46 to 44, 42 to 36, 33)(1)(2)
0: Wakeup with interrupt request from Line x is masked
1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

### 14.5.15 EXTI CPU wakeup with event mask register (EXTI_EMR2)

Address offset: 0x094

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. | EM41 | EM40 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | rw | rw | | | | | | | | |

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

Bit 9 **EM41:** CPU wakeup with event generation mask on event input 41
0: Wakeup with event generation from Line 41 is masked
1: Wakeup with event generation from Line 41 is unmasked

Bit 8 **EM40:** CPU wakeup with event generation mask on event input 40.
0: Wakeup with event generation from Line 40 is masked
1: Wakeup with event generation from Line 40 is unmasked

Bits 7:0 Reserved, must be kept at reset value.

### 14.5.16 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR2)

Address offset: 0x0D4

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. | EM41 | EM40 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | rw | rw | | | | | | | | |

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

Bit 9 **EM41:** CPU2 wakeup with event generation mask on event input 41
0: Wakeup with event generation from Line 41 is masked
1: Wakeup with event generation from Line 41 is unmasked

Bit 8 **EM40:** CPU2 wakeup with event generation mask on event input 40.
0: Wakeup with event generation from Line 40 is masked
1: Wakeup with event generation from Line 40 is unmasked

Bits 7:0 Reserved, must be kept at reset value.

### 14.5.17 EXTI register map

The following table gives the EXTI register map and the reset values.

**Table 62. EXTI 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 | **EXTI_RTSR1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | RT[19:0] | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x004 | **EXTI_FTSR1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | FT[19:0] | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x008 | **EXTI_SWIER1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SW[19:0] | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00C | **EXTI_RPR1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PIF[19:0] | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x010-0x01C | 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. |
| 0x020 | **EXTI_RTSR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | RT [41:40] | | Res. | Res. | Res. | Res. | Res. | Res. | RT33 | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | 0 | |
| 0x024 | **EXTI_FTSR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | FT [41:40] | | Res. | Res. | Res. | Res. | Res. | Res. | FT33 | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | 0 | |
| 0x028 | **EXTI_SWIER2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SWI [41:40] | | Res. | Res. | Res. | Res. | Res. | Res. | SWI33 | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | 0 | |
| 0x02C | **EXTI_PR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PIF [41:40] | | Res. | Res. | Res. | Res. | Res. | Res. | PIF33 | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | 0 | |
| 0x030-0x07C | 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. |
| 0x080 | **EXTI_IMR1** | Res. | IM [30:29] | | Res. | IM [27:26] | | Res. | IM 24 | Res. | IM 22 | Res. | Res. | IM[19:0] | | | | | | | | | | | | | | | | | | |
| | Reset value | | 1 | 1 | | 1 | 1 | | 1 | | 1 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x084 | **EXTI_EMR1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EM[19:17] | | | Res. | EM[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x088-0x08C | 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. |
| 0x090 | **EXTI_IMR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | IM[48] | Res. | IM[46:44] | | | Res. | IM [42:36] | | | | | | | Res. | Res. | IM[33] | Res. |
| | Reset value | | | | | | | | | | | | | | | | 1 | | 1 | 1 | 1 | | 1 | 0 | 0 | 1 | 1 | 1 | 1 | | | 0 | |
| 0x094 | **EXTI_EMR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EM [41:40] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | | |
| 0x098-0x0BC | 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. |

**Table 62. EXTI 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0C0 | **EXTI_C2IMR1** | Res. | IM [30:29] | | Res. | IM [27:26] | | Res. | IM 24 | Res. | IM 22 | Res. | Res. | IM[19:0] | | | | | | | | | | | | | | | | | | | |
| | Reset value | | 1 | 1 | | 1 | 1 | | 1 | | 1 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0C4 | **EXTI_C2EMR1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EM[19:17] | | | Res. | EM[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0C8-0x0CC | 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. |
| 0x0D0 | **EXTI_CIMR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | IM 48 | Res. | IM[46:44] | | | Res. | IM [42:36] | | | | | | | Res. | Res. | IM 33 | Res. |
| | Reset value | | | | | | | | | | | | | | | | 1 | | 1 | 1 | 1 | | 1 | 0 | 0 | 1 | 1 | 1 | 1 | | | 0 | |
| 0x0D4 | **EXTI_C2EMR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EM [41:40] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | | |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 15 Analog-to-digital converter (ADC)

## 15.1 Introduction

The ADC consists of a 12-bit successive approximation analog-to-digital converter.

The ADC has up to 14 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The ADC is mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows to improve analog performance while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

## 15.2 ADC main features

- High-performance features
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time is independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Manage single-ended or differential inputs
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Up to four injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - 4 dedicated data registers for the injected channels
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256
  - Programmable data shift up to 8-bit
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency

application (auto-delayed mode)

- 11 external analog input channels
- In addition, there are several internal dedicated channels
  - The internal reference voltage ($V_{REFINT}$)
  - The internal temperature sensor ($V_{TS}$)
  - The $V_{BAT}$ monitoring channel ($V_{BAT}/3$)
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - The ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{DDA}$

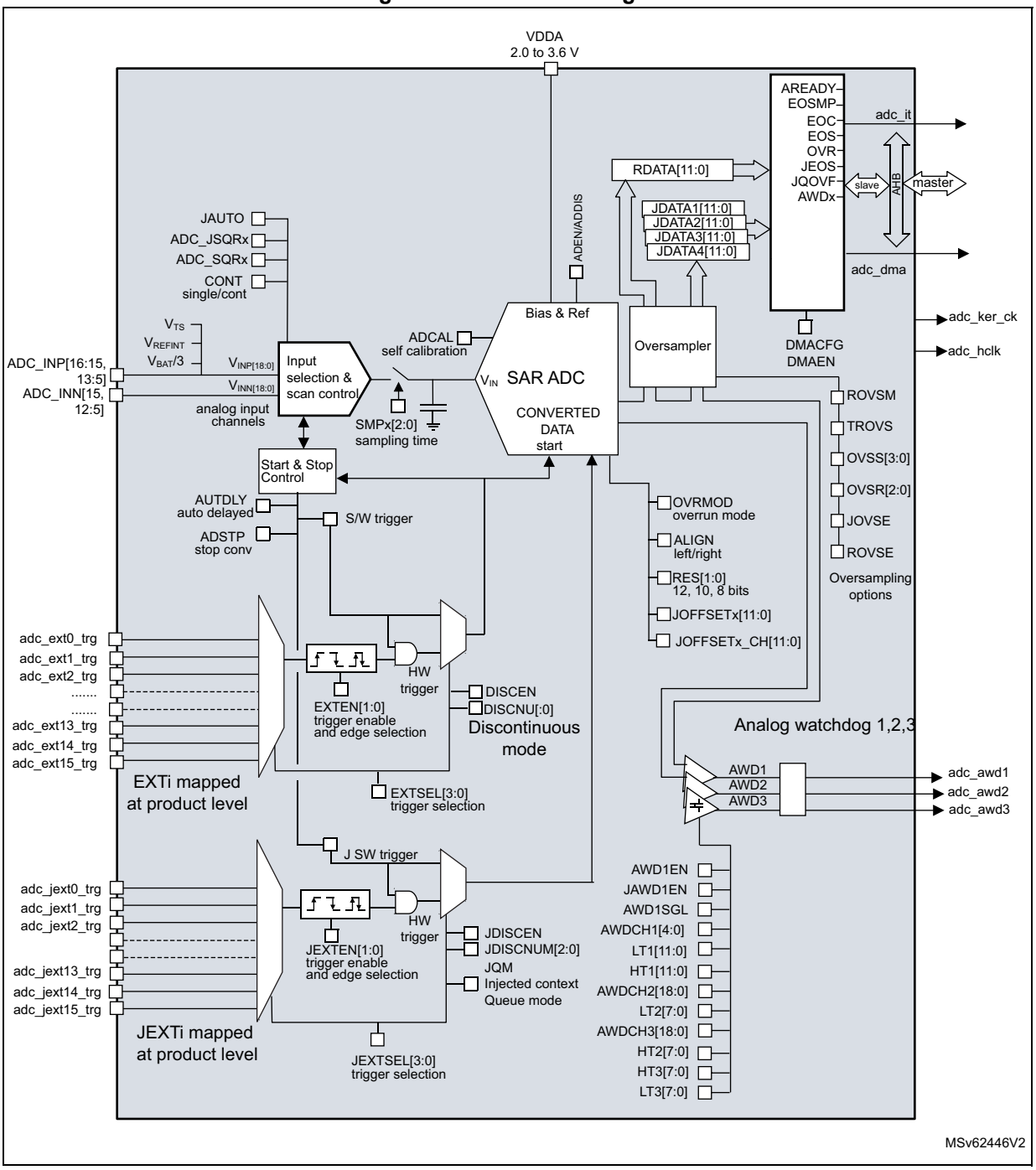
*Figure 32* shows the block diagram of one ADC.

## 15.3 ADC functional description

### 15.3.1 ADC block diagram

*Figure 32* shows the ADC block diagram and *Table 64* gives the ADC pin description.

**Figure 32. ADC block diagram**



### 15.3.2 ADC pins and internal signals

**Table 63. ADC internal input/output signals**

| Internal signal name | Signal type | Description |
|---|---|---|
| EXT[15:0] | Inputs | Up to 16 external trigger inputs for the regular conversions (can be connected to on-chip timers). |
| JEXT[15:0] | Inputs | Up to 16 external trigger inputs for the injected conversions (can be connected to on-chip timers). |
| ADC_AWDx_OUT | Output | Internal analog watchdog output signal connected to on-chip timers (x = Analog watchdog number 1,2,3) |
| $V_{TS}$ | Input | Output voltage from internal temperature sensor |
| $V_{REFINT}$ | Input | Output voltage from internal reference voltage |
| $V_{BAT}$ | Input supply | External battery voltage supply |

**Table 64. ADC input/output pins**

| Pin name | Signal type | Comments |
|---|---|---|
| VDDA | Input, analog supply | Analog power supply equal $V_{DDA}$ |
| VSSA | Input, analog supply ground | Ground for analog power supply. On device package which do not have a dedicated $V_{SSA}$ pin, $V_{SSA}$ is internally connected to $V_{SS}$. |
| $V_{INP}i$ | Positive analog input channels | Connected either to ADC1_INP*i* external channels or to internal channels. This input is converted in single-ended mode |
| $V_{INN}i$ | Negative analog input channels | Connected either to $V_{SSA}$ or to external channels: ADC1_INNi and ADC1_INP[i+1]. |
| ADC1_INNi | Negative external analog input signals | Up to 9 analog input channels<br>Refer to *Section 15.3.4: ADC1 connectivity* for details. |
| ADC1_INPi | Positive external analog input signals | Up to 11 analog input channels<br>Refer to *Section 15.3.4: ADC1 connectivity* for details |

### 15.3.3 ADC clocks

#### Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The input clock is and can be selected between two different clock sources (see *Figure 33: ADC clock scheme*):

1. The ADC clock can be a specific clock source, derived from the following clock sources:
   - The system clock
   - PLL “P” clock

   Refer to RCC Section for more information on how to generate ADC dedicated clock. To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be reset.
2. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).

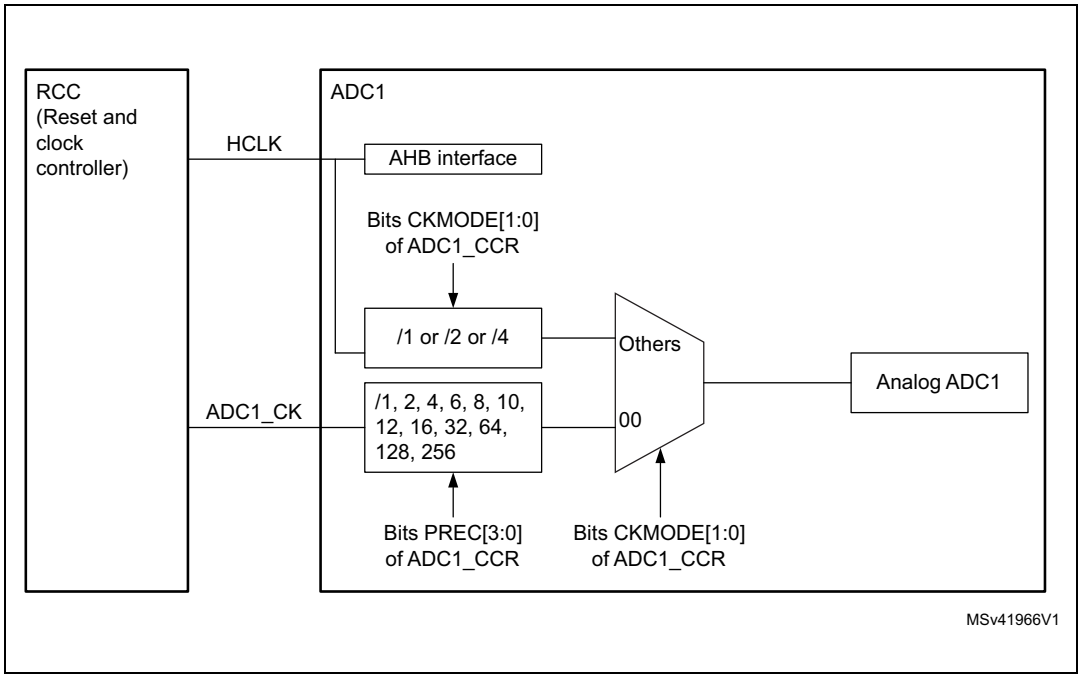
   To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be different from 00.

*Note: For option 2), a prescaling factor of 1 (CKMODE[1:0] = 01) can be used only if the AHB prescaler is set (HPRE[3:0] = 0xxx in RCC_CFGR register).*

Option 1) has the advantage of reaching the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256; using the prescaler configured with bits PRESC[3:0] in the ADCx_CCR register.

Option 2) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant time is added by the resynchronizations between the two clock domains).

**Figure 33. ADC clock scheme**



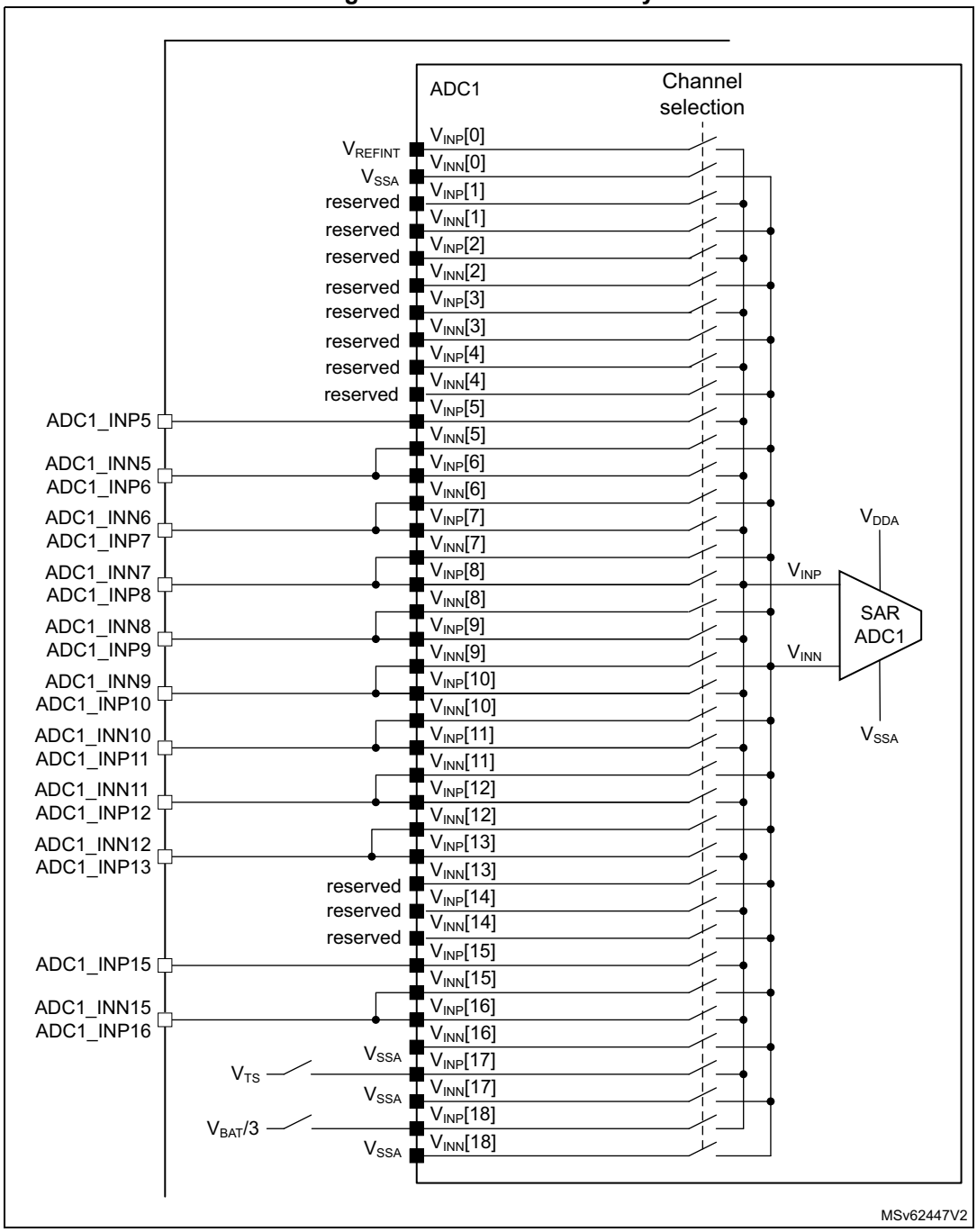

## Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{HCLK} \geq F_{ADC} / 4$ if the resolution of all channels are 12-bit or 10-bit
- $F_{HCLK} \geq F_{ADC} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolution)
- $F_{HCLK} \geq F_{ADC} / 2$ if there are some channels with resolutions equal to 6-bit

### 15.3.4 ADC1 connectivity

**Figure 34. ADC1 connectivity**



### 15.3.5 Slave AHB interface

The ADC implements an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

### 15.3.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in Deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC_CR register).

To start ADC operations, follow the sequence below:

1. Exit Deep-power-down mode by clearing DEEPPWD bit.
2. Enable the ADC voltage regulator by setting ADVREGEN.
3. Wait for the startup time to configure the ADC (refer to the device datasheet for the value of the startup time).

When ADC operations are complete, the ADC can be disabled (ADEN = 0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN = 0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC Deep-power-down mode by setting bit DEEPPWD = 1 into ADC_CR register. This is particularly interesting before entering Stop mode.

*Note: Writing DEEPPWD = 1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.*

*When the internal voltage regulator is disabled (ADVREGEN = 0), the internal analog calibration is kept.*

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or re-apply the calibration factor which was previously saved (refer to *Section 15.3.8: Calibration (ADCAL, ADCALDIF, ADC_CALFACT)*).

Before entering Stop 2 mode, the ADC needs to be disabled by setting ADDIS and clearing ADVREGEN.

### 15.3.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN = 0). Note that the DIFSEL[i] bits corresponding to single-ended channels are always programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the ADCy_INPx external voltage equal to $V_{INP[i]}$ (positive input) and $V_{SSA}$.

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the ADCy_INPx external voltage positive input equal to $V_{INP[i]}$, and the ADCy_INNx negative input equal to $V_{INN[i]}$.

When ADC is configured as differential mode, both inputs should be biased at ($V_{DDA}$) / 2 voltage. Refer to the device datasheet for the allowed common mode input voltage $V_{CMIN}$.

The input signals are supposed to be differential (common mode voltage should be fixed).

Internal channels (such as $V_{TS}$ and $V_{REFINT}$) are used in single-ended mode only.

For a complete description of how the input channels are connected, refer to *Section 15.3.4: ADC1 connectivity*.

**Caution:** When configuring the channel “i” in differential input mode, its negative input voltage $V_{INN[i]}$ is connected to another channel. As a consequence, this channel is no longer usable in single-ended mode or in differential mode and must never be configured to be converted.

### 15.3.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

The ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF = 0 before launching a calibration to be applied for single-ended input conversions.
- Write ADCALDIF = 1 before launching a calibration to be applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when the ADC is disabled (when ADEN = 0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADC_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

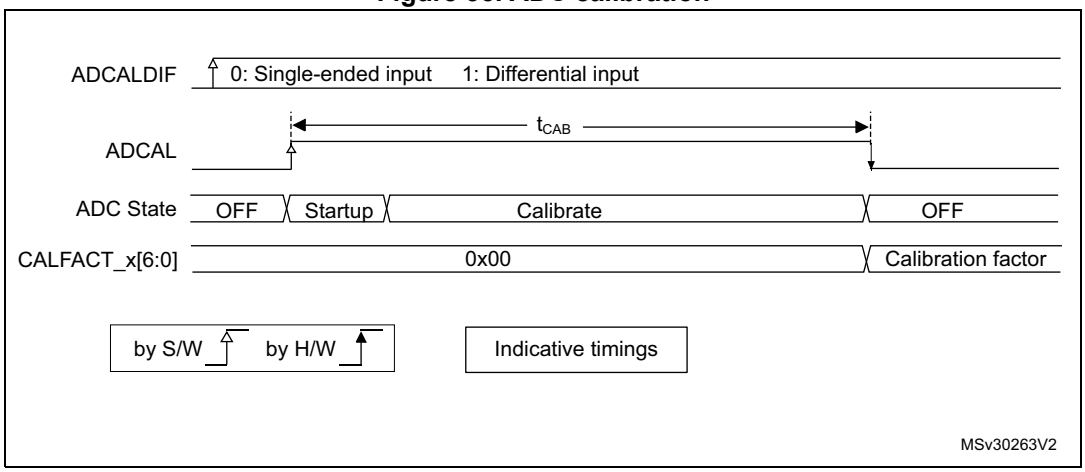
The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in Standby or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when $V_{DDA}$ voltage changed more than 10%.

### Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by setting ADCALDIF = 0 (single-ended input) or ADCALDIF = 1 (differential input).
4. Set ADCAL.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from ADC_CALFACT register.

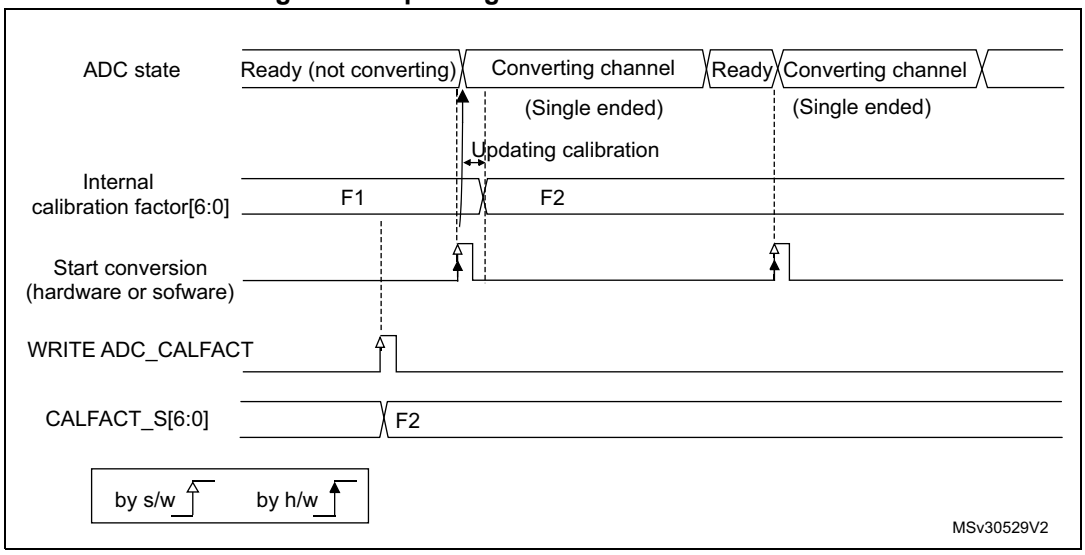
**Figure 35. ADC calibration**




### Software procedure to re-inject a calibration factor into the ADC

1. Ensure ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

**Figure 36. Updating the ADC calibration factor**

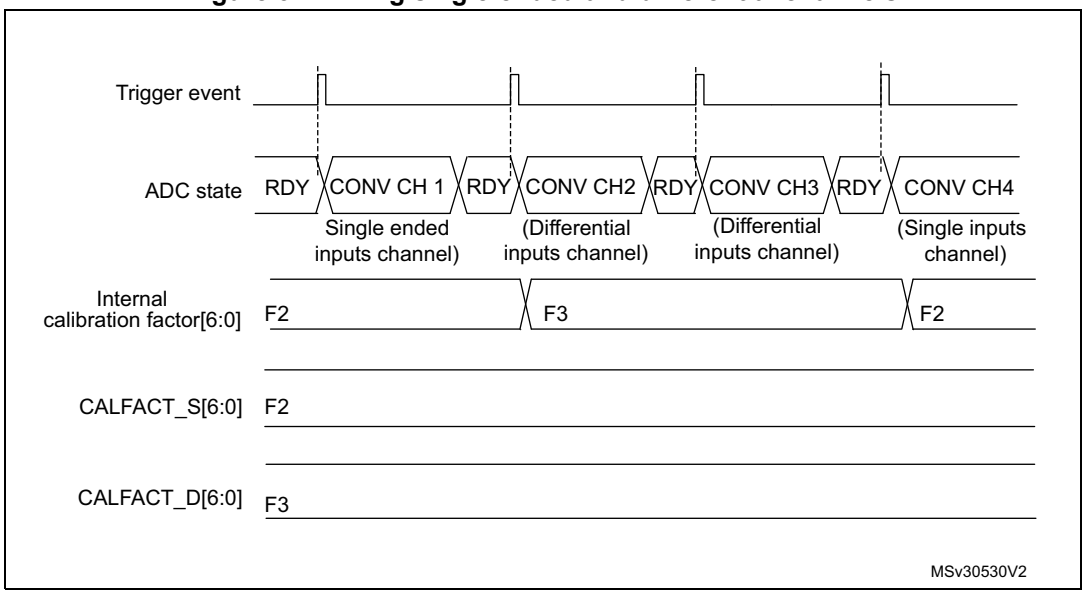



## Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration is automatically injected into the analog ADC.

**Figure 37. Mixing single-ended and differential channels**



### 15.3.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in *Section 15.3.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD = 0 and ADVREGEN = 1, the ADC can be enabled and the ADC needs a stabilization time of $t_{STAB}$ before it starts converting accurately, as shown in *Figure 38*. Two control bits enable or disable the ADC:

- ADEN = 1 enables the ADC. The flag ADRDY is set once the ADC is ready for operation.
- ADDIS = 1 disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART = 1 (refer to *Section 15.3.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART = 1 or when an external injected trigger event occurs, if injected triggers are enabled.

#### Software procedure to enable the ADC

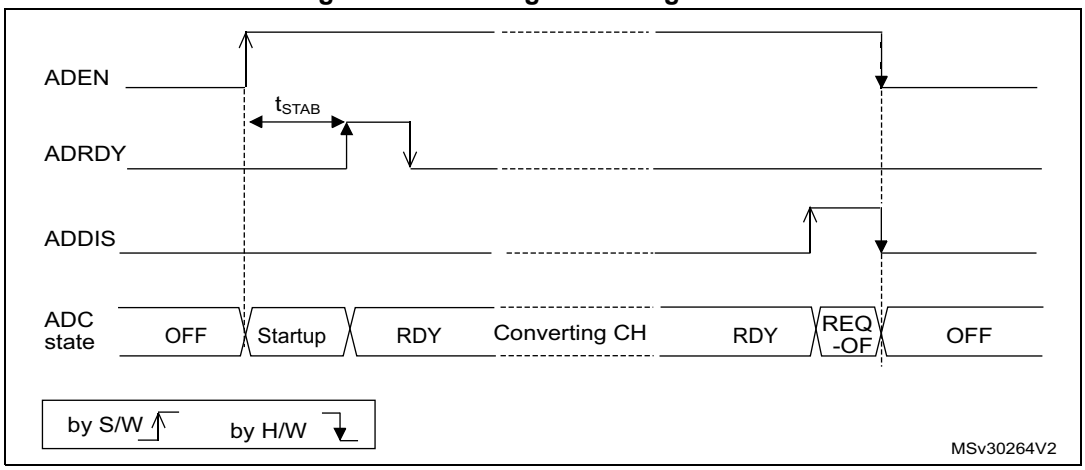
1. Clear the ADRDY bit in the ADC_ISR register by writing 1.
2. Set ADEN.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE = 1).
4. Clear the ADRDY bit in the ADC_ISR register by writing 1 (optional).

**Caution:** ADEN bit cannot be set when ADCAL is set and during four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).

#### Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP = 1 and JADSTP = 1 and then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS.
3. If required by the application, wait until ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

**Figure 38. Enabling / disabling the ADC**




## 15.3.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the DIFSEL[i] control bits in the ADC_DIFSEL register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRx, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).

The software is allowed to write the ADSTP or JADSTP control bits of the ADC_CR register only if the ADC is enabled, possibly converting, and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN = 1). Refer to *Section 15.6.16: ADC injected sequence register (ADC_JSQR)* for additional details.

*Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN as well as all the bits of ADC_CR register).*

### 15.3.11 Channel selection (SQRx, JSQRx)

There are up to 14 multiplexed channels:

- 11 analog inputs coming from GPIO pads
- The ADC is connected to the following internal analog inputs:
  - The internal reference voltage ($V_{REFINT}$) is connected to ADC1_INP0.
  - The internal temperature sensor ($V_{TS}$) is connected to ADC1_INP17.
  - The $V_{BAT}$ monitoring channel ($V_{BAT}/3$) is connected to ADC1_INP18.

*Note: To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, CH17SEL or CH18SEL in the ADCx_CCR registers.*

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC1_INP/INN5, ADC1_INP/INN8, ADC1_INP/INN7, ADC1_INN/INP7, ADC1_INP/INN0, ADC1_INP/INN7, ADC1_INP/INN7, ADC1_INP/INN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP = 1 (refer to *Section 15.3.17: Stopping an ongoing conversion (ADSTP, JADSTP)*).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set (injected conversions ongoing) only when the context queue is enabled (JQDIS = 0 in ADC_CFGR register). Refer to *Section 15.3.21: Queue of context for injected conversions*

### 15.3.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

$T_{CONV}$ = Sampling time + 12.5 ADC clock cycles

Example:

With $F_{ADC_CLK}$ = 30 MHz and a sampling time of 2.5 ADC clock cycles:

$T_{CONV}$ = (2.5 + 12.5) ADC clock cycles = 15 ADC clock cycles = 500 ns

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

#### Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

#### I/O analog switch voltage booster

The resistance of the I/O analog switches increases when the $V_{DDA}$ voltage is too low. The sampling time must consequently be adapted accordingly (refer to the device datasheet for the corresponding electrical characteristics). This resistance can be minimized at low $V_{DDA}$ voltage by enabling an internal voltage booster through the BOOSTEN bit of the SYSCFG_CFGR1 register.

### 15.3.13 Single conversion mode (CONT = 0)

In Single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

*Note: To convert a single channel, program a sequence with a length of 1.*

### 15.3.14 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

*Note: To convert a single channel, program a sequence with a length of 1.*

*It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.*

*Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section).*

### 15.3.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART = 1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN[1:0] = 00 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN[1:0] is not equal to 00

Software starts ADC injected conversions by setting JADSTART = 1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN[1:0] = 00 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN[1:0] is not equal to 00

*Note: In auto-injection mode (JAUTO = 1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).*

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0 and JADSTART = 0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT = 0, EXTSEL = 0x0)
  - At any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT=x, EXTSEL=x)
  - After execution of the ADSTP procedure asserted by the software.

*Note: In continuous mode (CONT = 1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.*

*When a hardware trigger is selected in single mode (CONT = 0 and EXTSEL≠0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.*

JADSTART is cleared by hardware:

- In single mode with software injected trigger (JEXTSEL = 0x0)
  - At any end of injected conversion sequence (JEOS assertion) or at any end of subgroup processing if JDISCEN = 1
- in all cases (JEXTSEL=x)
  - After execution of the JADSTP procedure asserted by the software.

*Note: When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.*

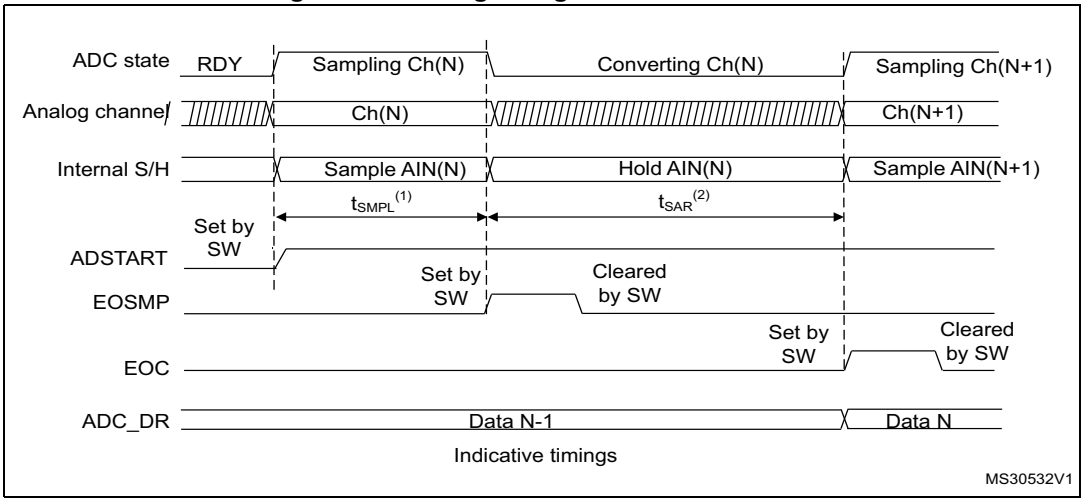
### 15.3.16 ADC timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$$T_{CONV} = T_{SMPL} + T_{SAR} = [2.5\ _{|min} + 12.5\ _{|12bit}] \times T_{ADC_CLK}$$

$$T_{CONV} = T_{SMPL} + T_{SAR} = 83.33\ ns\ _{|min} + 416.67\ ns\ _{|12bit} = 500.0\ ns\ (for\ F_{ADC_CLK} = 30\ MHz)$$

**Figure 39. Analog to digital conversion time**




1. $T_{SMPL}$ depends on SMP[2:0].
2. $T_{SAR}$ depends on RES[2:0].

### 15.3.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP = 1 and injected conversions ongoing by setting JADSTP = 1.

Stopping conversions resets the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would restart a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

*Note: In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).*

**Figure 40. Stopping ongoing regular conversions**

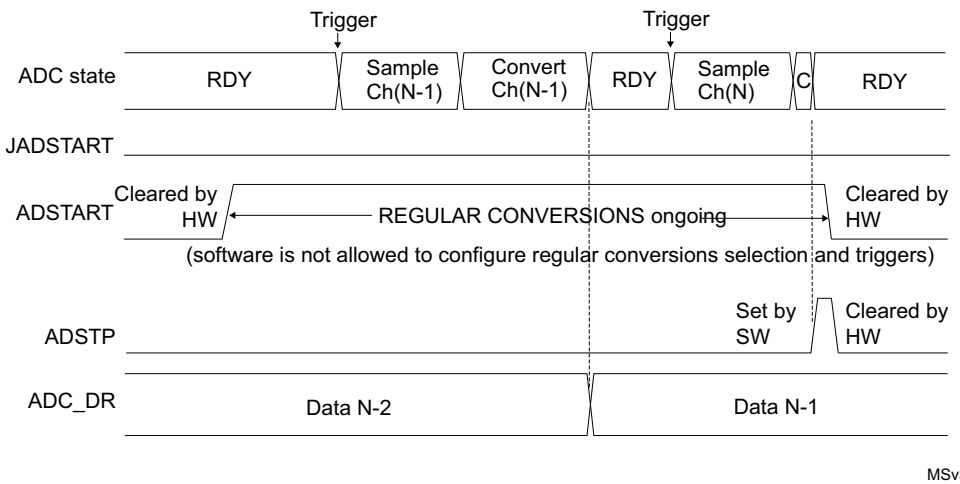
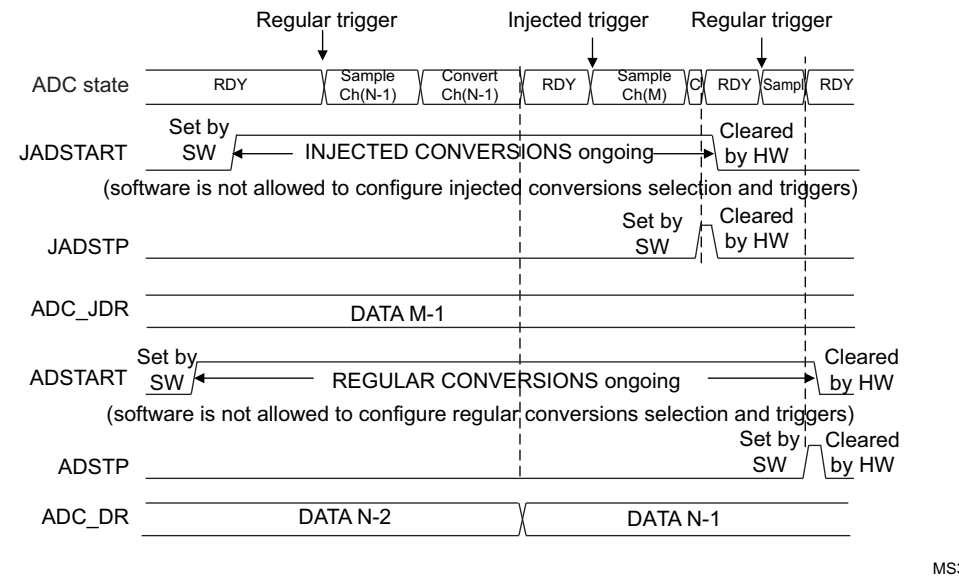



**Figure 41. Stopping ongoing regular and injected conversions**



### 15.3.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (e.g. timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS = 0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART = 1 and the injected trigger selection is effective once software has set bit JADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART = 0, any regular hardware triggers which occur are ignored.
- If bit JADSTART = 0, any injected hardware triggers which occur are ignored.

*Table 65* provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

**Table 65. Configuring the trigger polarity for regular external triggers**

| EXTEN[1:0] | Source |
|---|---|
| 00 | Hardware Trigger detection disabled, software trigger detection enabled |
| 01 | Hardware Trigger with detection on the rising edge |
| 10 | Hardware Trigger with detection on the falling edge |
| 11 | Hardware Trigger with detection on both the rising and falling edges |

*Note: The polarity of the regular trigger cannot be changed on-the-fly.*

**Table 66. Configuring the trigger polarity for injected external triggers**

| JEXTEN[1:0] | Source |
|---|---|
| 00 | – If JQDIS = 1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled<br>– If JQDIS = 0 (Queue enabled), Hardware and software trigger detection disabled |
| 01 | Hardware Trigger with detection on the rising edge |
| 10 | Hardware Trigger with detection on the falling edge |
| 11 | Hardware Trigger with detection on both the rising and falling edges |

*Note: The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS = 0). Refer to Section 15.3.21: Queue of context for injected conversions.*

The EXTSEL and JEXTSEL control bits select which out of 16 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

*Note: The regular trigger selection cannot be changed on-the-fly.*
*The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 15.3.21: Queue of context for injected conversions on page 377*

*Table 67* to *Table 68* give all the possible external triggers of the three ADCs for regular and injected conversions.

**Table 67. ADC1 - External triggers for regular channels**

| Name | Source | Type | EXTSEL[3:0] |
|---|---|---|---|
| EXT0 | TIM1_CC1 | Internal signal from on-chip timers | 0000 |
| EXT1 | TIM1_CC2 | Internal signal from on-chip timers | 0001 |
| EXT2 | TIM1_CC3 | Internal signal from on-chip timers | 0010 |
| EXT3 | TIM2_CC2 | Internal signal from on-chip timers | 0011 |
| EXT4 | - | - | 0100 |
| EXT5 | - | - | 0101 |
| EXT6 | EXTI Line 11 | External pin | 0110 |
| EXT9 | TIM1_TRGO | Internal signal from on-chip timers | 1001 |
| EXT10 | TIM1_TRGO2 | Internal signal from on-chip timers | 1010 |
| EXT11 | TIM2_TRGO | Internal signal from on-chip timers | 1011 |
| EXT13 | - | - | 1101 |
| EXT14 | - | - | 1110 |

**Table 68. ADC1 - External trigger for injected channels**

| Name | Source | Type | JEXTSEL[3:0] |
|---|---|---|---|
| JEXT0 | TIM1_TRGO | Internal signal from on-chip timers | 0000 |
| JEXT1 | TIM1_CC4 | Internal signal from on-chip timers | 0001 |
| JEXT2 | TIM2_TRGO | Internal signal from on-chip timers | 0010 |
| JEXT3 | TIM2_CH1 | Internal signal from on-chip timers | 0011 |
| JEXT4 | - | - | 0100 |
| JEXT5 | - | - | 0101 |
| JEXT6 | EXTI Line 15 | External pin | 0110 |
| JEXT8 | TIM1_TRGO2 | Internal signal from on-chip timers | 1000 |
| JEXT14 | - | - | 1110 |
| JEXT15 | - | - | 1111 |

### 15.3.19 Injected channel management

#### Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 42* shows the corresponding timing diagram.

*Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 2.5 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.*

#### Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

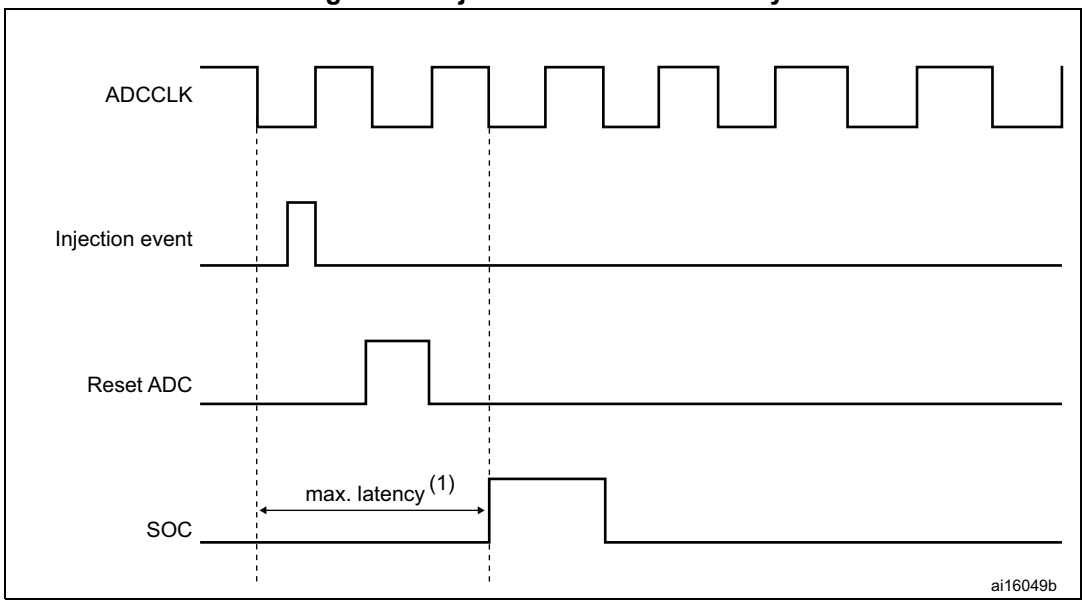
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

*Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.*

*When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence is stopped upon DMA Transfer Complete event.*

**Figure 42. Injected conversion latency**




1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

## 15.3.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

### Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

*Note: The channel numbers referred to in the above example might not be available on all microcontrollers.*

*When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).*

*When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.*

*It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.*

#### Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where 'n' is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

*Note: The channel numbers referred to in the above example might not be available on all microcontrollers.*

*When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.*

*It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.*

### 15.3.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)

All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM = 0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence is served according to the last active context.
  - If JQM = 1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP = 1 or when disabling the ADC by setting ADDIS = 1:
  - If JQM = 0, the Queue is maintained with the last active context.
  - If JQM = 1, the Queue becomes empty and triggers are ignored.

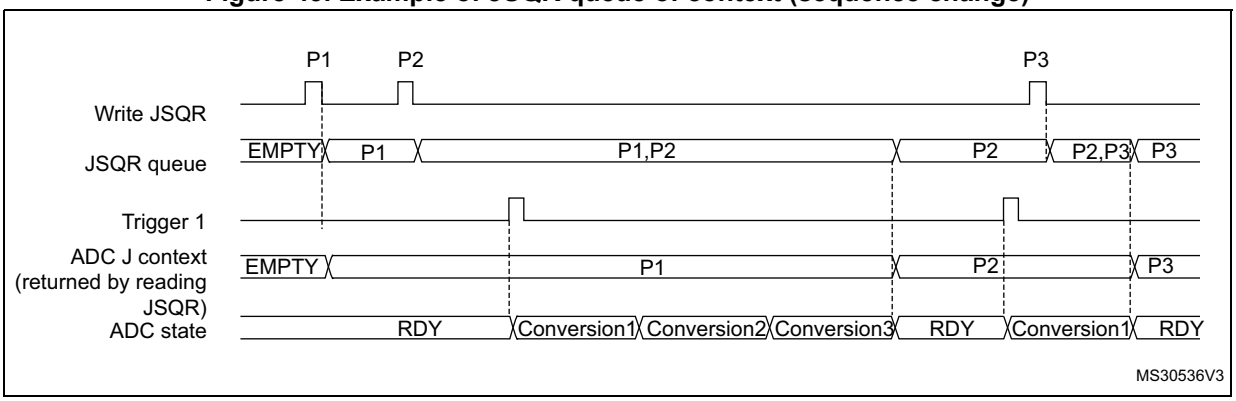
*Note: When configured in discontinuous mode (bit JDISCEN = 1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):*

- *1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out*
- *2nd trigger, discontinuous. Sequence 1: 2nd conversion.*
- *3rd trigger, discontinuous. Sequence 1: 3rd conversion.*
- *4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.*
- *5th trigger, discontinuous. Sequence 2: 2nd conversion.*
- *6th trigger, discontinuous. Sequence 2: 3rd conversion.*

### Behavior when changing the trigger or sequence context

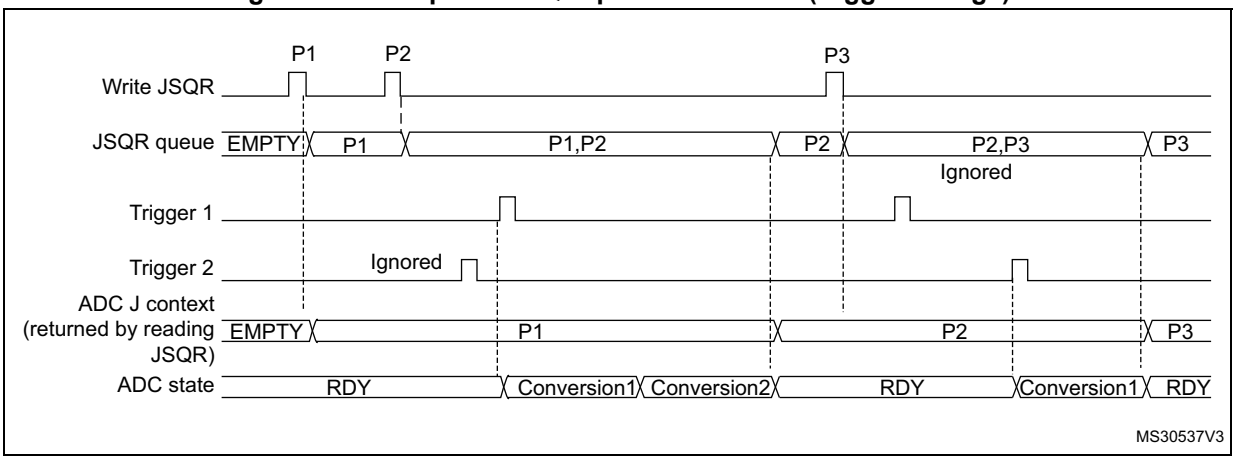
The *Figure 43* and *Figure 44* show the behavior of the context Queue when changing the sequence or the triggers.

**Figure 43. Example of JSQR queue of context (sequence change)**




1. Parameters:
   P1: sequence of 3 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 4 conversions, hardware trigger 1

**Figure 44. Example of JSQR queue of context (trigger change)**

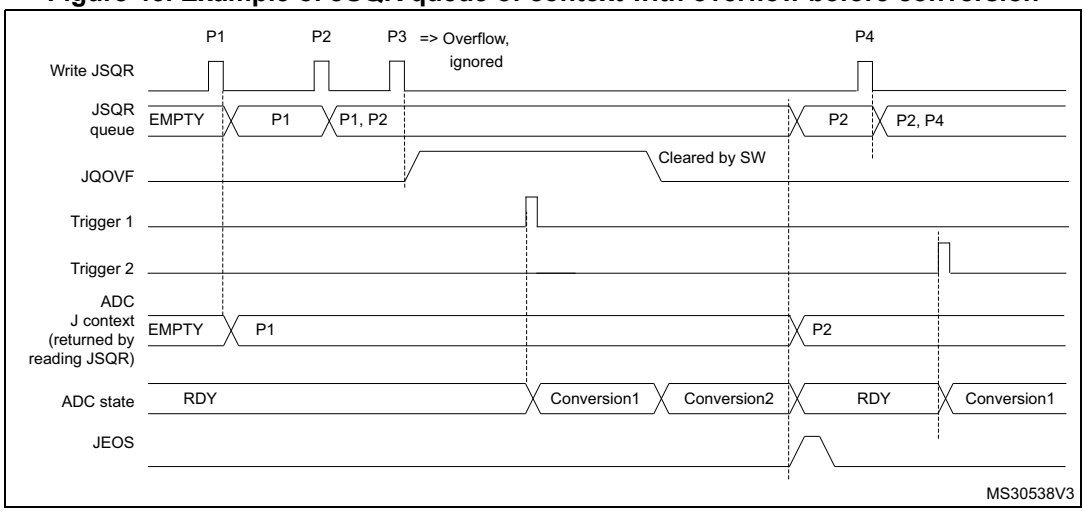



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

### Queue of context: Behavior when a queue overflow occurs

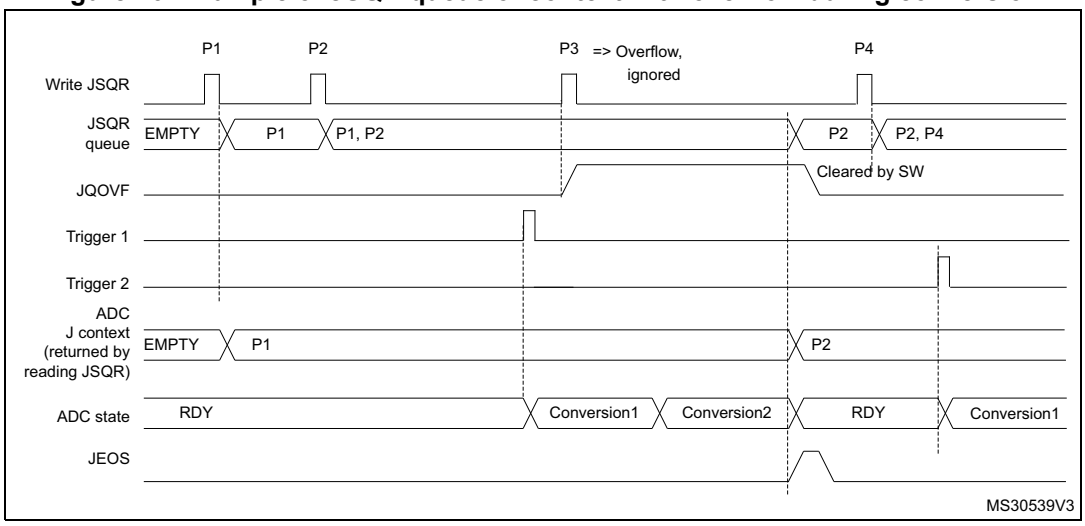
The *Figure 45* and *Figure 46* show the behavior of the context Queue if an overflow occurs before or during a conversion.

**Figure 45. Example of JSQR queue of context with overflow before conversion**




1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

**Figure 46. Example of JSQR queue of context with overflow during conversion**




1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

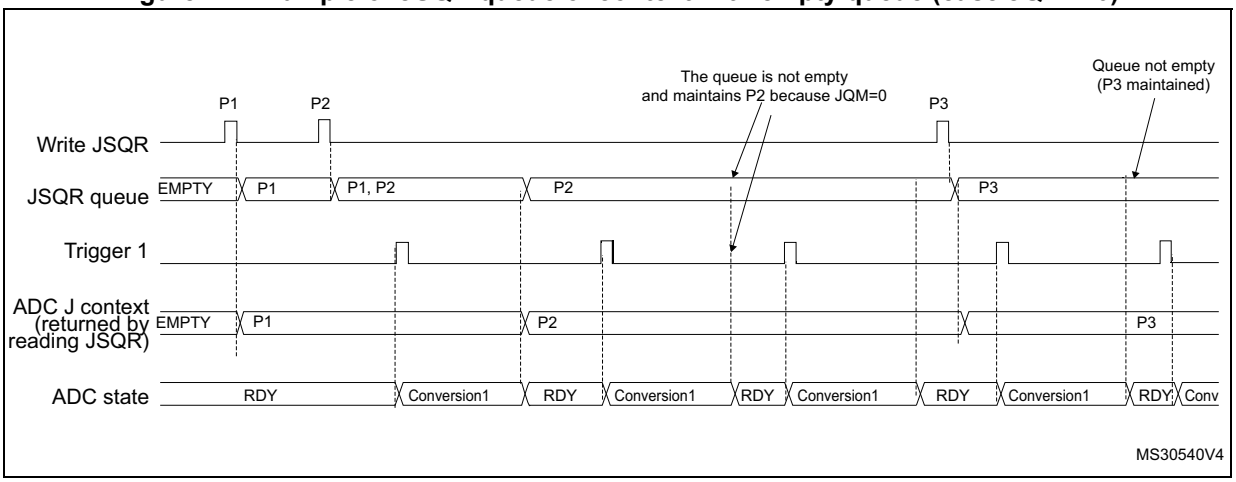
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

### Queue of context: Behavior when the queue becomes empty

*Figure 47* and *Figure 48* show the behavior of the context Queue when the Queue becomes empty in both cases JQM = 0 or 1.

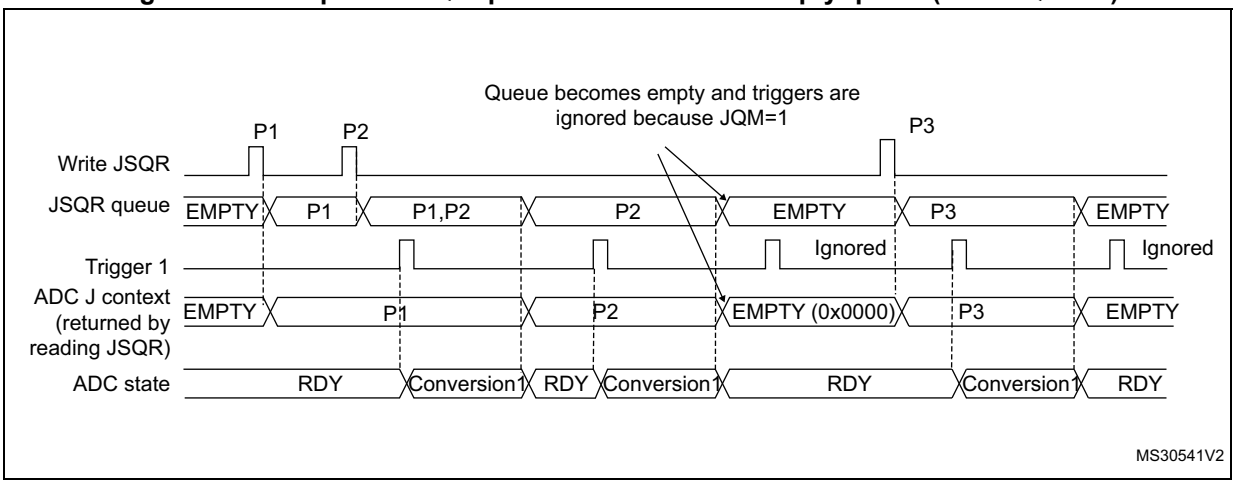
**Figure 47. Example of JSQR queue of context with empty queue (case JQM = 0)**




1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

*Note: When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.*

**Figure 48. Example of JSQR queue of context with empty queue (case JQM = 1)**

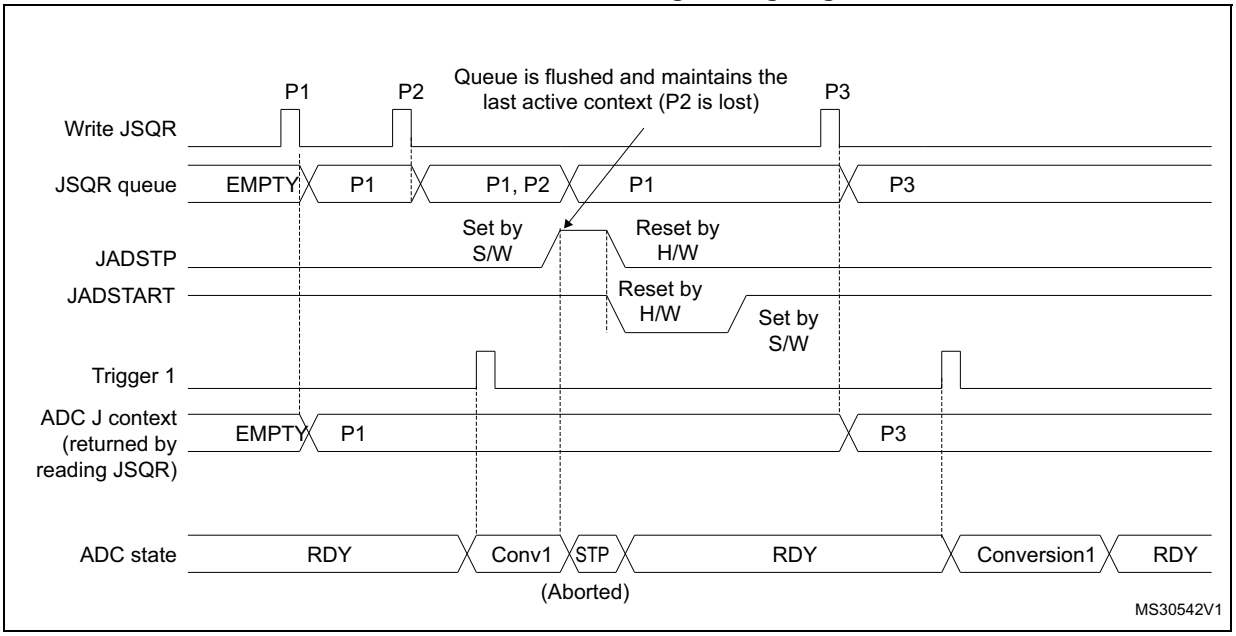



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

### Flushing the queue of context

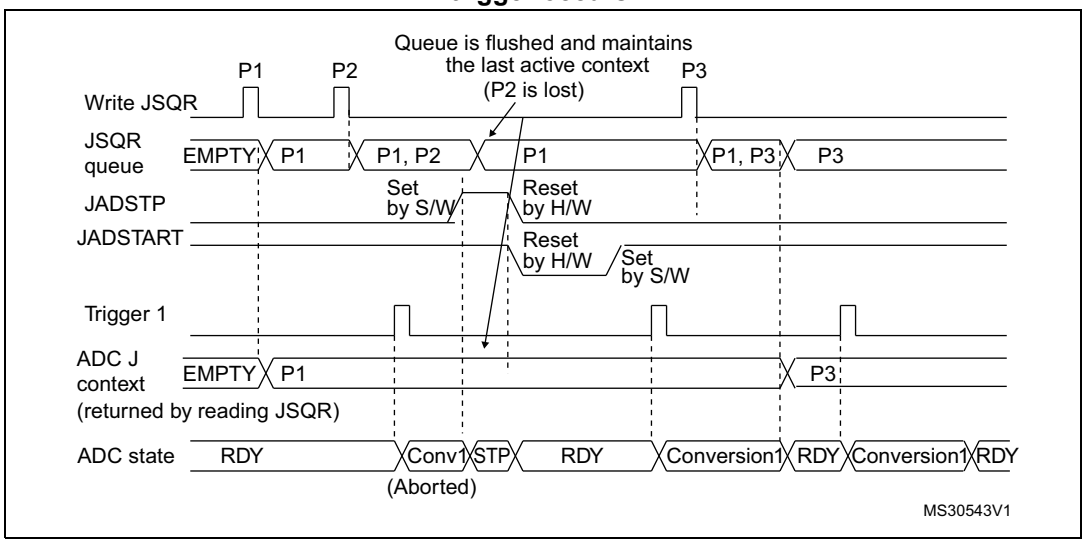
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

**Figure 49. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion**



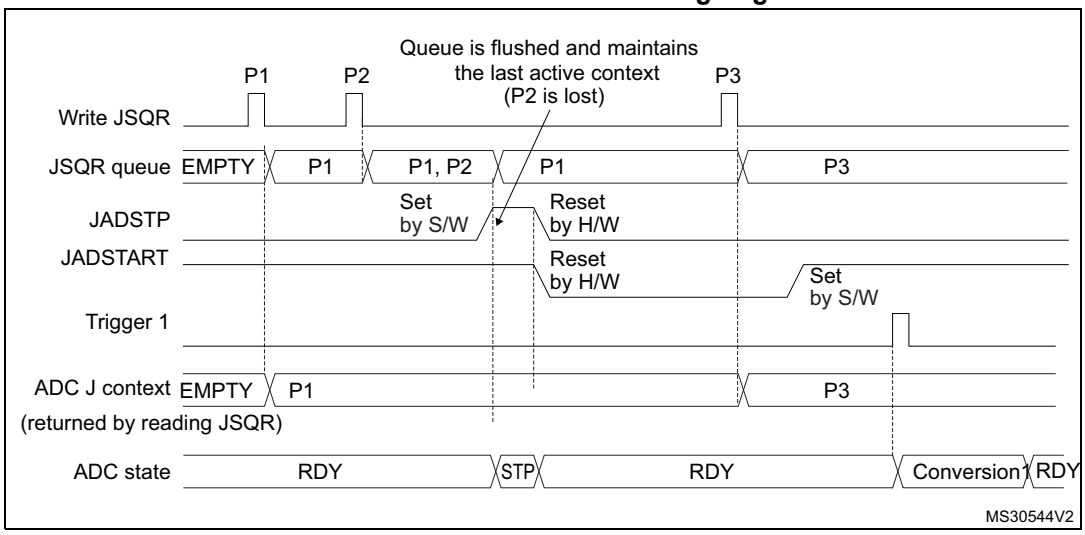

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

**Figure 50. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.**



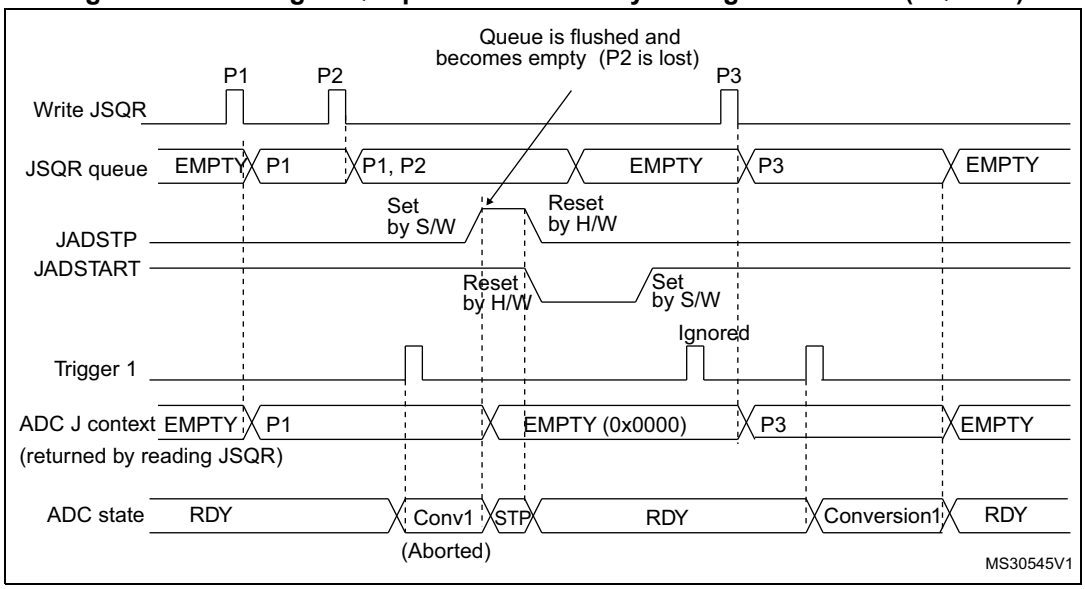

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

**Figure 51. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs outside an ongoing conversion**



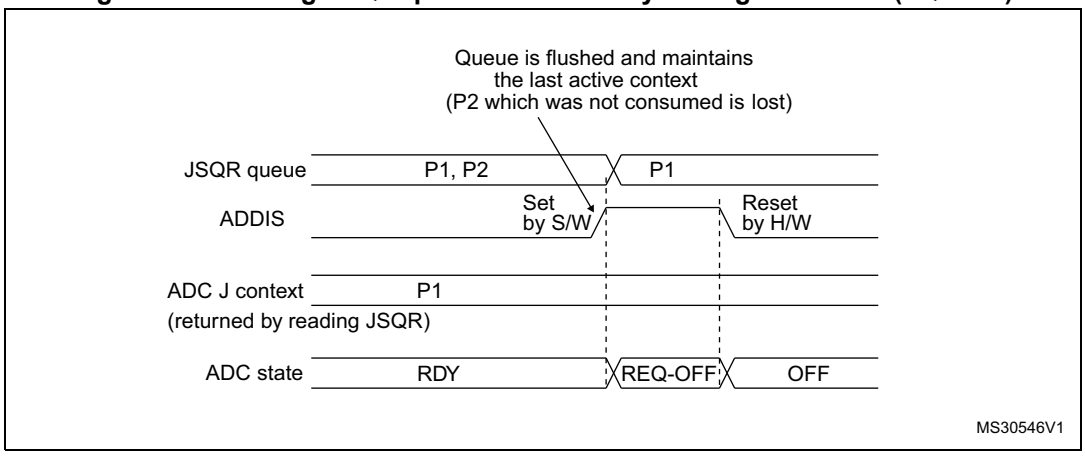

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

**Figure 52. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1)**



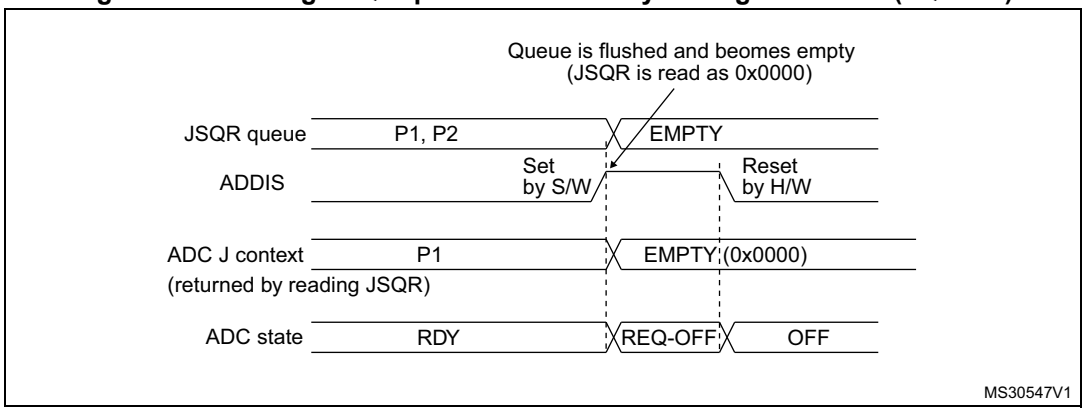

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

**Figure 53. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0)**




1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

**Figure 54. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1)**




1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

### Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN[1:0] not equal to 00 (otherwise triggering a software conversion).
6. Set JADSTART.
7. Set JADSTP.
8. Wait until JADSTART is reset.
9. Set JADSTART.

### Disabling the queue

It is possible to disable the queue by setting bit JQDIS = 1 into the ADC_CFGR register.

## 15.3.22 Programmable resolution (RES) - Fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. *Figure 59*, *Figure 60*, *Figure 61* and *Figure 62* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 69*.

**Table 69. $T_{SAR}$ timings depending on resolution**

| RES (bits) | $T_{SAR}$ (ADC clock cycles) | $T_{SAR}$ (ns) at $F_{ADC}$= 30 MHz | $T_{CONV}$ (ADC clock cycles) (with Sampling Time= 2.5 ADC clock cycles) | $T_{CONV}$ (ns) at $F_{ADC}$= 30 MHz |
|---|---|---|---|---|
| 12 | 12.5 ADC clock cycles | 416.67 ns | 15 ADC clock cycles | 500.0 ns |
| 10 | 10.5 ADC clock cycles | 350.0 ns | 13 ADC clock cycles | 433.33 ns |
| 8 | 8.5 ADC clock cycles | 203.33 ns | 11 ADC clock cycles | 366.67 ns |
| 6 | 6.5 ADC clock cycles | 216.67 ns | 9 ADC clock cycles | 300.0 ns |

### 15.3.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

### 15.3.24 End of conversion sequence (EOS, JEOS)

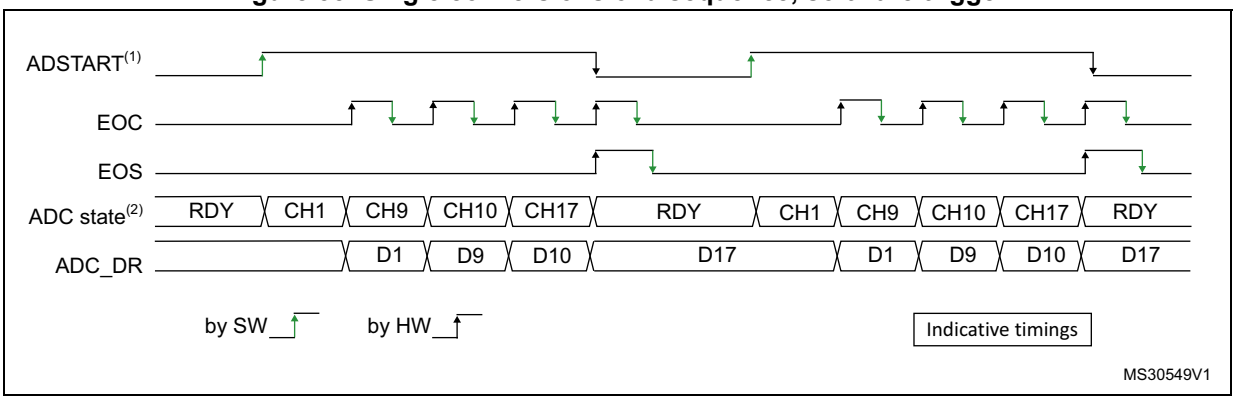
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

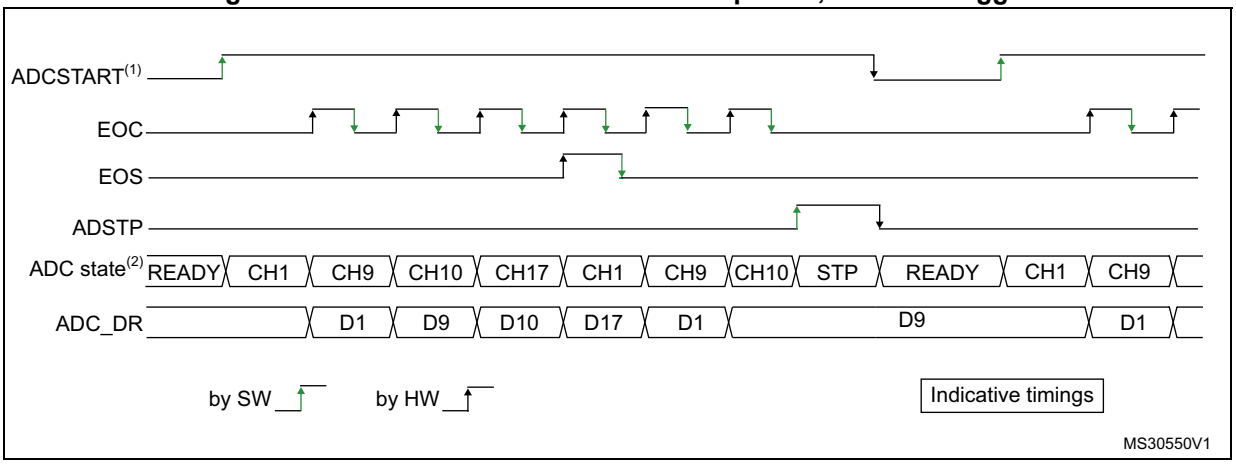
### 15.3.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

**Figure 55. Single conversions of a sequence, software trigger**



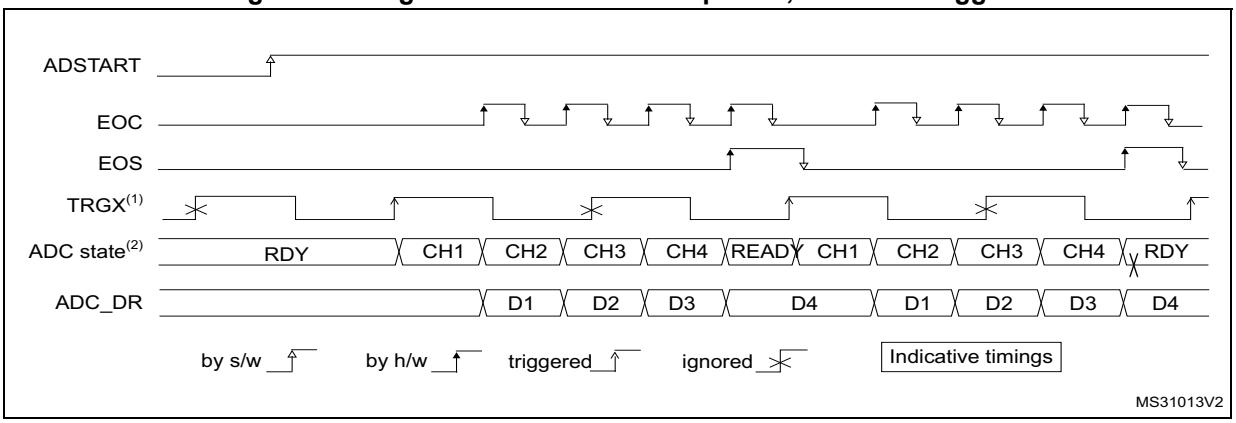

1. EXTEN[1:0] = 00, CONT = 0
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

**Figure 56. Continuous conversion of a sequence, software trigger**



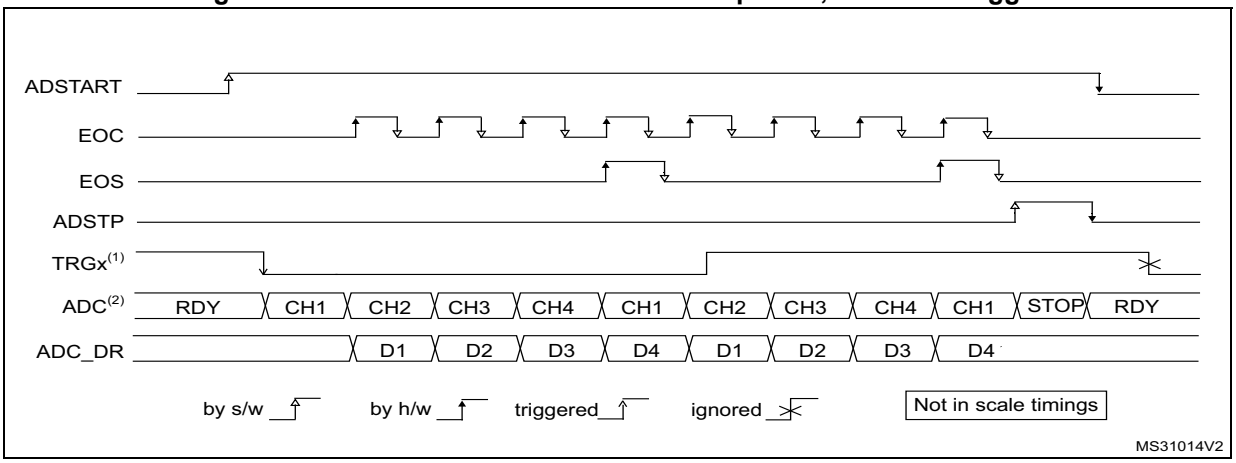

1. EXTEN[1:0] = 00, CONT = 1
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

**Figure 57. Single conversions of a sequence, hardware trigger**




1. TRGx (over-frequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

**Figure 58. Continuous conversions of a sequence, hardware trigger**




1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

### 15.3.26 Data management

#### Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)

#### Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADC_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 59*, *Figure 60*, *Figure 61* and *Figure 62*.

Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. In that case, the data are aligned on a byte basis as shown in *Figure 61* and *Figure 62*.

*Note: Left-alignment is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the ALIGN bit value is ignored and the ADC only provides right-aligned data.*

#### Offset

An offset y (y = 1,2,3,4) can be applied to a channel by setting the bit OFFSETy_EN = 1 into ADC_OFRy register. The channel to which the offset is applied is programmed into the bits OFFSETy_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSETy[11:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

*Note: Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy_EN bit in ADC_OFRy register is ignored (considered as reset).*

*Table 72* describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

**Table 70. Offset computation versus data resolution**

| Resolution (bits RES[1:0]) | Subtraction between raw converted data and offset | | Result | Comments |
|---|---|---|---|---|
| | Raw converted Data, left aligned | Offset | | |
| 00: 12-bit | DATA[11:0] | OFFSET[11:0] | Signed 12-bit data | - |
| 01: 10-bit | DATA[11:2],00 | OFFSET[11:0] | Signed 10-bit data | The user must configure OFFSET[1:0] to 00 |

**Table 70. Offset computation versus data resolution (continued)**

| Resolution (bits RES[1:0]) | Subtraction between raw converted data and offset | | Result | Comments |
|---|---|---|---|---|
| | Raw converted Data, left aligned | Offset | | |
| 10: 8-bit | DATA[11:4],0000 | OFFSET[11:0] | Signed 8-bit data | The user must configure OFFSET[3:0] to 0000 |
| 11: 6-bit | DATA[11:6],000000 | OFFSET[11:0] | Signed 6-bit data | The user must configure OFFSET[5:0] to 000000 |

When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y = 1,2,3,4) corresponding to the channel “i”:

- If one of the offsets is enabled (bit OFFSETy_EN = 1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

*Figure 59*, *Figure 60*, *Figure 61* and *Figure 62* show alignments for signed and unsigned data.

**Figure 59. Right alignment (offset disabled, unsigned value)**

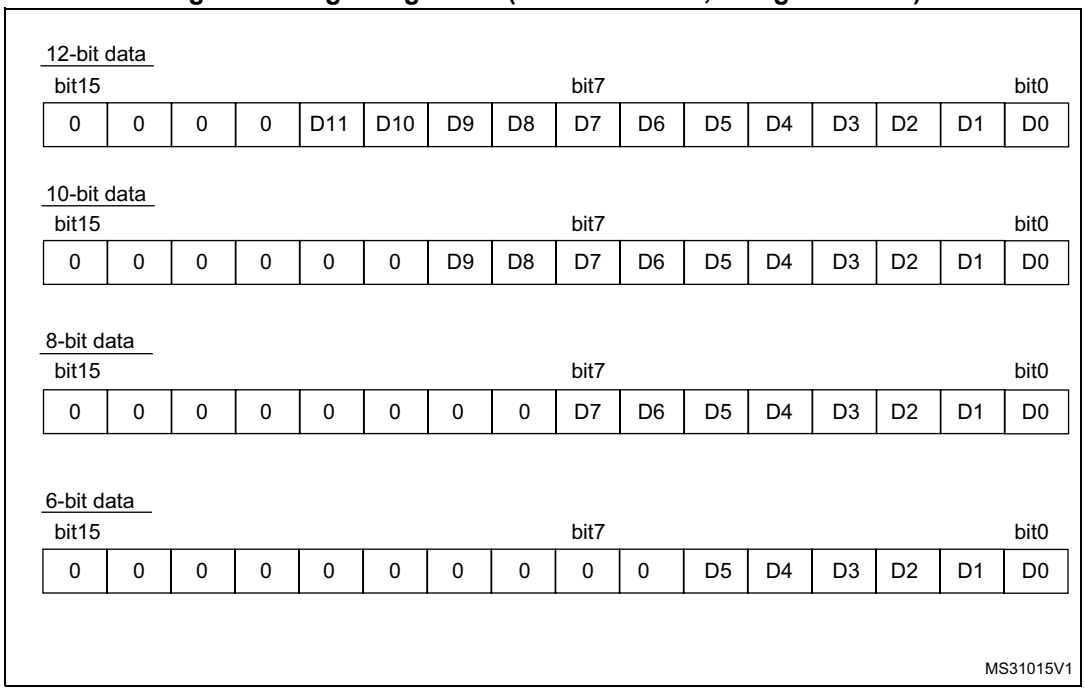


**Figure 60. Right alignment (offset enabled, signed value)**

12-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | SEXT | SEXT | SEXT | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

10-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

8-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

6-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | D5 | D4 | D3 | D2 | D1 | D0 |

MS31016V1

**Figure 61. Left alignment (offset disabled, unsigned value)**

12-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 |

10-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 | 0 | 0 |

8-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

6-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 |

MS31017V1

**Figure 62. Left alignment (offset enabled, signed value)**

12-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 |

10-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 | 0 |

8-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

6-bit data

| bit15 | | | | | | | | bit7 | | | | | | | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | SEXT | D5 | D4 | D3 | D2 | D1 | D0 | 0 |

MS31018V1

## ADC overrun (OVR, OVRMOD)

The overrun flag (OSR) notifies of that a buffer overrun event occurred when the regular converted data has not been read (by the CPU or the DMA) before new converted data became available.

The OVR flag is set if the EOC flag is still 1 at the time when a new conversion completes. An interrupt can be generated if bit OVRIE = 1.

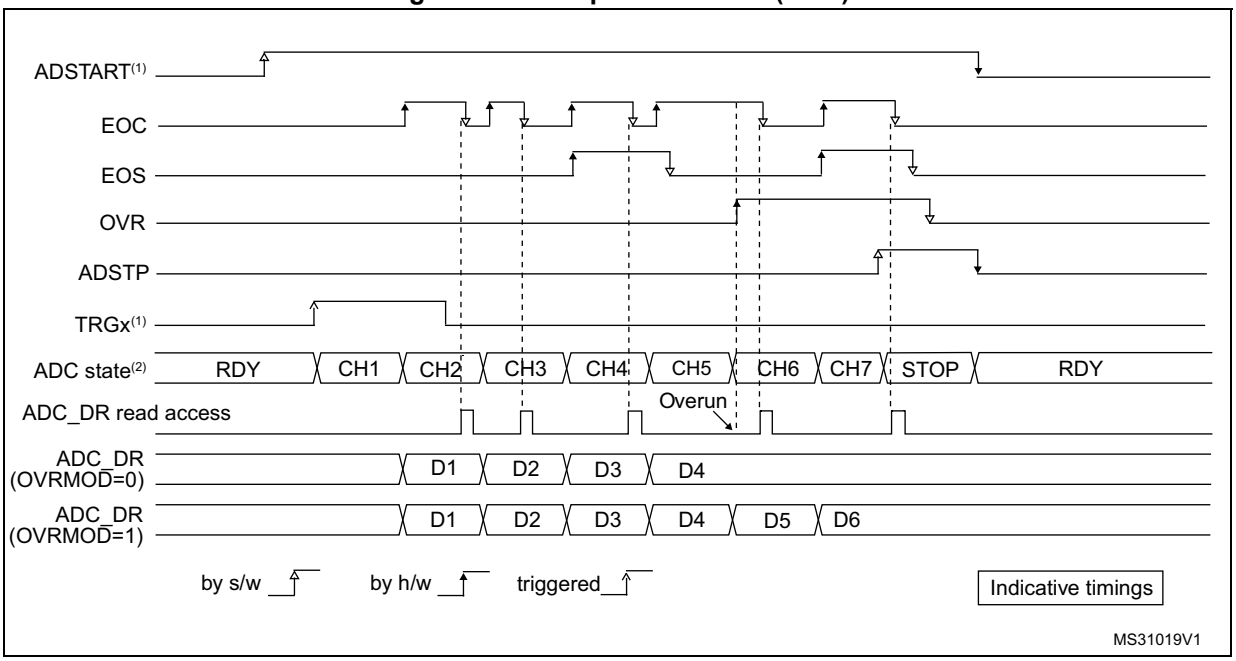
When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting bit ADSTP = 1.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0: The overrun event preserves the data register from being overrun: the old data is maintained and the new conversion is discarded and lost. If OVR remains at 1, any further conversions occur but the result data is also discarded.
- OVRMOD = 1: The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, any further conversions operate normally and the ADC_DR register always contains the latest converted data.

**Figure 63. Example of overrun (OVR)**




*Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.*

### Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD should be configured to 0 to manage overrun events as an error.

### Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

### Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMAEN bit set in the ADC_CFGR register, a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if an overrun occurs (OVR = 1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG of the ADC_CFGR register:

- DMA one shot mode (DMACFG = 0).
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMACFG = 1)
  This mode is suitable when programming the DMA in circular mode.

#### DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs - refer to DMA section) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

#### DMA circular mode (DMACFG = 1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

### 15.3.27 Dynamic low-power features

#### Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY = 1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see *Figure 64*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 65*).

This is a way to automatically adapt the speed of the ADC to the speed of the system which reads the data.

The delay is inserted after each regular conversion (whatever DISCEN = 0 or 1) and after each sequence of injected conversions (whatever JDISCEN = 0 or 1).

*Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.*

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

*Note: This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to restart a conversion: it is up to the software to read the data before launching a new conversion.*

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see *Figure 65*).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see *Figure 67*).

The behavior is slightly different in auto-injected mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 68*).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO = 1, CONT = 1 and AUTDLY = 1), follow the following procedure:

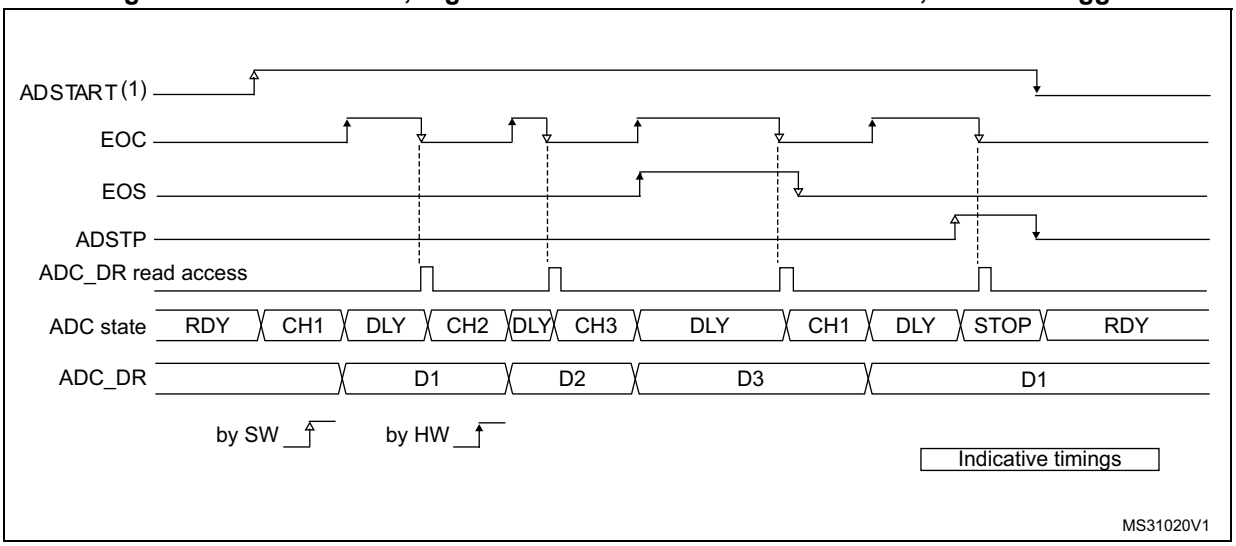
1. Wait until JEOS = 1 (no more conversions are restarted)
2. Clear JEOS.
3. Set ADSTP.
4. Read the regular data.

If this procedure is not respected, a new regular sequence can restart if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

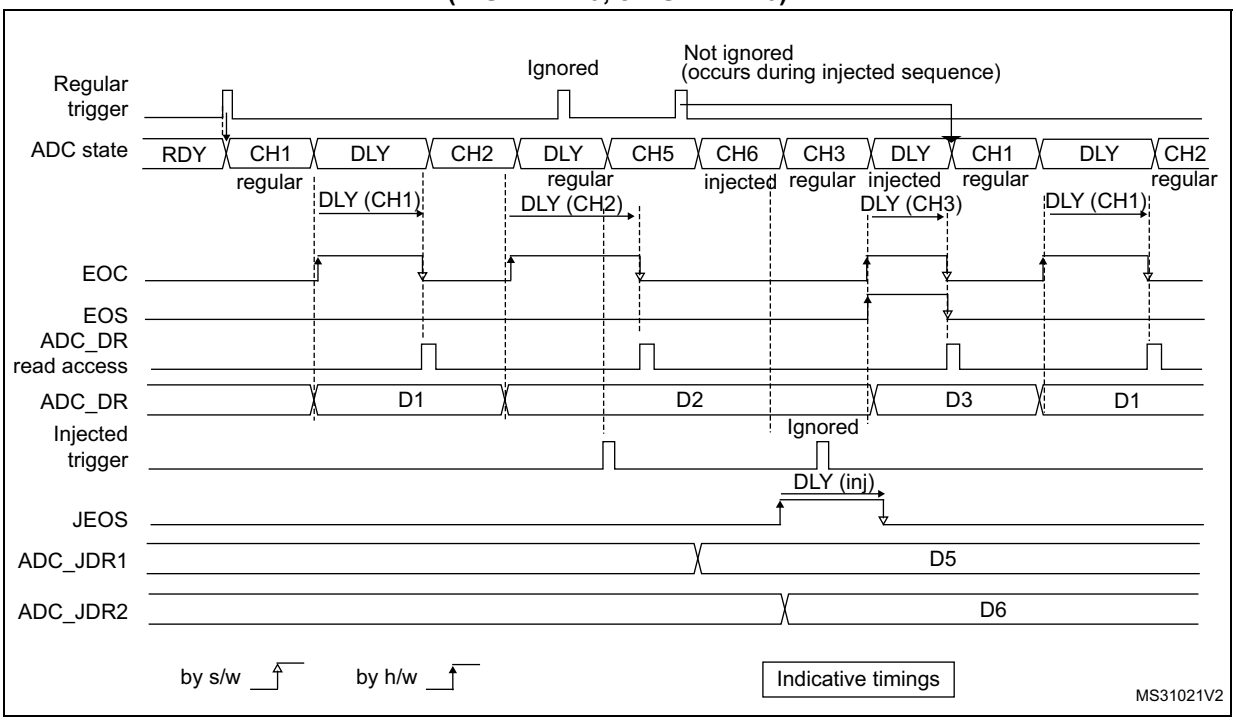
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

**Figure 64. AUTODLY = 1, regular conversion in continuous mode, software trigger**



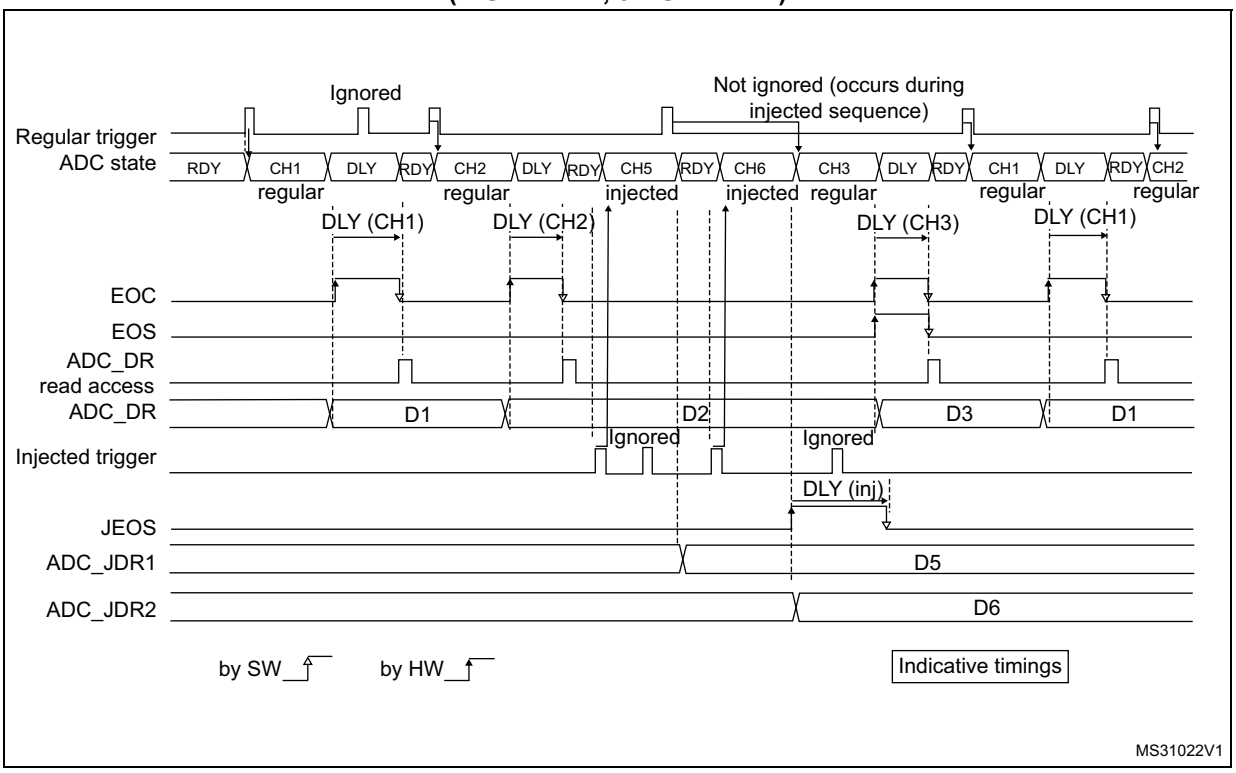

1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, CHANNELS = 1,2,3.
3. Injected configuration DISABLED.

**Figure 65. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)**



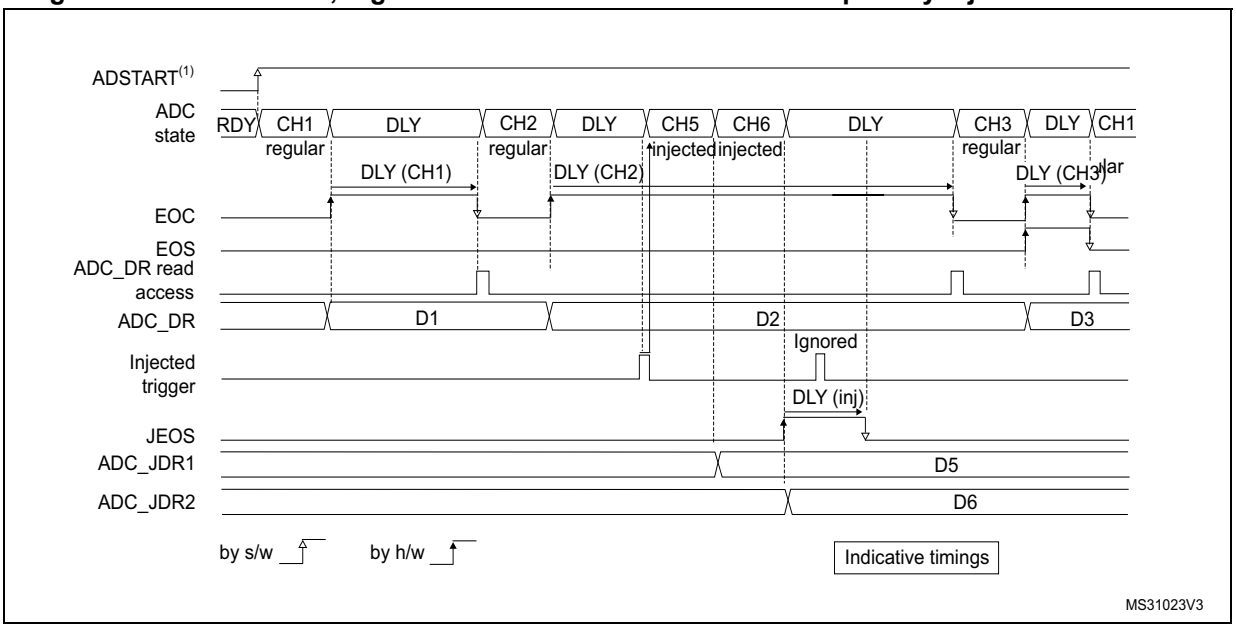

1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 01 (HW trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN[1:0] = 01 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

**Figure 66. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)**



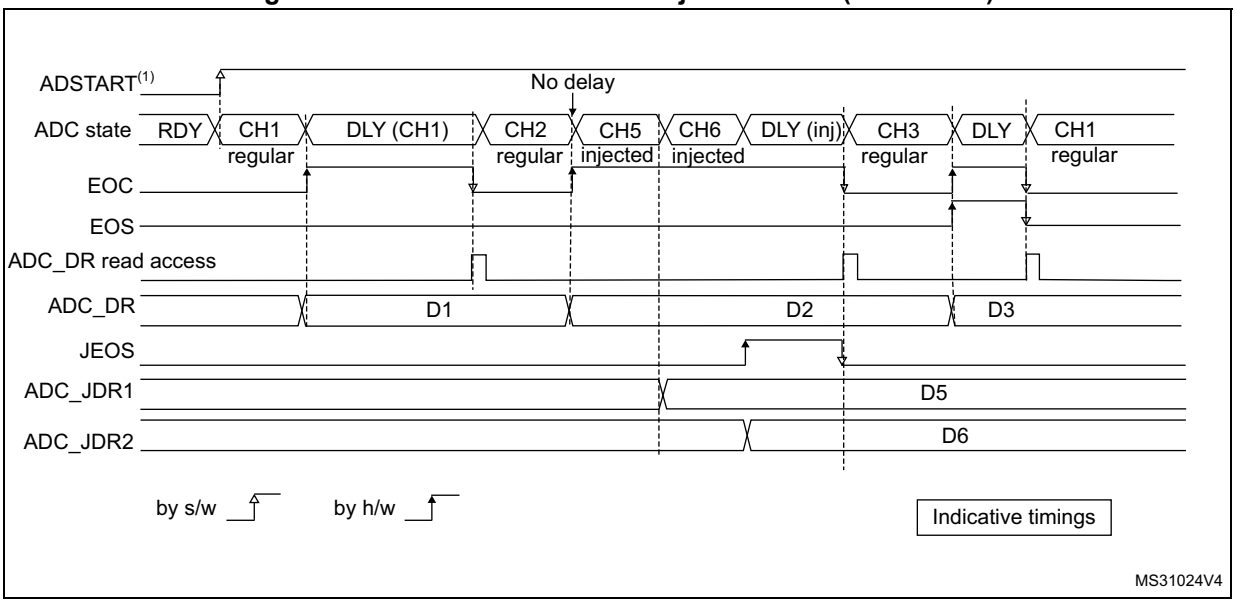

1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 01 (HW trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (HW Trigger), JDISCEN = 1, CHANNELS = 5,6

**Figure 67. AUTODLY = 1, regular continuous conversions interrupted by injected conversions**




1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN[1:0] = 01 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

**Figure 68. AUTODLY = 1 in auto- injected mode (JAUTO = 1)**

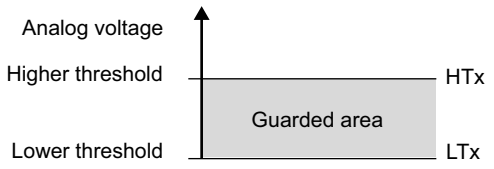



1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2
3. Injected configuration: JAUTO = 1, CHANNELS = 5,6

### 15.3.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

**Figure 69. Analog watchdog guarded area**




#### AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADC_IER register (x = 1,2,3).

AWDx (x = 1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

**Description of analog watchdog 1**

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels(1) remain within a configured voltage range (window).

*Table 71* shows how the ADC_CFGR registers should be configured to enable the analog watchdog on one or more channels.

**Table 71. Analog watchdog channel selection**

| Channels guarded by the analog watchdog | AWD1SGL bit | AWD1EN bit | JAWD1EN bit |
|---|---|---|---|
| None | x | 0 | 0 |
| All injected channels | 0 | 0 | 1 |
| All regular channels | 0 | 1 | 0 |
| All regular and injected channels | 0 | 1 | 1 |
| Single(1) injected channel | 1 | 0 | 1 |
| Single(1) regular channel | 1 | 1 | 0 |
| Single(1) regular or injected channel | 1 | 1 | 1 |

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed in bits HT1[11:0] and LT1[11:0] of the ADC_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

*Table 72* describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

**Table 72. Analog watchdog 1 comparison**

| Resolution( bit RES[1:0]) | Analog watchdog comparison between: | | Comments |
|---|---|---|---|
| | Raw converted data, left aligned | Thresholds | |
| 00: 12-bit | DATA[11:0] | LT1[11:0] and HT1[11:0] | - |
| 01: 10-bit | DATA[11:2],00 | LT1[11:0] and HT1[11:0] | User must configure LT1[1:0] and HT1[1:0] to 00 |
| 10: 8-bit | DATA[11:4],0000 | LT1[11:0] and HT1[11:0] | User must configure LT1[3:0] and HT1[3:0] to 0000 |
| 11: 6-bit | DATA[11:6],000000 | LT1[11:0] and HT1[11:0] | User must configure LT1[5:0] and HT1[5:0] to 000000 |

### Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDxCH[18:0] (x=2,3).

The corresponding watchdog is enabled when any bit of AWDxCH[18:0] (x=2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. *Table 73* describes how the comparison is performed for all the possible resolutions.

**Table 73. Analog watchdog 2 and 3 comparison**

| Resolution (bits RES[1:0]) | Analog watchdog comparison between: | | Comments |
|---|---|---|---|
| | Raw converted data, left aligned | Thresholds | |
| 00: 12-bit | DATA[11:4] | LTx[7:0] and HTx[7:0] | DATA[3:0] are not relevant for the comparison |
| 01: 10-bit | DATA[11:4] | LTx[7:0] and HTx[7:0] | DATA[3:2] are not relevant for the comparison |
| 10: 8-bit | DATA[11:4] | LTx[7:0] and HTx[7:0] | - |
| 11: 6-bit | DATA[11:6],00 | LTx[7:0] and HTx[7:0] | User must configure LTx[1:0] and HTx[1:0] to 00 |

### ADCy_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y=ADC number, x=watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCy_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS = 1). Note that stopping regular or injected conversions (setting ADSTP = 1 or JADSTP = 1) has no influence on the generation of ADCy_AWDx_OUT.

*Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).*

**Figure 70. ADCy_AWDx_OUT signal generation (on all regular channels)**

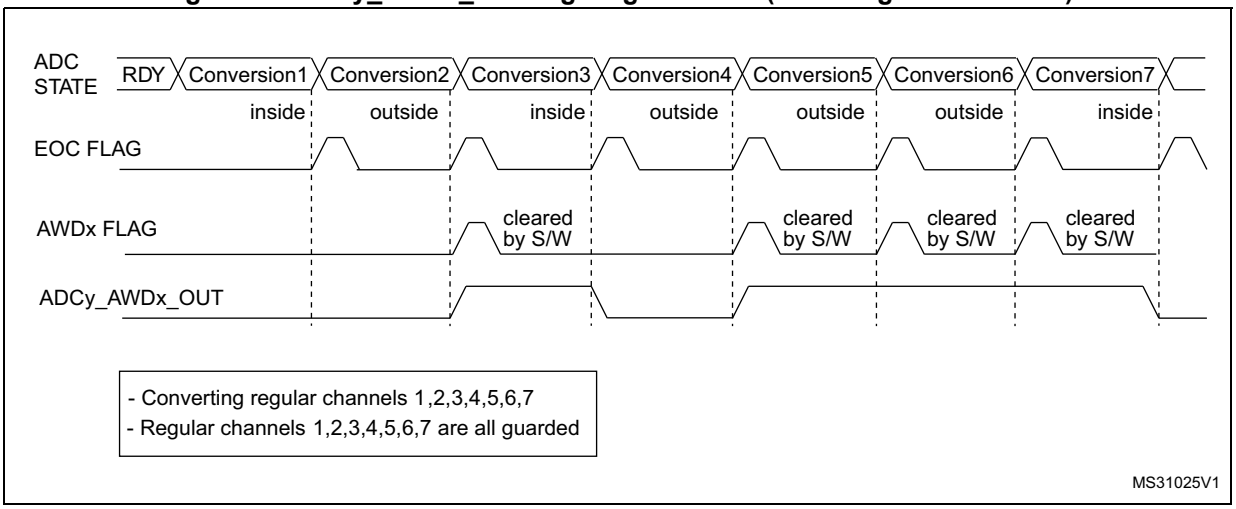


**Figure 71. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software)**

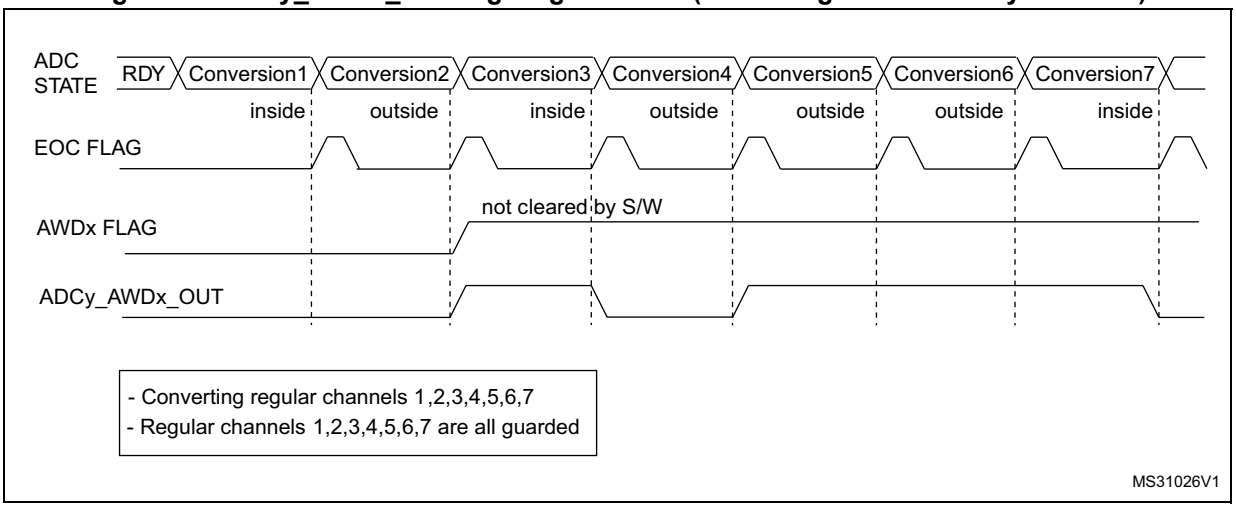



**Figure 72. ADCy_AWDx_OUT signal generation (on a single regular channel)**

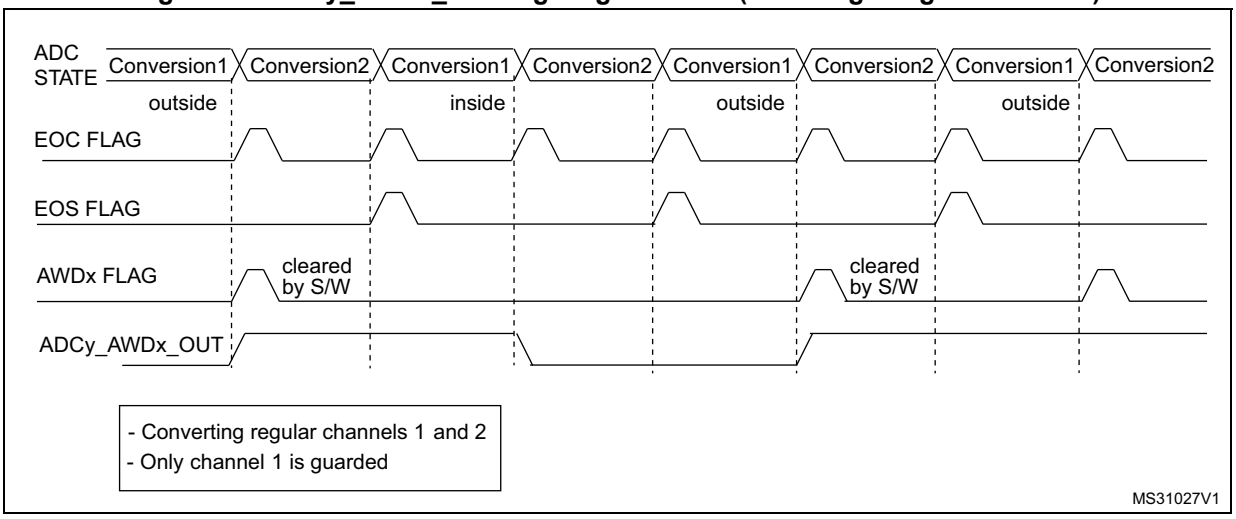
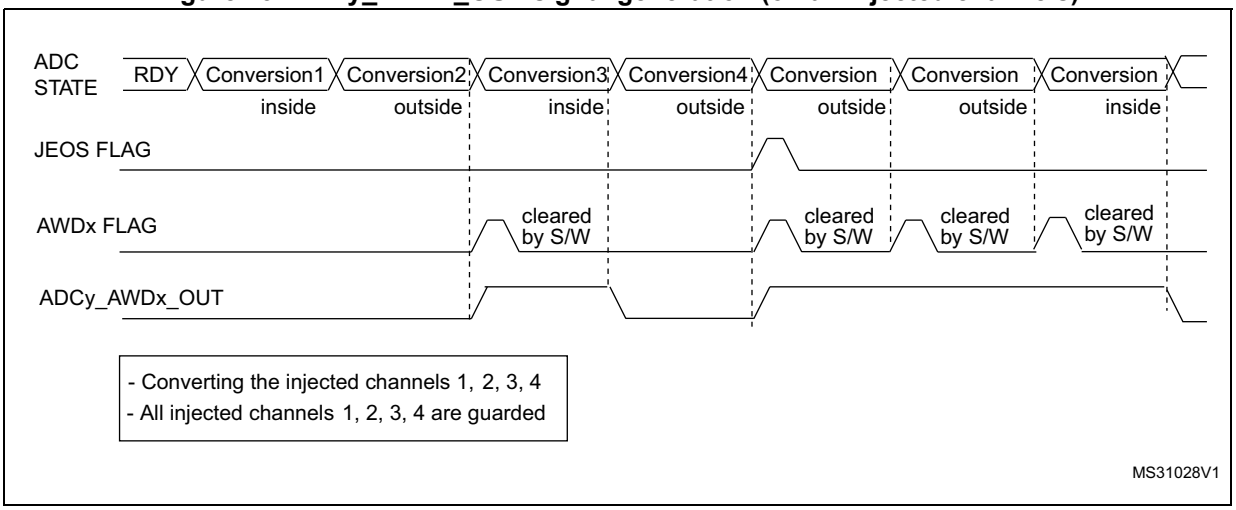



**Figure 73. ADCy_AWDx_OUT signal generation (on all injected channels)**



### 15.3.29 Oversampler

The oversampling unit performs data pre-processing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC_CFGR2 register, and can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to the 16 least significant bits, rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

*Note: If the intermediary result after the shifting exceeds 16-bit, the result is truncated as is, without saturation.*

**Figure 74. 20-bit to 16-bit result truncation**

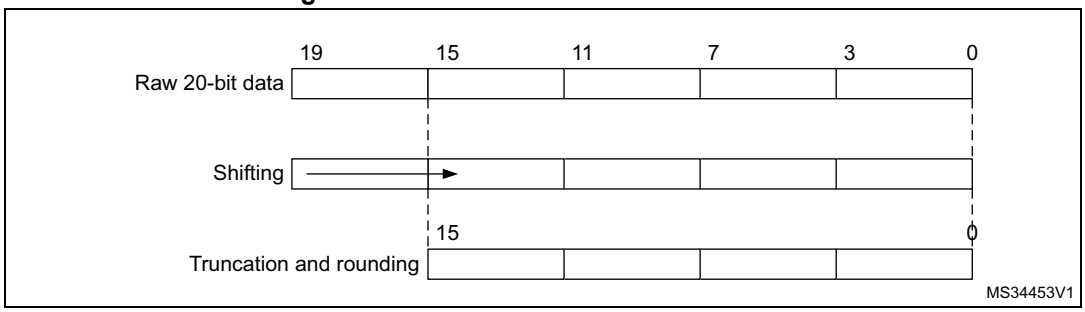



*Figure 75* gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 75. Numerical example with 5-bit shift and rounding**

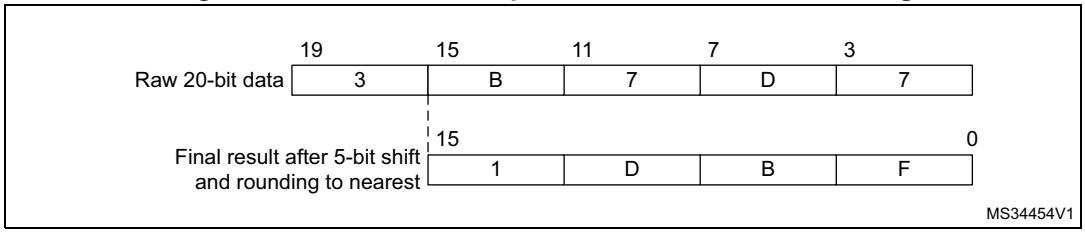



*Table 74* gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

**Table 74. Maximum output results versus N and M (gray cells indicate truncation)**

| Over sampling ratio | Max Raw data | No-shift OVSS = 0000 | 1-bit shift OVSS = 0001 | 2-bit shift OVSS = 0010 | 3-bit shift OVSS = 0011 | 4-bit shift OVSS = 0100 | 5-bit shift OVSS = 0101 | 6-bit shift OVSS = 0110 | 7-bit shift OVSS = 0111 | 8-bit shift OVSS = 1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2x | 0x1FFE | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 | 0x0080 | 0x0040 | 0x020 |
| 4x | 0x3FFC | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 | 0x0080 | 0x0040 |
| 8x | 0x7FF8 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 | 0x0080 |
| 16x | 0xFFF0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 |
| 32x | 0x1FFE0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 |
| 64x | 0x3FFC0 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 |
| 128x | 0x7FF80 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 |
| 256x | 0xFFF00 | 0xFF00 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF |

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversions, with an equivalent delay equal to N x $T_{CONV}$ = N x ($t_{SMPL}$ + $t_{SAR}$). The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase
- The end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (i.e. after N x sequence length conversions total)

### ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous mode conversions
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

*Note: The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR1 is ignored and the data are always provided right-aligned.*

*Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy_EN bit in ADC_OFRy register is ignored (considered as reset).*

### Analog watchdog

The analog watchdog functionality is maintained, with the following difference:

- The RES[1:0] bits are ignored, comparison is always done using the full 12-bit values HT[11:0] and LT[11:0]
- the comparison is performed on the most significant 12-bit of the 16-bit oversampled results ADC_DR[15:4]

*Note: Care must be taken when using high shifting values, since this reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HT[0:7] / LT[[0:7], and HT[11:8] / LT[11:8] must be kept reset.*

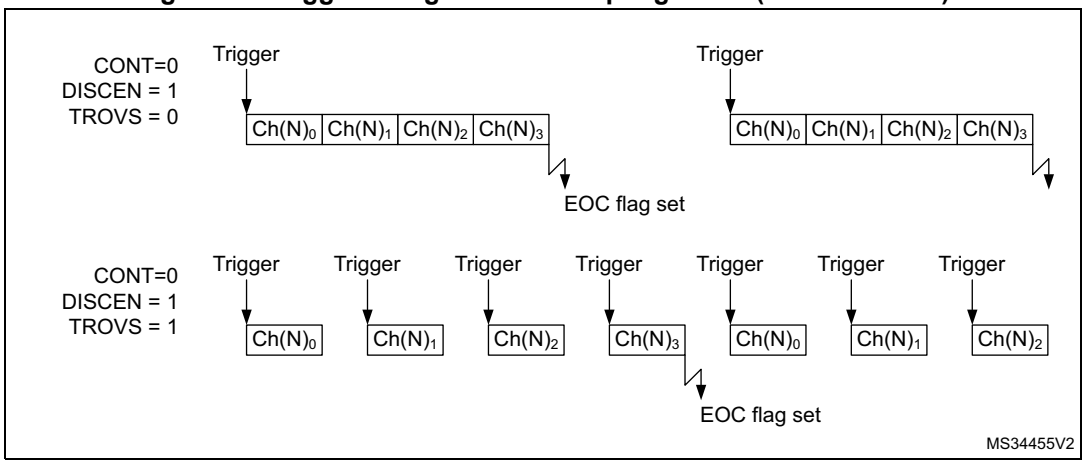
### Triggered mode

The averager can also be used for basic filtering purpose. Although not a very powerful filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

*Figure 76* below shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

**Figure 76. Triggered regular oversampling mode (TROVS bit = 1)**




### Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

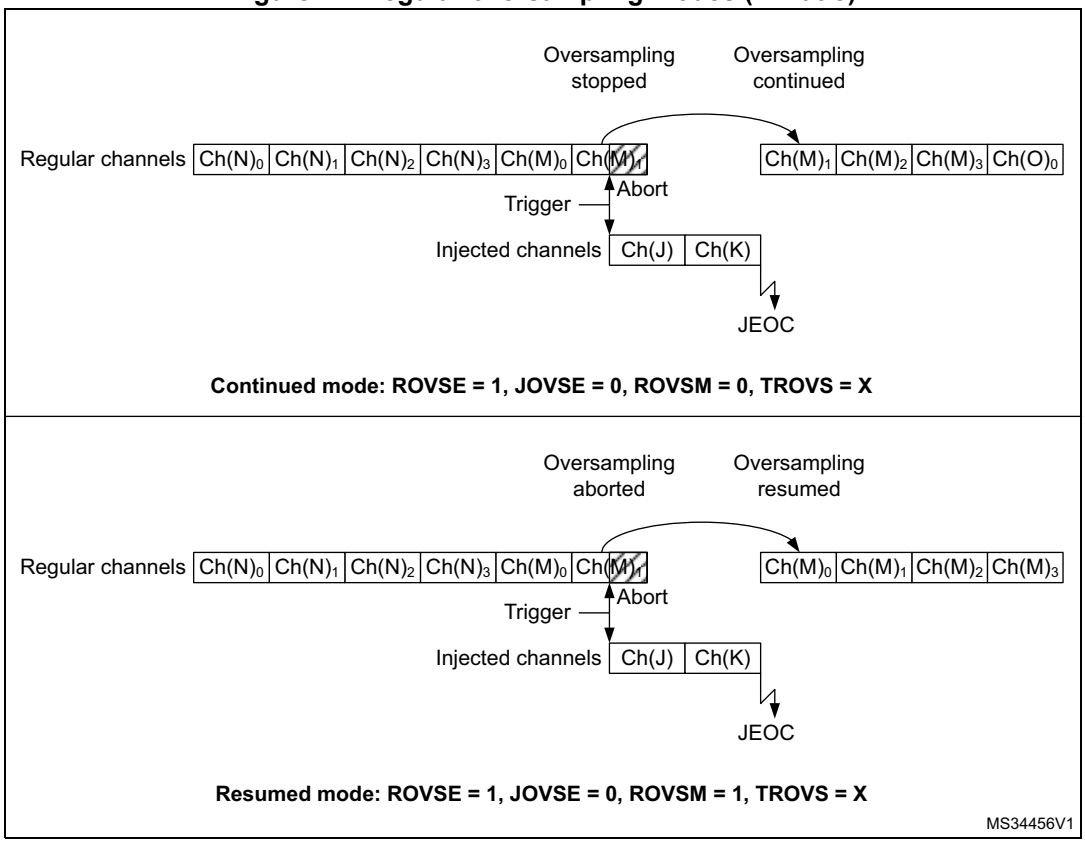
### Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);
- In resumed mode, the accumulation restarts from 0 (previous conversion results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be complete and the regular sequencer is blocked.

*Figure 77* gives examples for a 4x oversampling ratio.

**Figure 77. Regular oversampling modes (4x ratio)**



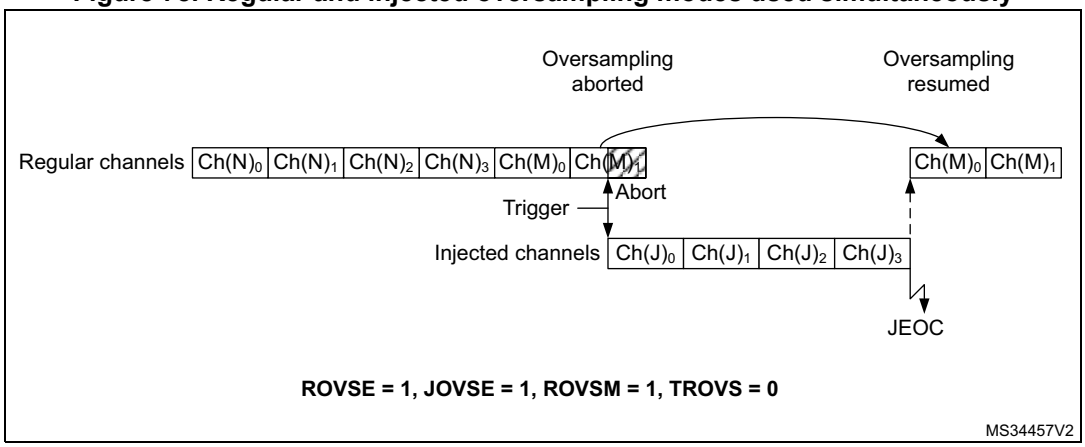

### Oversampling Injected channels only

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

### Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on *Figure 78* below.

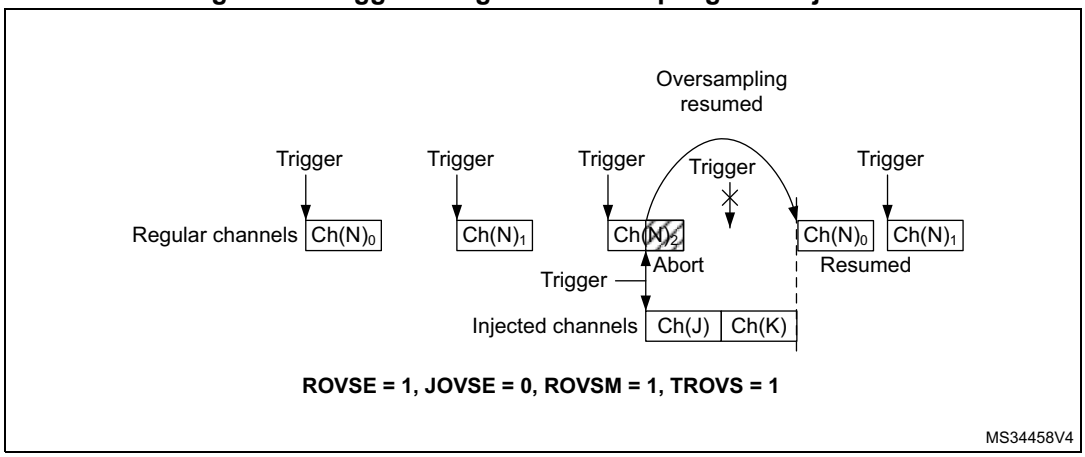
**Figure 78. Regular and injected oversampling modes used simultaneously**




### Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on *Figure 79* below.

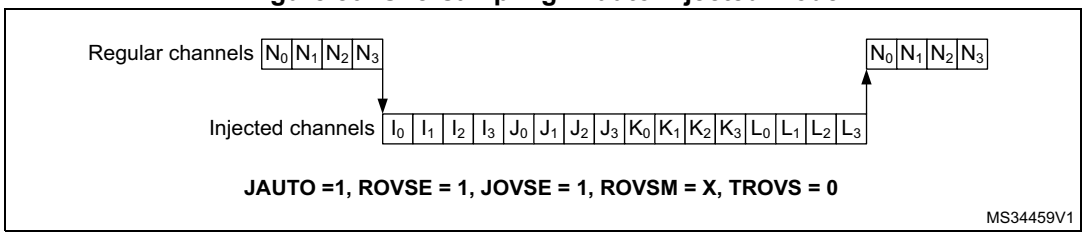
**Figure 79. Triggered regular oversampling with injection**



### Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The *Figure 80* below shows how the conversions are sequenced.

**Figure 80. Oversampling in auto-injected mode**




It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO = 1, DISCEN = 0, JDISCEN = 0, ROVSE = 1, JOVSE = 1 and TROVSE = 1.

## 15.3.30 Temperature sensor

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. When not in use, the sensor can be put in power down mode. It support the temperature range –40 to 125 °C.

*Figure 81* shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

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.

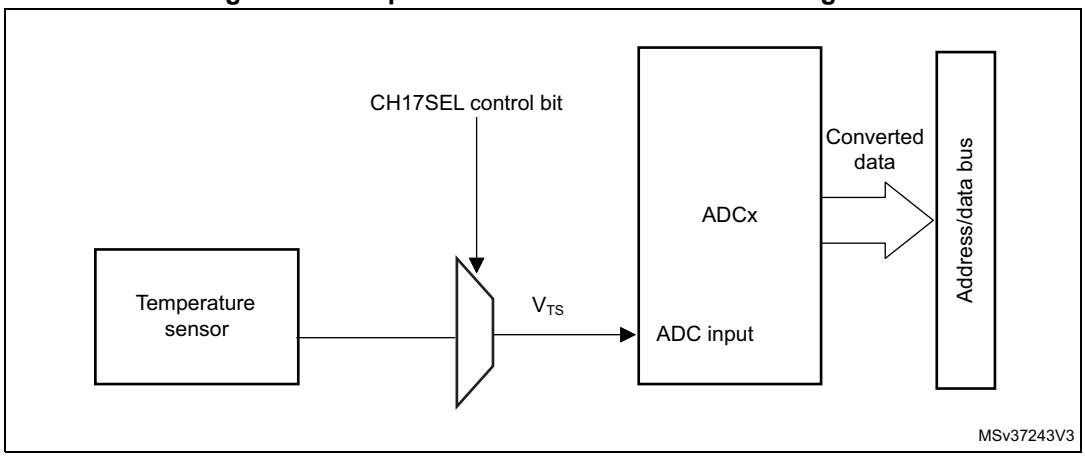
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

*Figure 81* shows the block diagram of the temperature sensor.

**Figure 81. Temperature sensor channel block diagram**




## Reading the temperature

To use the sensor:

1. Select the ADC input channel that is connected to $V_{TS}$.
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the CH17SEL bit in the ADCx_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting $V_{TS}$ data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{temperature (in °C)} = \frac{\text{TS_CAL2_TEMP} - \text{TS_CAL1_TEMP}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + \text{TS_CAL1_TEMP}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by ADC.

  Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

*Note: The sensor has a startup time after waking from power-down mode before it can output $V_{TS}$ at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and CH17SEL bits should be set at the same time.*

*The above formula is given for TS_DATA measurement done with the same $V_{DDA}$ voltage as TS_CAL1/TS_CAL2 values. If $V_{DDA}$ is different, the formula must be adapted. For example if $V_{DDA}$ = 3.3 V and TS_CAL data are acquired at $V_{DDA}$ = 3.0 V, TS_DATA must be replaced by TS_DATA x (3.3/3.0).*

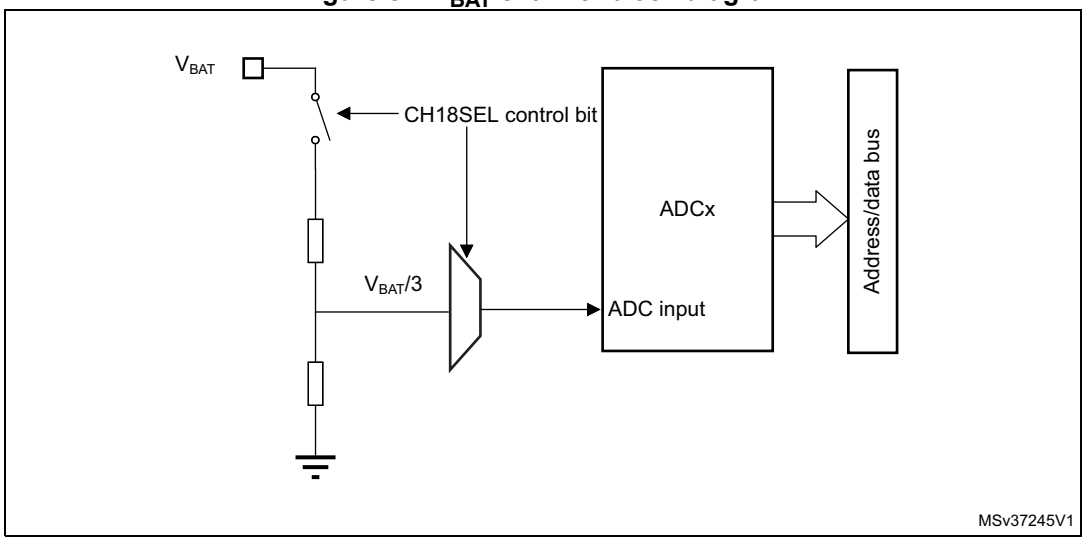
### 15.3.31 $V_{BAT}$ supply monitoring

The CH18SEL bit in the ADCx_CCR register is used to switch to the battery voltage. As the $V_{BAT}$ voltage could be higher than $V_{DDA}$, to ensure the correct operation of the ADC, the $V_{BAT}$ pin is internally connected to a bridge divider by 3. This bridge is automatically enabled when CH18SEL is set, to connect $V_{BAT}/3$ to the ADC input channels. As a consequence, the converted digital value is one third of the $V_{BAT}$ voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/3$ voltage.

The figure below shows the block diagram of the $V_{BAT}$ sensing feature.

**Figure 82. $V_{BAT}$ channel block diagram**




1. The CH18SEL bit must be set to enable the conversion of internal channel for $V_{BAT}/3$.

### 15.3.32 Monitoring the internal voltage reference

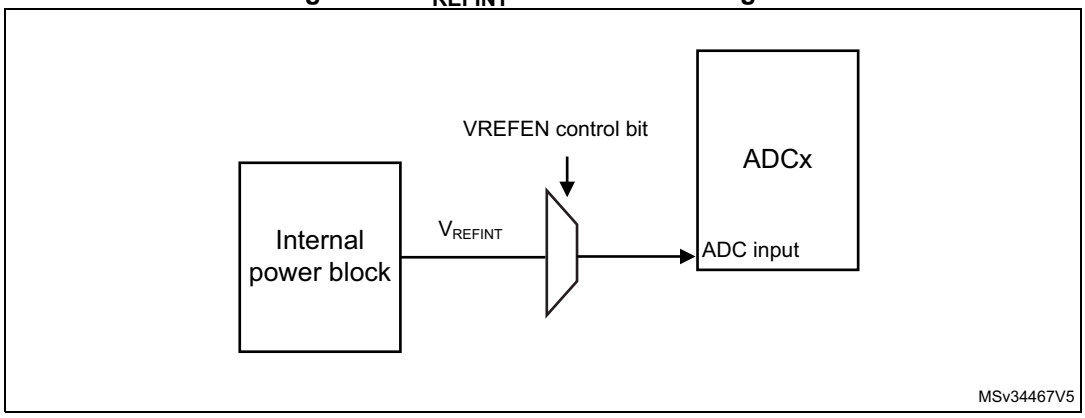
It is possible to monitor the internal voltage reference ($V_{REFINT}$) to have a reference point for evaluating the ADC $V_{DDA}$ voltage level.

The internal voltage reference is internally connected to the input channel 0 of the ADC1 (ADC1_INP0).

Refer to the electrical characteristics section of the product datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

*Figure 83* shows the block diagram of the $V_{REFINT}$ sensing feature.

**Figure 83. $V_{REFINT}$ channel block diagram**




1. The VREFEN bit into ADCx_CCR register must be set to enable the conversion of internal channels ($V_{REFINT}$).

### Calculating the actual $V_{DDA}$ voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. $V_{REFINT}$ and its calibration data, acquired by the ADC during the manufacturing process at $V_{DDA_Charac}$, can be used to evaluate the actual $V_{DDA}$ voltage level.

The following formula gives the actual $V_{DDA}$ voltage supplying the device:

$$V_{DDA} = V_{DDA_Charac} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- $V_{DDA_Charac}$ is the value of $V_{DDA}$ voltage characterized at $V_{REFINT}$ during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the $V_{REFINT}$ calibration value
- VREFINT_DATA is the actual $V_{REFINT}$ output value converted by ADC

### Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between $V_{DDA}$ and the voltage applied on the converted channel.

For most applications $V_{DDA}$ value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from $V_{DDA}$:

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL_SCALE} \times ADC_DATA$$

For applications where $V_{DDA}$ is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL_SCALE} \times ADC_DATA$$

Where:

- $V_{DDA_Charac}$ is the value of $V_{DDA}$ voltage characterized at $V_{REFINT}$ during the manufacturing process.
- VREFINT_CAL is the $V_{REFINT}$ calibration value
- ADC_DATA is the value measured by the ADC on channel x (right-aligned)
- VREFINT_DATA is the actual $V_{REFINT}$ output value converted by the ADC
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, it is $2^{12}$ - 1 = 4095 or with 8-bit resolution, $2^8$ - 1 = 255.

*Note: If ADC measurements are done using an output format other than 16-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.*

## 15.4 ADC in low-power mode

**Table 75. Effect of low-power modes on the ADC**

| Mode | Description |
|---|---|
| Sleep | No effect.<br>DMA requests are functional. |
| Low-power run | No effect. |
| Low-power sleep | No effect.<br>DMA requests are functional. |
| Stop 0/Stop 1 | The ADC is not operational. Its state is kept.<br>The ADC consumes the static current recommended to disable the peripheral in advance in order to reduce power consumption. |
| Stop 2 | The ADC is not operational. Its state is kept.<br>The ADC regulator is disabled by hardware.<br>ADC and regulator must be disabled before entering Stop 2 mode. |
| Standby | The ADC is powered down and must be reinitialized after exiting Standby or Shutdown mode. |
| Shutdown | |

## 15.5 ADC interrupts

An interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

**Table 76. ADC interrupts**

| Interrupt event | Event flag | Enable control bit |
|---|---|---|
| ADC ready | ADRDY | ADRDYIE |
| End of conversion of a regular group | EOC | EOCIE |
| End of sequence of conversions of a regular group | EOS | EOSIE |
| End of conversion of a injected group | JEOC | JEOCIE |
| End of sequence of conversions of an injected group | JEOS | JEOSIE |
| Analog watchdog 1 status bit is set | AWD1 | AWD1IE |
| Analog watchdog 2 status bit is set | AWD2 | AWD2IE |
| Analog watchdog 3 status bit is set | AWD3 | AWD3IE |
| End of sampling phase | EOSMP | EOSMPIE |
| Overrun | OVR | OVRIE |
| Injected context queue overflows | JQOVF | JQOVFIE |

## 15.6 ADC registers

Refer to *Section 1.2 on page 51* for a list of abbreviations used in register descriptions.

### 15.6.1 ADC interrupt and status register (ADC_ISR)

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. | JQOVF | AWD3 | AWD2 | AWD1 | JEOS | JEOC | OVR | EOS | EOC | EOSMP | ADRDY |
| | | | | | rc_w1 | rc_w1 | rc_w1 | 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 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to *Section 15.3.21: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software. writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

### 15.6.2 ADC interrupt enable register (ADC_IER)

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. | JQOVFIE | AWD3IE | AWD2IE | AWD1IE | JEOSIE | JEOCIE | OVRIE | EOSIE | EOCIE | EOSMPIE | ADRDYIE |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.
0: JEOC interrupt disabled.
1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.
0: Overrun interrupt disabled
1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.
0: EOS interrupt disabled
1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.
0: EOC interrupt disabled.
1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.
0: EOSMP interrupt disabled.
1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.
0: ADRDY interrupt disabled
1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 15.6.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x2000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADCAL | ADCALDIF | DEEPPWD | ADVREGEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| rs | 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. | Res. | JADSTP | ADSTP | JADSTART | ADSTART | ADDIS | ADEN |
| | | | | | | | | | | rs | rs | rs | rs | rs | rs |

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

*Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.*

*The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing)*

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

0: Writing ADCAL launches a calibration in single-ended inputs mode.

1: Writing ADCAL launches a calibration in differential inputs mode.

*Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

*Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bits is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled

1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 15.3.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when the ADC is disabled (ADCAL = *0*, JADSTART = *0*, ADSTART = *0*, ADSTP = *0*, ADDIS = *0* and ADEN = *0*).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

*Note: The software is allowed to set JADSTP only when JADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC)*

*In Auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)*

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

*Note: The software is allowed to set ADSTP only when ADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).*

*In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).*

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (JEXTSEL = *0*x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.

– in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

*Note: The software is allowed to set JADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).*

*In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)*

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN[1:0], a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (EXTSEL = *0*x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.

– in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

*Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC)*

*In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)*

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

*Note: The software is allowed to set ADDIS only when ADEN = 1 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

*Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)*

### 15.6.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C

Reset value: 0x8000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JQDIS | AWD1CH[4:0] | | | | | JAUTO | JAWD1EN | AWD1EN | AWD1SGL | JQM | JDISCEN | DISCNUM[2:0] | | | DISCEN |
| 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 |
| Res. | AUTDLY | CONT | OVRMOD | EXTEN[1:0] | | EXTSEL3 | EXTSEL2 | EXTSEL1 | EXTSEL0 | ALIGN | RES[1:0] | | Res. | DMACFG | DMAEN |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw |

Bit 31 **JQDIS**: Injected Queue disable

These bits are set and cleared by software to disable the Injected Queue mechanism :

0: Injected Queue enabled

1: Injected Queue disabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).*

*A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.*

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

10010: ADC analog input channel 18 monitored by AWD1

others: reserved, must not be used

*Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.*

*The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.*

*The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).*

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 21 **JQM**: JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: JSQR mode 0: The Queue is never empty and maintains the last written configuration into JSQR.

1: JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 15.3.21: Queue of context for injected conversions* for more information.

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

*It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.*

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

*Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.*

*It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.*

*The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off

1: Auto-delayed conversion mode on

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

*Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.*

*The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bits 9:6 **EXTSEL[3:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

0000: Event 0
0001: Event 1
0010: Event 2
0011: Event 3
0100: Event 4
0101: Event 5
0110: Event 6
0111: Event 7
...
1111: Event 15

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to *Section : Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)*

0: Right alignment
1: Left alignment

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit
01: 10-bit
10: 8-bit
11: 6-bit

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMACFG**: Direct memory access configuration

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = *1*.

0: DMA One Shot mode selected
1: DMA Circular mode selected

For more details, refer to *Section : Managing conversions using the DMA*

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA to manage automatically the converted data. For more details, refer to *Section : Managing conversions using the DMA*.

0: DMA disabled
1: DMA enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 15.6.5 ADC configuration register 2 (ADC_CFGR2)

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. | ROVSM | TROVS | OVSS[3:0] | | | | OVSR[2:0] | | | JOVSE | ROVSE |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 27:17 Reserved, must be kept at reset value.

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular Oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 9 **TROVS**: Triggered Regular Oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift
0001: Shift 1-bit
0010: Shift 2-bits
0011: Shift 3-bits
0100: Shift 4-bits
0101: Shift 5-bits
0110: Shift 6-bits
0111: Shift 7-bits
1000: Shift 8-bits
Other codes reserved

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 4:2 **OVSR[2:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

000: 2x
001: 4x
010: 8x
011: 16x
100: 32x
101: 64x
110: 128x
111: 256x

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 1 **JOVSE**: Injected Oversampling Enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected Oversampling disabled
1: Injected Oversampling enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*

Bit 0 **ROVSE**: Regular Oversampling Enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular Oversampling disabled
1: Regular Oversampling enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*

## 15.6.6 ADC sample time register 1 (ADC_SMPR1)

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. | SMP9[2:0] | | | SMP8[2:0] | | | SMP7[2:0] | | | SMP6[2:0] | | | SMP5[2:1] | |
| | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMP5[0] | SMP4[2:0] | | | SMP3[2:0] | | | SMP2[2:0] | | | SMP1[2:0] | | | SMP0[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 6.5 ADC clock cycles
010: 12.5 ADC clock cycles
011: 24.5 ADC clock cycles
100: 47.5 ADC clock cycles
101: 92.5 ADC clock cycles
110: 247.5 ADC clock cycles
111: 640.5 ADC clock cycles

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.*

## 15.6.7 ADC sample time register 2 (ADC_SMPR2)

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. | SMP18[2:0] | | | SMP17[2:0] | | | SMP16[2:0] | | | SMP15[2:1] | |
| | | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMP15[0] | SMP14[2:0] | | | SMP13[2:0] | | | SMP12[2:0] | | | SMP11[2:0] | | | SMP10[2:0] | | |
| rw | rw | 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 26:0 **SMPx[2:0]:** Channel x sampling time selection (x = 18 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 6.5 ADC clock cycles
010: 12.5 ADC clock cycles
011: 24.5 ADC clock cycles
100: 47.5 ADC clock cycles
101: 92.5 ADC clock cycles
110: 247.5 ADC clock cycles
111: 640.5 ADC clock cycles

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.*

### 15.6.8 ADC watchdog threshold register 1 (ADC_TR1)

Address offset: 0x20

Reset value: 0x0FFF 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | HT1[11:0] | | | | | | | | | | | |
| | | | | 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. | Res. | LT1[11:0] | | | | | | | | | | | |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to *Section 15.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

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

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to *Section 15.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 15.6.9 ADC watchdog threshold register 2 (ADC_TR2)

Address offset: 0x24

Reset value: 0x00FF 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. | HT2[7:0] | | | | | | | |
| | | | | | | | | 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. | Res. | Res. | Res. | Res. | LT2[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 23:16 **HT2[7:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to *Section 15.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT2[7:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to *Section 15.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

## 15.6.10 ADC watchdog threshold register 3 (ADC_TR3)

Address offset: 0x28

Reset value: 0x00FF 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. | HT3[7:0] | | | | | | | |
| | | | | | | | | 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. | Res. | Res. | Res. | Res. | LT3[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to *Section 15.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 15.6.11 ADC regular sequence register 1 (ADC_SQR1)

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. | SQ4[4:0] | | | | | Res. | SQ3[4:0] | | | | | Res. | SQ2[4] |
| | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SQ2[3:0] | | | | Res. | SQ1[4:0] | | | | | Res. | Res. | L[3:0] | | | |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | | | rw | rw | rw | rw |

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 1st in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 15.6.12 ADC regular sequence register 2 (ADC_SQR2)

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. | SQ9[4:0] | | | | | Res. | SQ8[4:0] | | | | | Res. | SQ7[4] |
| | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SQ7[3:0] | | | | Res. | SQ6[4:0] | | | | | Res. | SQ5[4:0] | | | | |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 9th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 8th in the regular conversion sequence

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 7th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 6th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 5th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 15.6.13 ADC regular sequence register 3 (ADC_SQR3)

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. | SQ14[4:0] | | | | | Res. | SQ13[4:0] | | | | | Res. | SQ12[4] |
| | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SQ12[3:0] | | | | Res. | SQ11[4:0] | | | | | Res. | SQ10[4:0] | | | | |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 14th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 13th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 12th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 11th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 10th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 15.6.14 ADC regular sequence register 4 (ADC_SQR4)

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. | SQ16[4:0] | | | | | Res. | SQ15[4:0] | | | | |
| | | | | | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |

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

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 16th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 15th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 15.6.15 ADC regular data register (ADC_DR)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDATA[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 **RDATA[15:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 15.3.26: Data management*.

### 15.6.16 ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | JSQ4[4:0] | | | | | Res. | JSQ3[4:0] | | | | | Res. | JSQ2[4:2] | | |
| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JSQ2[1:0] | | Res. | JSQ1[4:0] | | | | | JEXTEN[1:0] | | JEXTSEL[3:0] | | | | JL[1:0] | |
| rw | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the injected conversion sequence.

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the injected conversion sequence.

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 19 Reserved, must be kept at reset value.

Bits 18:14 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the injected conversion sequence.

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 13 Reserved, must be kept at reset value.

Bits 12:8 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 1st in the injected conversion sequence.

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bits 7:6 **JEXTEN[1:0]**: External Trigger Enable and Polarity Selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS = 0 (queue enabled), Hardware and software trigger detection disabled

00: If JQDIS = 1 (queue disabled), Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

*If JQM = 1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 15.3.21: Queue of context for injected conversions)*

Bits 5:2 **JEXTSEL[3:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

0000: Event 0

0001: Event 1

0010: Event 2

0011: Event 3

0100: Event 4

0101: Event 5

0110: Event 6

0111: Event 7

...

1111: Event 15

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bits 1:0 **JL[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 15.6.17 ADC offset y register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y -1), (y= 1 to 4)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OFFSET_EN | OFFSET_CH[4:0] | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| 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. | OFFSET[11:0] | | | | | | | | | | | |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **OFFSET_EN:** Offset y enable

This bit is written by software to enable or disable the offset programmed into bits OFFSETy[11:0].

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 30:26 **OFFSET_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[11:0] applies.

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically and must not be selected for the data offset y.*

Bits 25:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET[11:0]:** Data offset y for the channel programmed into bits OFFSETy_CH[4:0]

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.*
*Ex: if OFFSET1_CH[4:0]=4 and OFFSET2_CH[4:0]=4, this is OFFSET1[11:0] which is subtracted when converting channel 4.*

### 15.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y - 1), (y = 1 to 4)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JDATA[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 **JDATA[15:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in *Section 15.3.26: Data management*.

### 15.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0xA0

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. | AWD2CH[18:16] | | |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AWD2CH[15:0] | | | | | | | | | | | | | | | |
| 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.

Bits 18:0 **AWD2CH[18:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWD2CH[18:0] = 000..0, the analog watchdog 2 is disabled

*Note: The channels selected by AWD2CH must be also selected into the SQRi or JSQRi registers.*

*The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically and must not be selected for the analog watchdog.*

### 15.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0xA4

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. | AWD3CH[18:16] | | |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AWD3CH[15:0] | | | | | | | | | | | | | | | |
| 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.

Bits 18:0 **AWD3CH[18:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWD3CH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWD3CH[18:0] = 000..0, the analog watchdog 3 is disabled

*Note: The channels selected by AWD3CH must be also selected into the SQRi or JSQRi registers.*

*The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically and must not be selected for the analog watchdog.*

### 15.6.21 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0xB0

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. | DIFSEL[18:16] | | |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DIFSEL[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | r |

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

Bits 18:0 **DIFSEL[18:0]**: Differential mode for channels 18 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or differential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in differential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).

*The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

### 15.6.22 ADC calibration factors (ADC_CALFACT)

Address offset: 0xB4

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. | CALFACT_D[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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CALFACT_S[6:0] | | | | | | |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **CALFACT_D[6:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new differential calibration is launched.

*Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).*

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT_S[6:0]**: Calibration Factors In single-ended mode

These bits are written by hardware or by software.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new single-ended calibration is launched.

*Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).*

## 15.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

### 15.7.1 ADC common status register (ADC_CSR)

Address offset: 0x300

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR 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. | JQOVF_MST | AWD3_MST | AWD2_MST | AWD1_MST | JEOS_MST | JEOC_MST | OVR_MST | EOS_MST | EOC_MST | EOSMP_MST | ADRDY_MST |
| | | | | | r | r | r | r | r | r | r | r | r | r | r |

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

Bit 10 **JQOVF_MST:** Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

### 15.7.2 ADC common control register (ADC_CCR)

Address offset: 0x308

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. | CH18SEL | CH17SEL | VREFEN | PRESC[3:0] | | | | CKMODE[1:0] | |
| | | | | | | | 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |

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

Bit 24 **CH18SEL**: CH18 selection

This bit is set and cleared by software to control channel 18.

0: $V_{BAT}$ channel disabled.

1: $V_{BAT}$ channel enabled

Bit 23 **CH17SEL**: CH17 selection

This bit is set and cleared by software to control channel 17.

0: Temperature sensor channel disabled

1: Temperature sensor channel enabled

Bit 22 **VREFEN**: $V_{REFINT}$ enable

This bit is set and cleared by software to enable/disable the $V_{REFINT}$ channel.

0: $V_{REFINT}$ channel disabled

1: $V_{REFINT}$ channel enabled

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
other: reserved

*Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0). The ADC prescaler value is applied only when CKMODE[1:0] = 00.*

Bits 17:16 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: CK_ADCx (x = 123) (Asynchronous clock mode), generated at product level (refer to *Section 6: Reset and clock control (RCC)*)

01: HCLK/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set (HPRE[3:0] = 0xxx in RCC_CFGR register) and if the system clock has a 50% duty cycle.

10: HCLK/2 (Synchronous clock mode)

11: HCLK/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

*Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bits 15:0 Reserved, must be kept at reset value.

## 15.8 ADC register map

**Table 77. ADC 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | ADC_ISR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JQOVF | AWD3 | AWD2 | AWD1 | JEOS | JEOC | OVR | EOS | EOC | EOSMP | ADRDY |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x04 | ADC_IER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JQOVFIE | AWD3IE | AWD2IE | AWD1IE | JEOSIE | JEOCIE | OVRIE | EOSIE | EOCIE | EOSMPIE | ADRDYIE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x08 | ADC_CR | ADCAL | ADCALDIF | DEEPPWD | ADVREGEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JADSTP | ADSTP | JADSTART | ADSTART | ADDIS | ADEN |
| | Reset value | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0C | ADC_CFGR | JQDIS | AWD1CH[4:0] | | | | | JAUTO | JAWD1EN | AWD1EN | AWD1SGL | JQM | JDISCEN | DISCNUM [2:0] | | | DISCEN | Res. | AUTDLY | CONT | OVRMOD | EXTEN[1:0] | | EXTSEL3 | EXTSEL2 | EXTSEL1 | EXTSEL0 | ALIGN | RES [1:0] | | Res. | DMACFG | DMAEN |
| | Reset value | 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 | 0 | 0 | | 0 | 0 |
| 0x10 | ADC_CFGR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ROVSM | TROVS | OVSS[3:0] | | | | OVSR [2:0] | | | JOVSE | ROVSE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x14 | ADC_SMPR1 | Res. | Res. | SMP9 [2:0] | | | SMP8 [2:0] | | | SMP7 [2:0] | | | SMP6 [2:0] | | | SMP5 [2:0] | | | SMP4 [2:0] | | | SMP3 [2:0] | | | SMP2 [2:0] | | | SMP1 [2:0] | | | SMP0 [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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x18 | ADC_SMPR2 | Res. | Res. | Res. | Res. | Res. | SMP18 [2:0] | | | SMP17 [2:0] | | | SMP16 [2:0] | | | SMP15 [2:0] | | | SMP14 [2:0] | | | SMP13 [2:0] | | | SMP12 [2:0] | | | SMP11 [2:0] | | | SMP10 [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 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1C | Reserved | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x20 | ADC_TR1 | Res. | Res. | Res. | Res. | HT1[11:0] | | | | | | | | | | | | Res. | Res. | Res. | Res. | LT1[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 |
| 0x24 | ADC_TR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | HT2[[7:0] | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LT2[7:0] | | | | | | | |
| | Reset value | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x28 | ADC_TR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | HT3[[7:0] | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LT3[7:0] | | | | | | | |
| | Reset value | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x2C | Reserved | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x30 | ADC_SQR1 | Res. | Res. | Res. | SQ4[4:0] | | | | | Res. | SQ3[4:0] | | | | | Res. | SQ2[4:0] | | | | | Res. | SQ1[4:0] | | | | | Res. | Res. | L[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 |
| 0x34 | ADC_SQR2 | Res. | Res. | Res. | SQ9[4:0] | | | | | Res. | SQ8[4:0] | | | | | Res. | SQ7[4:0] | | | | | Res. | SQ6[4:0] | | | | | Res. | SQ5[4:0] | | | | |
| | 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 |
| 0x38 | ADC_SQR3 | Res. | Res. | Res. | SQ14[4:0] | | | | | Res. | SQ13[4:0] | | | | | Res. | SQ12[4:0] | | | | | Res. | SQ11[4:0] | | | | | Res. | SQ10[4:0] | | | | |
| | 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 |
| 0x3C | ADC_SQR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SQ16[4:0] | | | | | Res. | SQ15[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 |
| 0x40 | ADC_DR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | regular RDATA[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x44-0x48 | Reserved | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 77. ADC 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x4C | **ADC_JSQR** | Res. | JSQ4[4:0] | | | | | Res. | JSQ3[4:0] | | | | | Res. | JSQ2[4:0] | | | | | Res. | JSQ1[4:0] | | | | | JEXTEN[1:0] | | JEXTSEL [3:0] | | | | JL[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 |
| 0x50-0x5C | Reserved | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x60 | **ADC_OFR1** | OFFSET1_EN | OFFSET1_CH[4:0] | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OFFSET1[11:0] | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x64 | **ADC_OFR2** | OFFSET2_EN | OFFSET2_CH[4:0] | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OFFSET2[11:0] | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x68 | **ADC_OFR3** | OFFSET3_EN | OFFSET3_CH[4:0] | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OFFSET3[11:0] | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x6C | **ADC_OFR4** | OFFSET4_EN | OFFSET4_CH[4:0] | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OFFSET4[11:0] | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x70-0x7C | Reserved | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x80 | **ADC_JDR1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JDATA1[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x84 | **ADC_JDR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JDATA2[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x88 | **ADC_JDR3** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JDATA3[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x8C | **ADC_JDR4** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JDATA4[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x8C-0x9C | Reserved | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA0 | **ADC_AWD2CR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | AWD2CH[18:0] | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xA4 | **ADC_AWD3CR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | AWD3CH[18:0] | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xA8-0xAC | 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. |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xB0 | **ADC_DIFSEL** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DIFSEL[18:0] | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 77. ADC 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xB4 | **ADC_CALFACT** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CALFACT_D[6:0] | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CALFACT_S[6:0] | | | | | | |
| | Reset value | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 78. ADC register map and reset values (master and slave ADC common registers) offset = 0x300**

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | **ADC_CSR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JQOVF_MST | AWD3_MST | AWD2_MST | AWD1_MST | JEOS_MST | JEOC_MST | OVR_MST | EOS_MST | EOC_MST | EOSMP_MST | ADRDY_MST |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x04 | Reserved | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x08 | **ADC_CCR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH18SEL | CH17SEL | VREFEN | PRESC[3:0] | | | | CKMODE[1:0] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 16 True random number generator (RNG)

## 16.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG can be used to construct a NIST compliant deterministic random bit generator (DRBG), acting as a live entropy source.

The RNG true random number generator has been tested using the German BSI statistical tests of AIS-31 (T0 to T8).

## 16.2 RNG main features

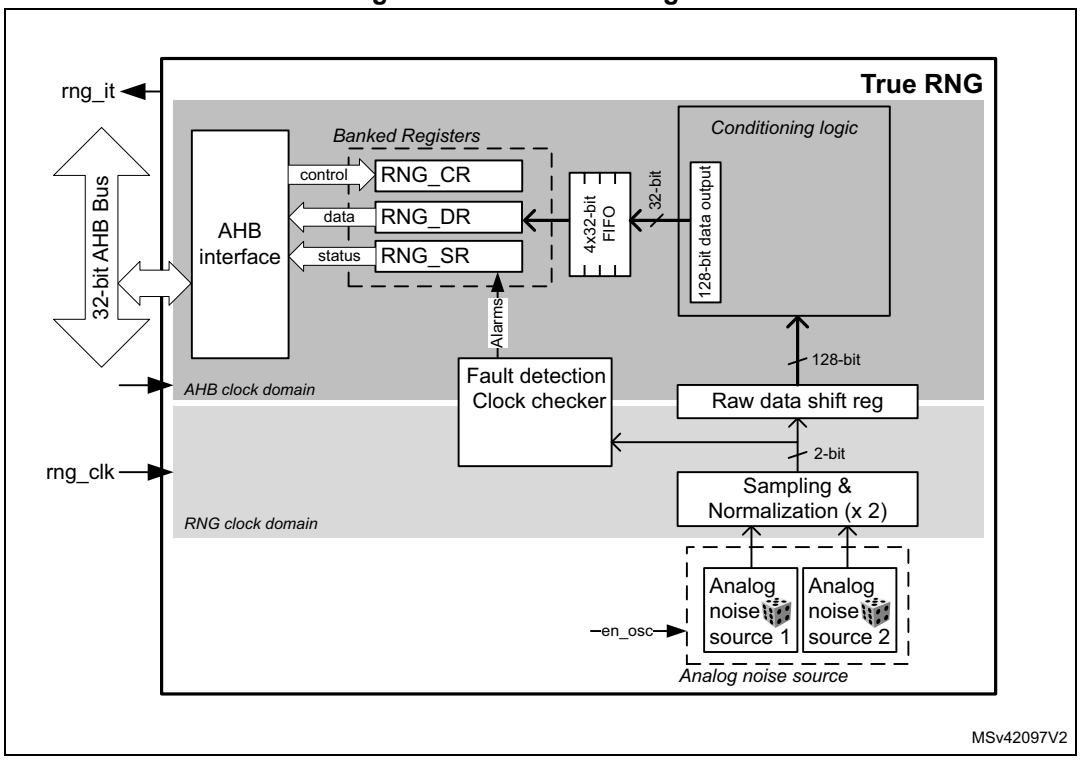
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source processed by a high quality conditioning stage.
- In the NIST configuration, it produces four 32-bit random samples every 16x $\frac{f_{AHB}}{f_{RNG}}$ AHB clock cycles, if value is higher than 213 cycles (213 cycles otherwise).
- It allows embedded continuous basic health tests with associated error management
  - Includes too low sampling clock detection and repetition count tests.
- It can be disabled to reduce power consumption.
- It has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

## 16.3 RNG functional description

### 16.3.1 RNG block diagram

*Figure 84* shows the RNG block diagram.

**Figure 84. RNG block diagram**




### 16.3.2 RNG internal signals

*Table 79* describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

**Table 79. RNG internal input/output signals**

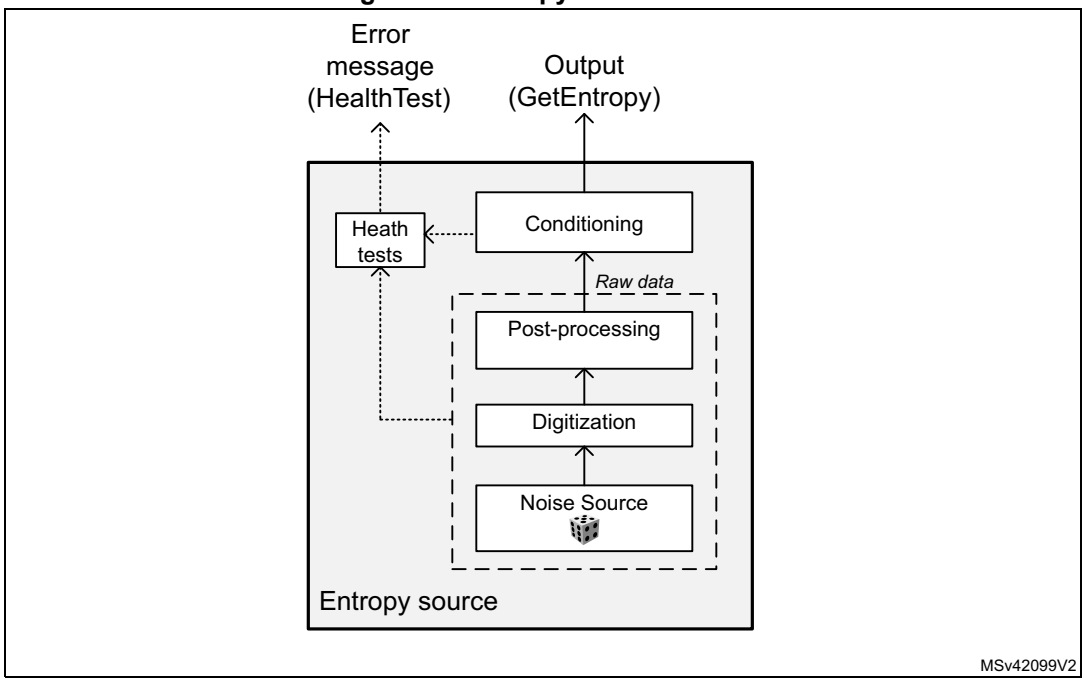
| Signal name | Signal type | Description |
|---|---|---|
| rng_it | Digital output | RNG global interrupt request |
| rng_hclk | Digital input | AHB clock |
| rng_clk | Digital input | RNG dedicated clock, asynchronous to rng_hclk |

### 16.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals. Within its boundary the RNG implements the entropy source model pictured on *Figure 85*.

It includes an analog noise source, a digitization stage with post-processing, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

**Figure 85. Entropy source model**




The components pictured above are detailed hereafter.

#### Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. It is composed of:

- Two analog noise sources, each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 16.3.8: RNG low-power use*.
- A sampling stage of these outputs clocked by a dedicated clock input (rng_clk), delivering a 2-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk).

*Note: In Section 16.6: RNG entropy source validation the recommended RNG clock frequencies are given.*

### Post processing

The sample values obtained from a true random noise source consist of 2-bit bitstrings. Because this noise source output is biased, the RNG implements a post-processing component that reduces that bias to a tolerable level.

More specifically, for each of the two noise source bits the RNG takes half of the bits from the sampled noise source, and half of the bits from the inverted sampled noise source. Thus, if the source generates more ‘1’ than ‘0’ (or the opposite), it is filtered.

### Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit).

Also note that post-processing computations are triggered when at least 32 bits of raw datum is received and when output FIFO needs a refill. Thus, the RNG output entropy is maximum when the RNG 128-bit FIFO is emptied by application after 64 RNG clock cycles.

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in *Section 16.5: RNG processing time*.

The conditioning component is clocked by the faster AHB clock.

### Output buffer

A data output buffer can store up to four 32-bit words, which have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register 213 AHB clock cycles later.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

*Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.*

### Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features.

1. Continuous health tests, running indefinitely on the output of the noise source
   - Repetition count test, flagging an error when:
   - a) One of the noise source has provided more than 64 consecutive bits at a constant value (“0” or “1”), or more than 32 consecutive occurrence of two bits patterns (“01” or “10”)
   - b) Both noise sources have delivered more than 32 consecutive bits at a constant value (“0” or “1”), or more than 16 consecutive occurrence of two bits patterns (“01” or “10”)
2. Vendor specific continuous test
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle is smaller than AHB clock cycle divided by 32.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 16.3.7: Error management*.

*Note: An interrupt can be generated when an error is detected.*

## 16.3.4 RNG initialization

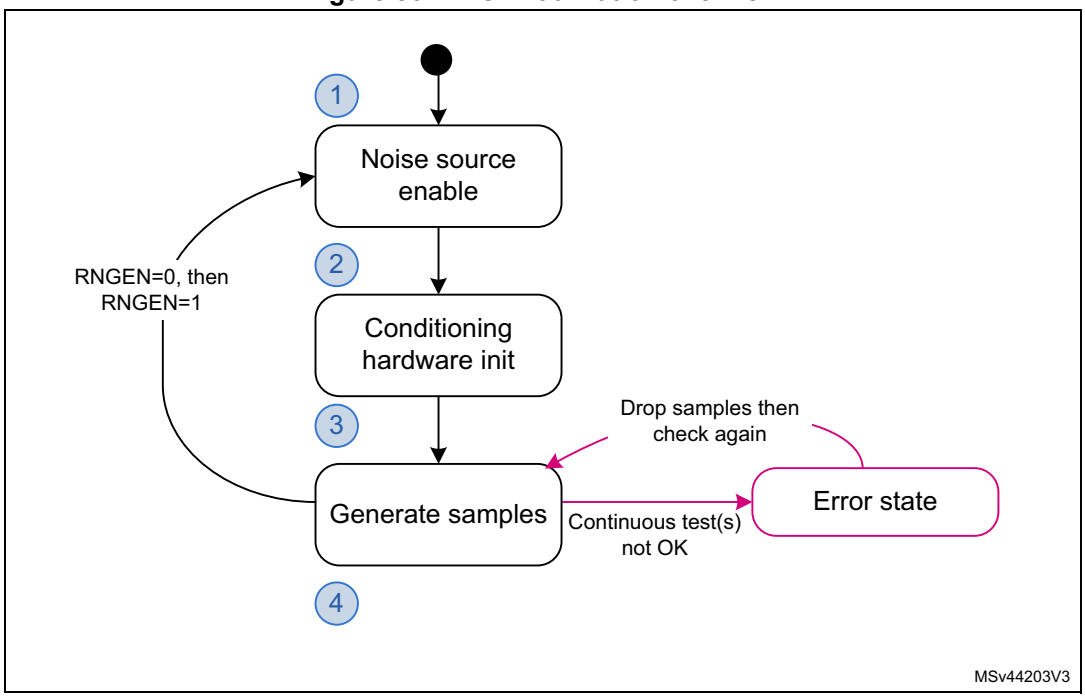
The RNG simplified state machine is pictured on *Figure 86*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and logic immediately starts sampling the analog output, filling the 128-bit conditioning shift register.
2. The conditioning logic is enabled and the post-processing context is initialized using two 128 noise source bits.
3. The conditioning stage internal input data buffer is filled again with 128-bit and one conditioning round is performed. The output buffer is then filled with the post processing result.
4. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 16.5: RNG processing time*.

**Figure 86. RNG initialization overview**




## 16.3.5 RNG operation

### Normal operations

To run the RNG using interrupts, the following steps are recommended:

1. Enable the interrupts by setting the IE bit in the RNG_CR register. At the same time, enable the RNG by setting the bit RNGEN=1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 16.3.7* must be used.

To run the RNG in polling mode following steps are recommended:

1. Enable the random number generation by setting the RNGEN bit to “1” in the RNG_CR register.
2. Read the RNG_SR register and check that:
   – No error occurred (the SEIS and CEIS bits must be set to 0)
   – A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 16.3.7* must be used.

*Note: When data is not ready (DRDY = 0) RNG_DR returns zero.*
*It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).*

#### Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 16.3.8: RNG low-power use*.

#### Software post-processing

If a NIST approved DRBG with 128 bits of security strength is required an approved random generator software must be built around the RNG true random number generator.

Built-in health check functions are described in *Section 16.3.3: Random number generation*.

### 16.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock is used for noise source sampling. Recommended clock configurations are detailed in *Section 16.6: RNG entropy source validation*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency the **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 16.3.1: RNG block diagram* for details (AHB and RNG clock domains).

### 16.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

#### Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see

*Section 16.3.6: RNG clocking*) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

*Note: The clock error has no impact on generated random numbers that is the application can still read the RNG_DR register.*

*CEIS is set only when CECS is set to 1 by RNG.*

#### Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy. If the error was detected during the initialization phase the whole initialization sequence is automatically restarted by the RNG.

The following sequence must be used to fully recover from a seed error after the RNG initialization:

1. Clear the SEIS bit by writing it to “0”.
2. Read out 12 words from the RNG_DR register, and discard each of them in order to clean the pipeline.
3. Confirm that SEIS is still cleared. Random number generation is back to normal.

### 16.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to the software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case the RNG must be re-enabled by the application until at least 32 new bits are collected from the noise source and a complete conditioning round is done. It corresponds to 16 RNG clock cycles to sample new bits, and 216 AHB clock cycles to run a conditioning round.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1.

## 16.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 16.3.7: Error management*
- Clock error, see *Section 16.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 80*.

**Table 80. RNG interrupt requests**

| Interrupt acronym | Interrupt event | Event flag | Enable control bit | Interrupt clear method |
|---|---|---|---|---|
| RNG | Data ready flag | DRDY | IE | None (automatic) |
| | Seed error flag | SEIS | IE | Write 0 to SEIS |
| | Clock error flag | CEIS | IE | Write 0 to CEIS |

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

*Note: Interrupts are generated only when RNG is enabled.*

## 16.5 RNG processing time

The conditioning stage can produce four 32-bit random numbers every 16x $\frac{f_{AHB}}{f_{RNG}}$ clock cycles, if the value is higher than 213 cycles (213 cycles otherwise).
More time is needed for the first set of random numbers after the device exits reset (see *Section 16.3.4: RNG initialization*). Indeed, after enabling the RNG for the first time, random data is first available after either:

- 128 RNG clock cycles + 426 AHB cycles, if $f_{AHB} < f_{threshold}$
- 192 RNG clock cycles + 213 AHB cycles, if $f_{AHB} \geq f_{threshold}$

With $f_{threshold} = (213 \times f_{RNG})/ 64$

## 16.6 RNG entropy source validation

### 16.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using the German BSI AIS-31 statistical tests (T0 to T8).

### 16.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk= 48 MHz (CED bit = ‘0’ in RNG_CR register) and rng_clk = 400 kHz (CED bit = ‘1’ in RNG_CR register).

## 16.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

### 16.7.1 RNG control register (RNG_CR)

Address offset: 0x000

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. | CED | Res. | IE | RNGEN | Res. | Res. |
| | | | | | | | | | | rw | | rw | rw | | |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection enabled

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.

1: True random number generator is enabled.

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

### 16.7.2 RNG status register (RNG_SR)

Address offset: 0x004

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. | SEIS | CEIS | Res. | Res. | SECS | CECS | DRDY |
| | | | | | | | | | rc_w0 | rc_w0 | | | r | r | r |

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

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0. Writing 1 has no effect.
0: No faulty sequence detected
1: At least one faulty sequence is detected. See SECS bit description for details.
An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.
0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$)
1: The RNG is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)
An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.
1: At least one of the following faulty sequences has been detected:

- One of the noise sources has provided more than 64 consecutive bits at a constant value (“0” or “1”), or more than 32 consecutive occurrence of two bit patterns (“01” or “10”)
- Both noise sources have delivered more than 32 consecutive bits at a constant value (“0” or “1”), or more than 16 consecutive occurrence of two bit patterns (“01” or “10”)

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.
1: The RNG clock is too slow ($f_{RNGCLK} < f_{HCLK}/32$).

*Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.*

Bit 0 **DRDY:** Data ready

0: The RNG_DR register is not yet valid, no random data is available.
1: The RNG_DR register contains valid random data.
Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

*Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG_CR register).*

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY = 1.

### 16.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. After being read, this register delivers a new random value after 216 periods of AHB clock if the output FIFO is empty.

The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RNDATA[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 |
| RNDATA[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data, which are valid when DRDY = 1. When DRDY = 0, the RNDATA value is zero.

When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. The zero value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

## 16.7.4 RNG register map

**Table 81. RNG register map and reset map**

| Offset | Register name | 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 | RNG_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. | CED | Res. | IE | RNGEN | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | 0 | 0 | | |
| 0x004 | RNG_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. | SEIS | CEIS | Res. | Res. | SECS | CECS | DRDY |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | | | 0 | 0 | 0 |
| 0x008 | RNG_DR | RNDATA[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* for the register boundary addresses.

# 17 AES hardware accelerator (AES)

## 17.1 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).

## 17.2 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

## 17.3 AES implementation

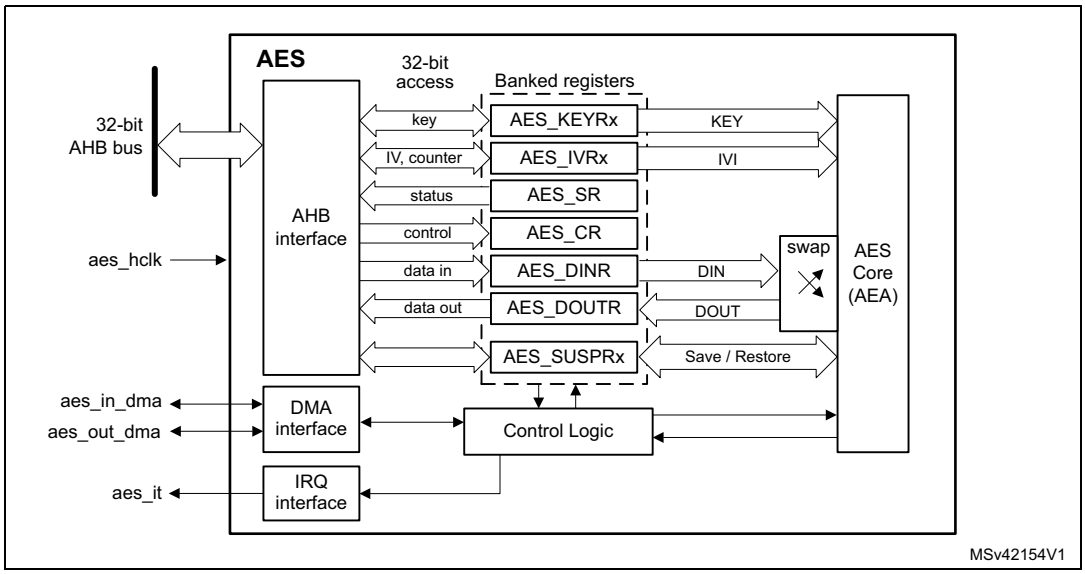
The devices have one AES peripheral, AES2.

## 17.4 AES functional description

### 17.4.1 AES block diagram

*Figure 87* shows the block diagram of AES.

**Figure 87. AES block diagram**




### 17.4.2 AES internal signals

*Table 82* describes the user relevant internal signals interfacing the AES peripheral.

**Table 82. 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 |

### 17.4.3 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 17.4.4: AES procedure to perform a cipher operation on page 464*.

*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)

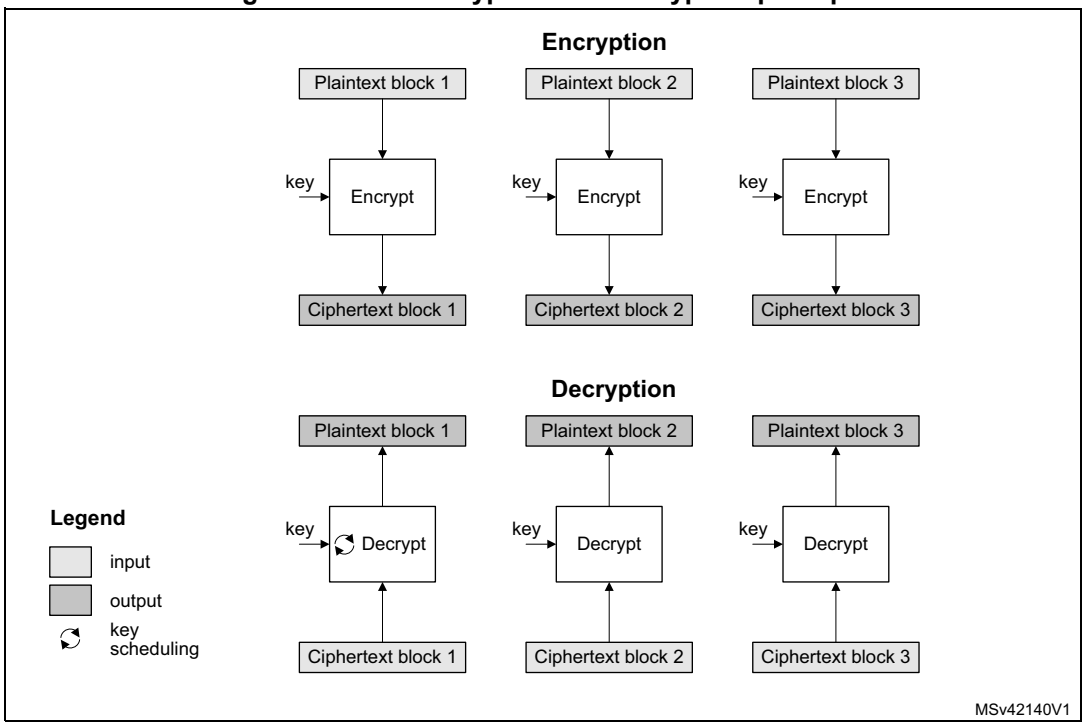
*Note: The chaining mode may be changed only when AES is disabled (bit EN of the AES_CR register cleared).*

Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from *Section 17.4.8: AES basic chaining modes (ECB, CBC)*.

### Electronic codebook (ECB) mode

**Figure 88. ECB encryption and decryption principle**

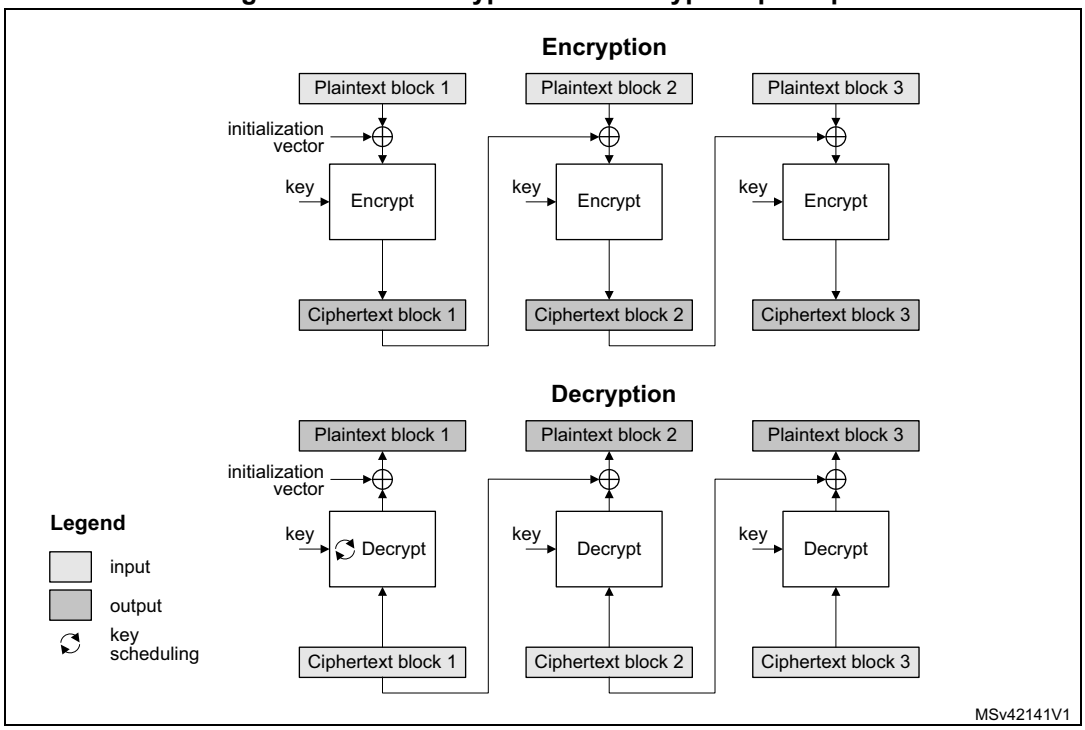



ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

*Note: For decryption, a special key scheduling is required before processing the first block.*

### Cipher block chaining (CBC) mode

**Figure 89. CBC encryption and decryption principle**

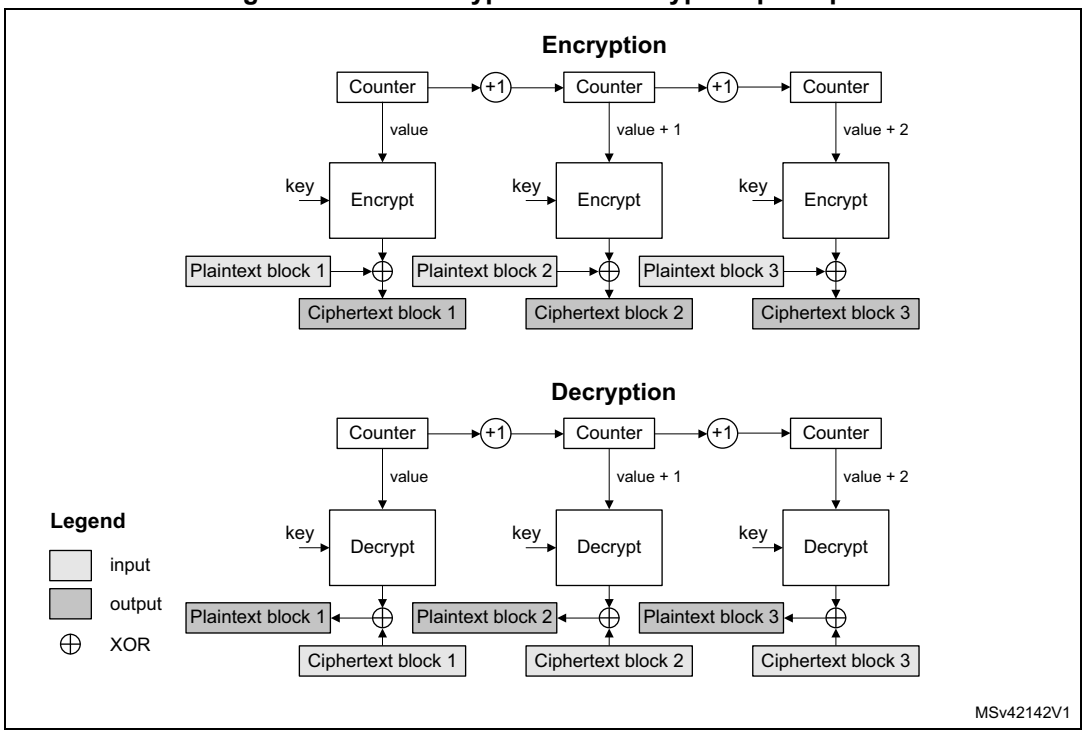



In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

*Note: For decryption, a special key scheduling is required before processing the first block.*

## Counter (CTR) mode

**Figure 90. CTR encryption and decryption principle**

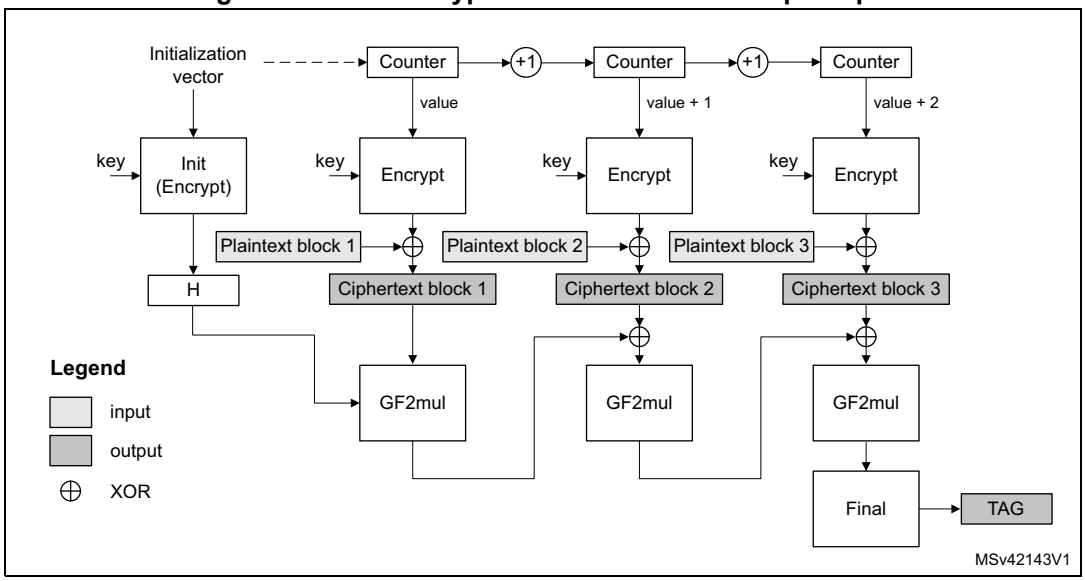



The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

*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 91. GCM encryption and authentication principle**

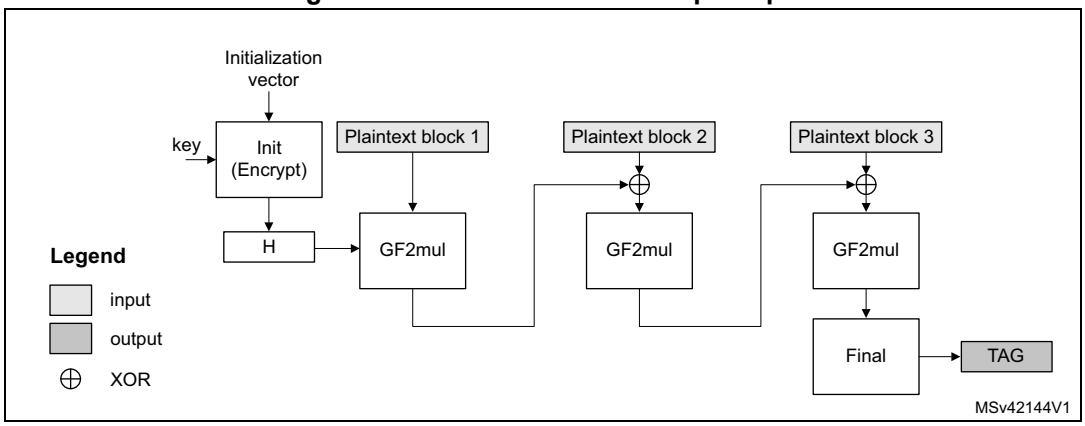



In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block 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 92. GMAC authentication principle**

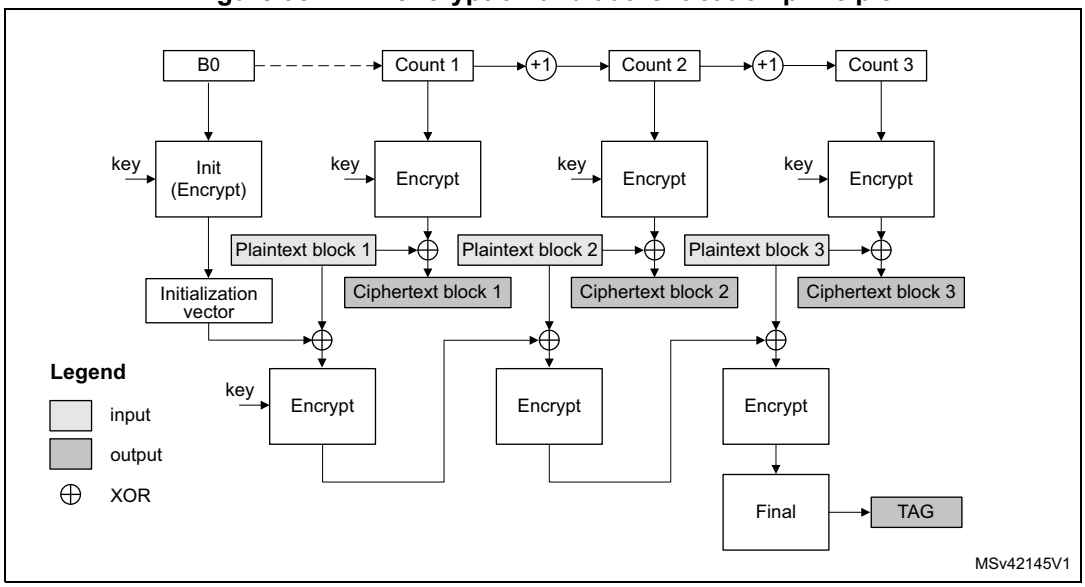



Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

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 93. 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.

## 17.4.4 AES procedure to perform a cipher operation

### Introduction

A typical cipher operation is explained below. Detailed information is provided in sections starting from *Section 17.4.8: 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 17.4.5: 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 17.4.6: 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 17.4.16: 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.*

### 17.4.5 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.

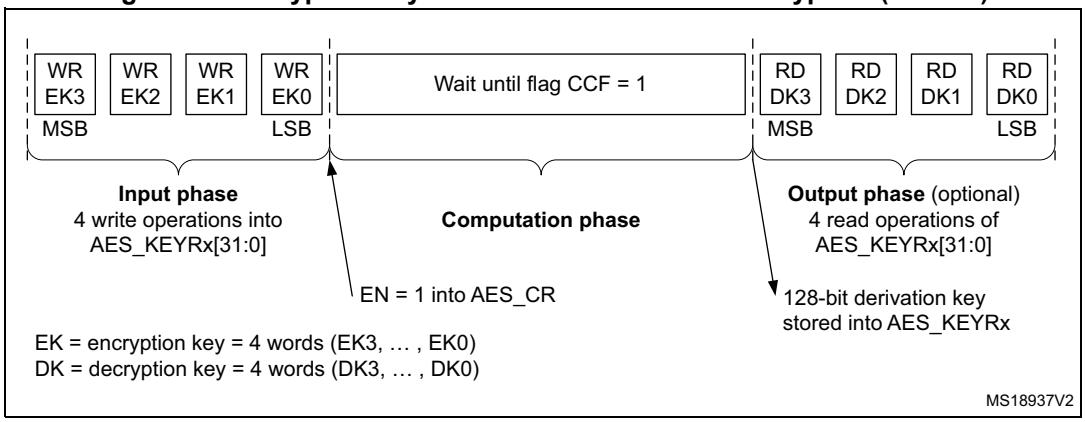
Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES_CR (key process only), then proceed with the decryption by setting MODE[1:0] to 10 (Mode 3, decryption only). Mode 2 usage is described below:

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 94*. 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 94* (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 94. 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).*

### 17.4.6 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 17.4.4: AES procedure to perform a cipher operation on page 464* 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 17.4.13: AES data registers and data swapping for details).*

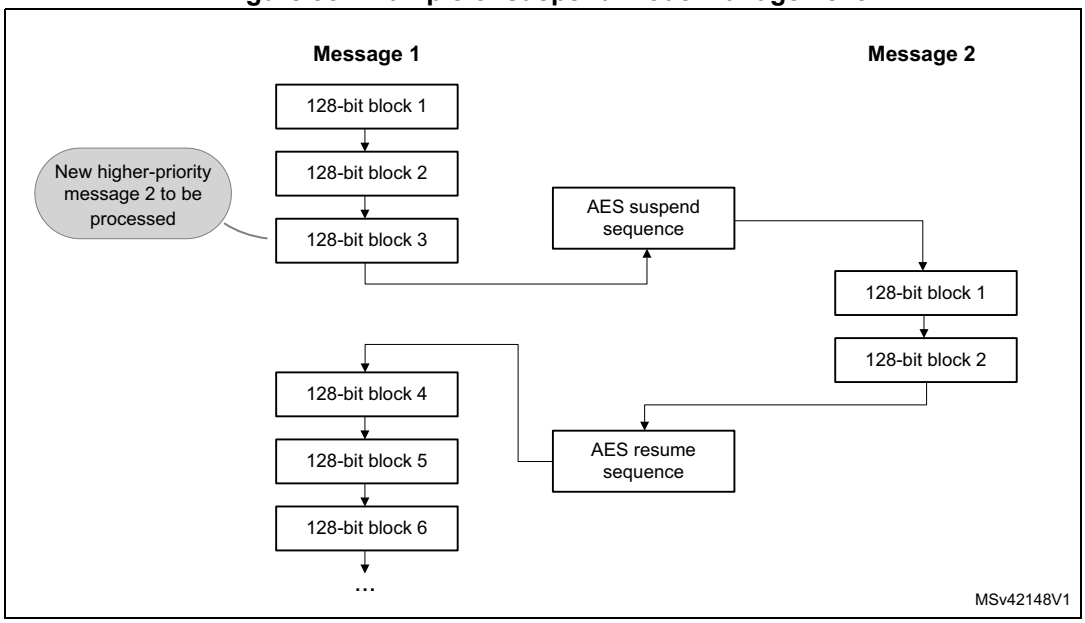
### 17.4.7 AES task suspend and resume

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 95* gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

**Figure 95. Example of suspend mode management**



A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

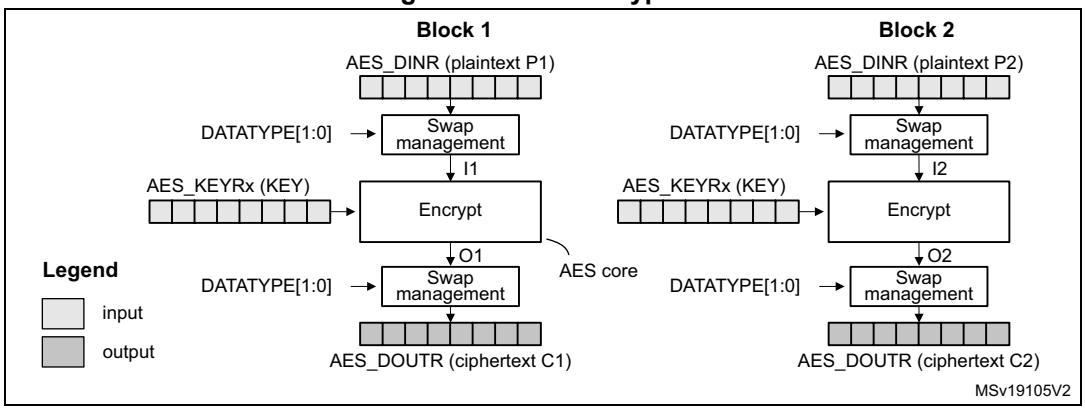
## 17.4.8 AES basic chaining modes (ECB, CBC)

### 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 96* illustrates the electronic codebook (ECB) encryption.

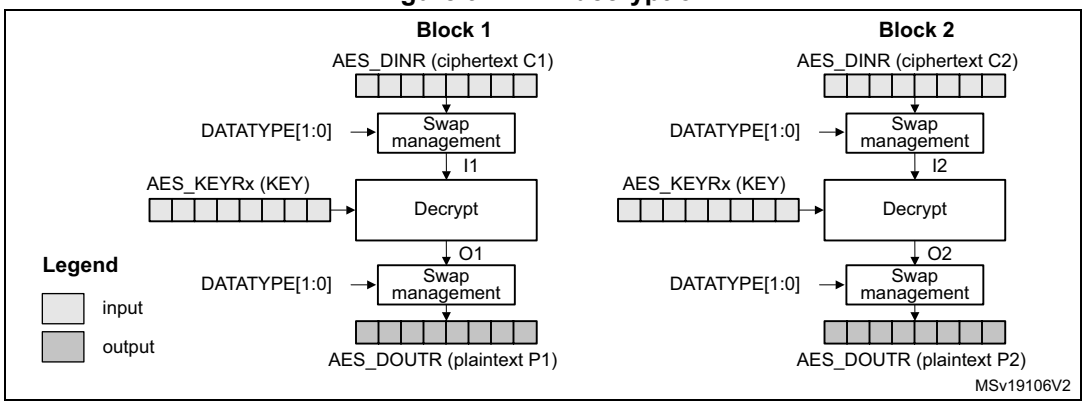
**Figure 96. ECB encryption**




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 97* illustrates the electronic codebook (ECB) decryption.

**Figure 97. ECB decryption**

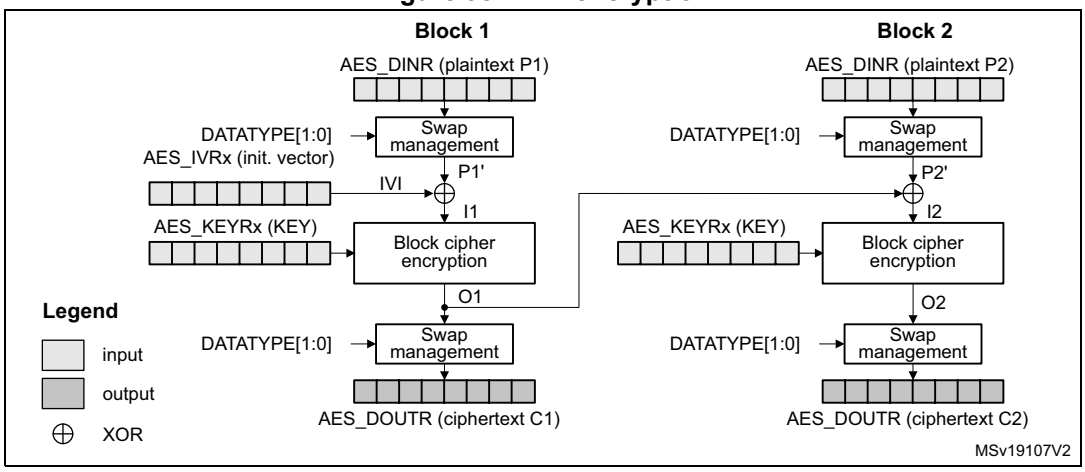



To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported 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 98* illustrates the cipher block chaining (CBC) encryption.

**Figure 98. CBC encryption**

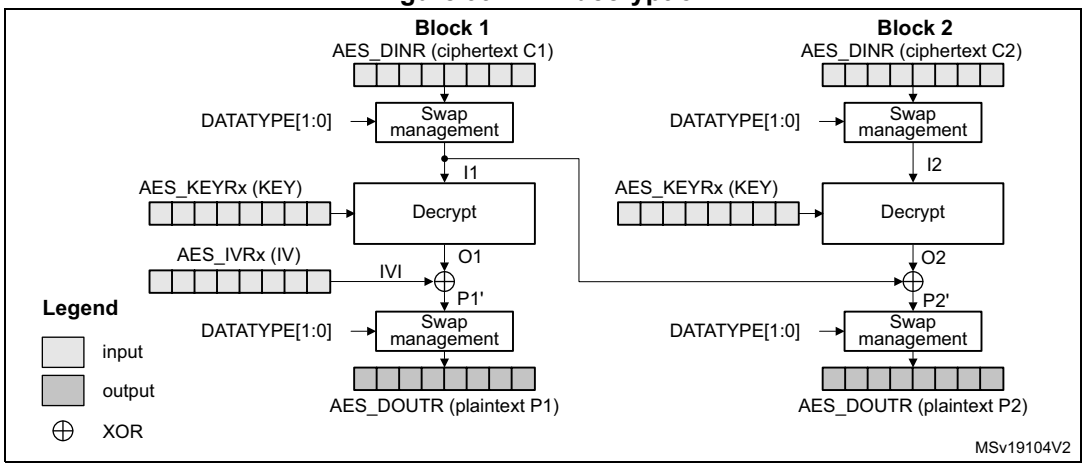



In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after 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 17.4.6: AES ciphertext stealing and data padding*.

*Figure 99* illustrates the cipher block chaining (CBC) decryption.

**Figure 99. 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 17.4.6: AES ciphertext stealing and data padding*.

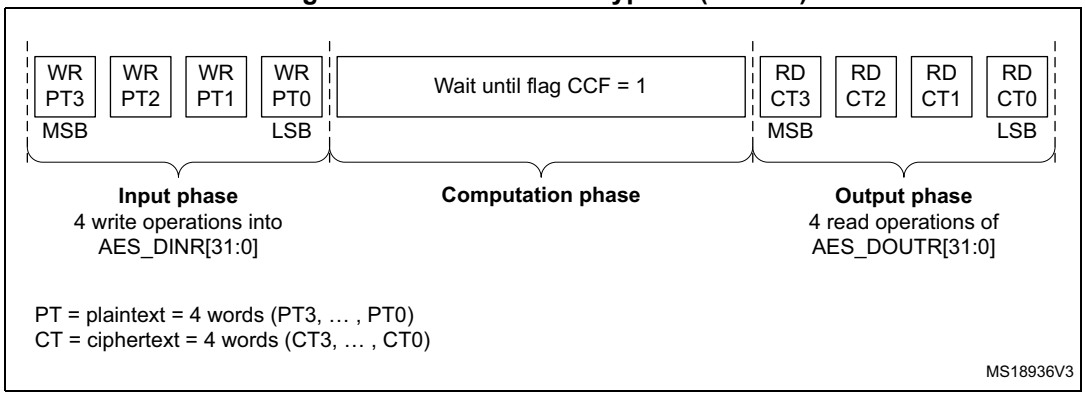
For more information on data swapping, refer to *Section 17.4.13: AES data registers and data swapping*.

### ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in *Section 17.4.4*):

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 100*.
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 100*. 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 100. ECB/CBC encryption (Mode 1)**

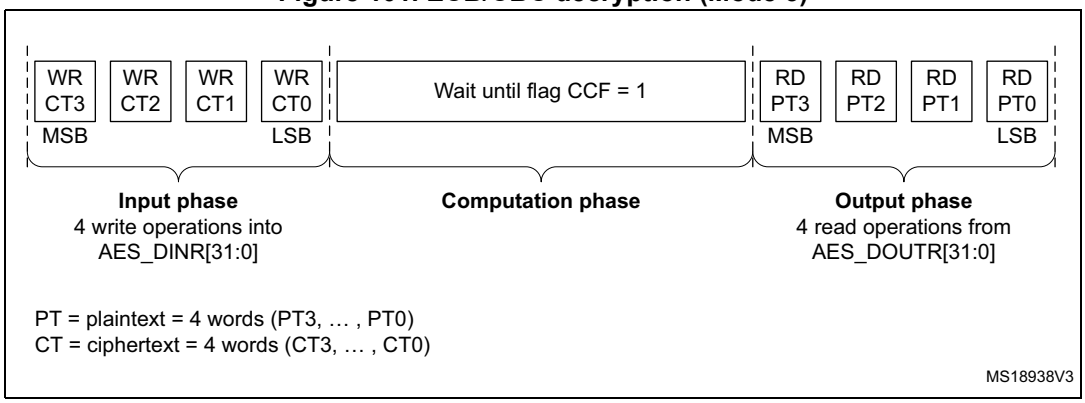


### ECB/CBC decryption sequence

The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in *Section 17.4.4*).

1. Follow the steps described in *Section 17.4.5: 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 101*.
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 101*. 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 101. 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 17.4.5: 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.*

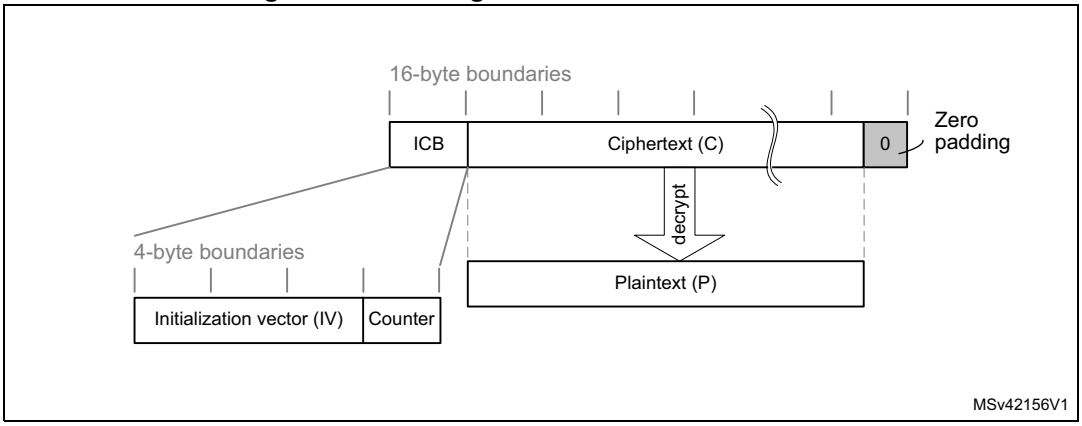
### 17.4.9 AES counter (CTR) mode

#### 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 102*.

**Figure 102. 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 103* and *Figure 104* 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 103. CTR encryption**

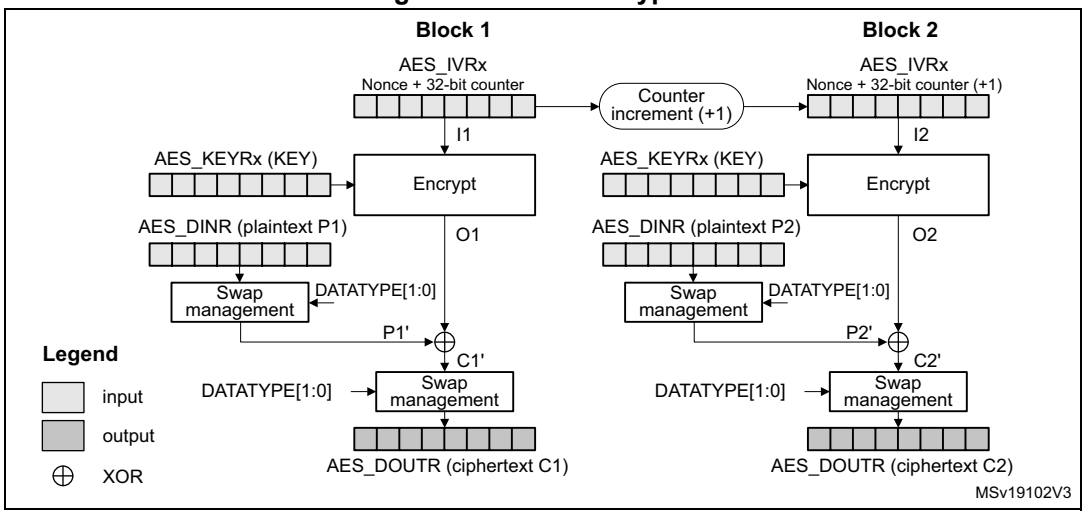
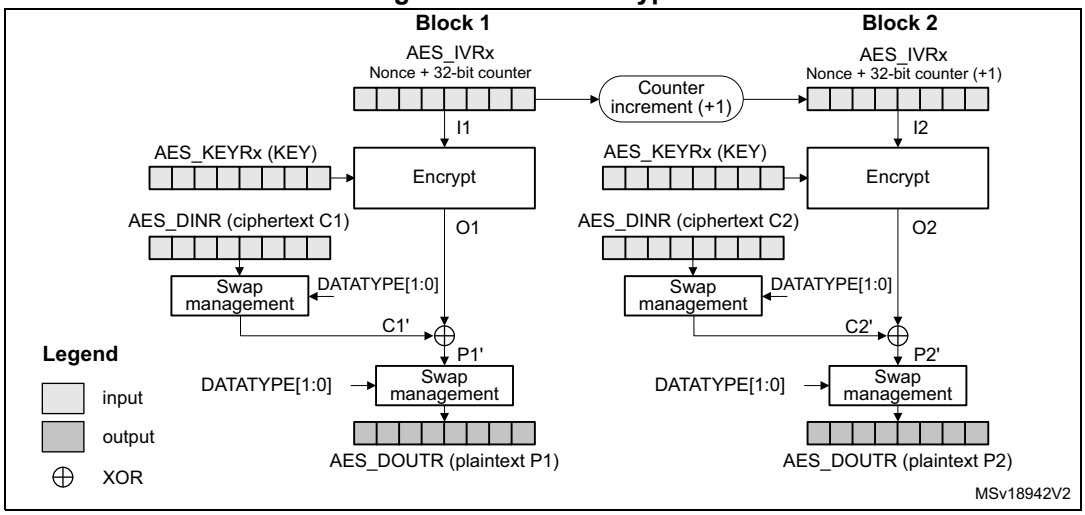



**Figure 104. CTR decryption**




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 83*.

**Table 83. 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}<br>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 83*.
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 17.4.4: 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 17.4.8: AES basic chaining modes (ECB, CBC)*.

*Note: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.*

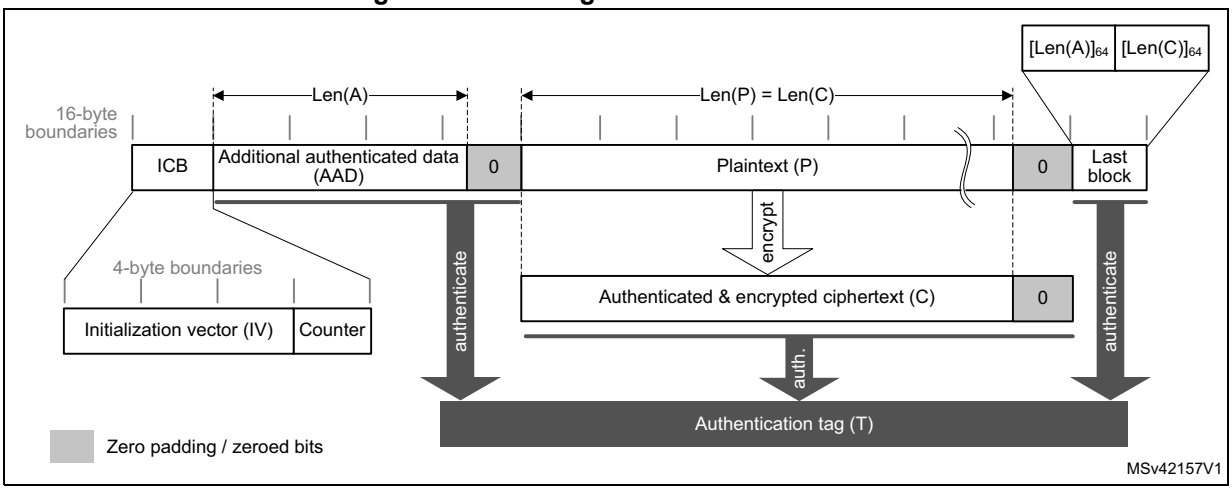
### 17.4.10 AES Galois/counter mode (GCM)

#### 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 105*.

**Figure 105. 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 84*.

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.

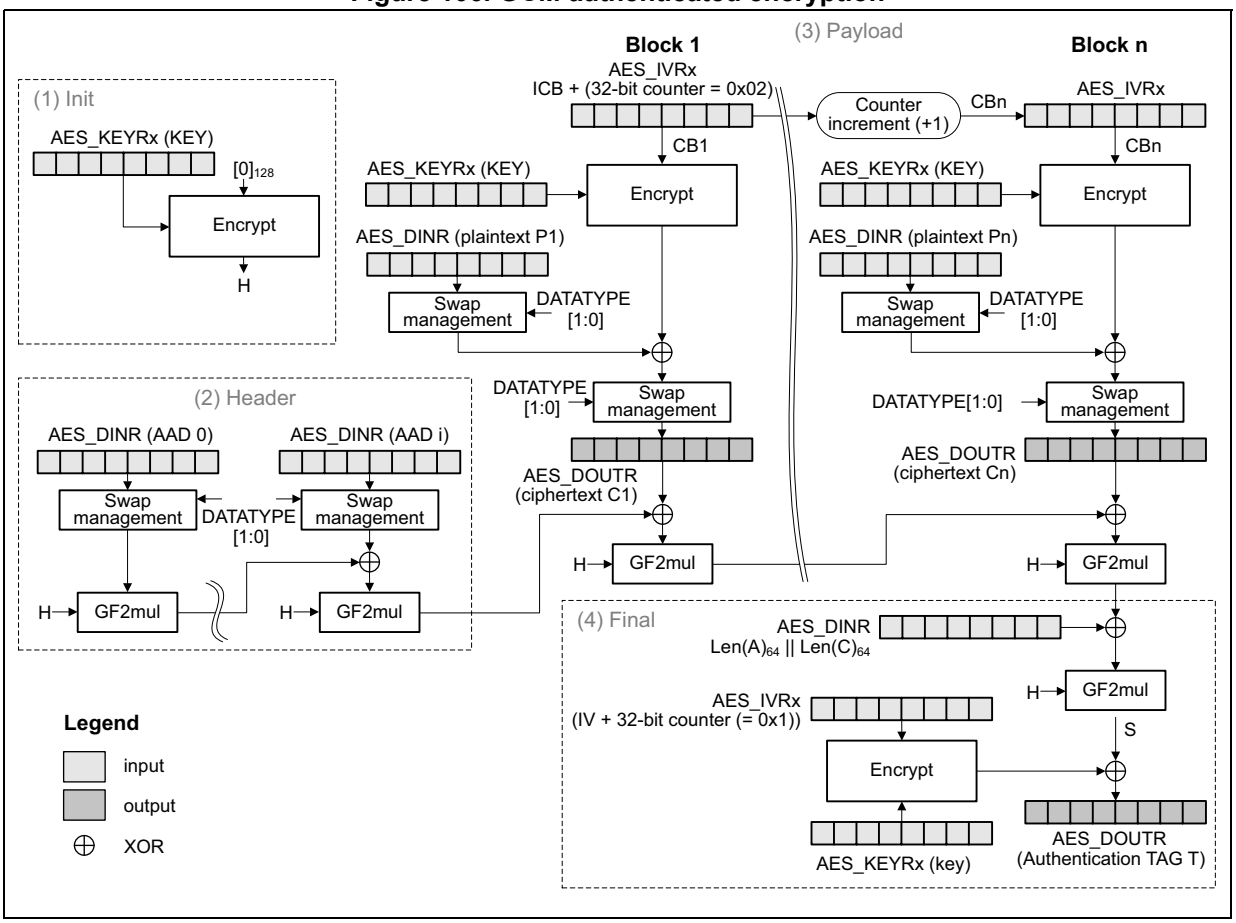
**Table 84. GCM last block definition**

| 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 106* 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 106. 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 85*).

**Table 85. 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]<br>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 85*.
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 17.4.4: 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 17.4.4: AES procedure to perform a cipher operation on page 464*, 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 84*. 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.

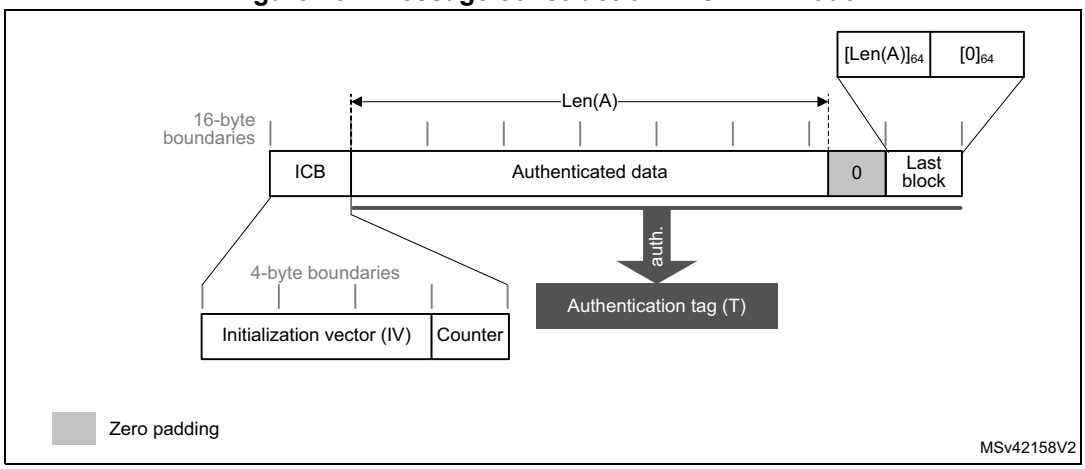
## 17.4.11 AES Galois message authentication code (GMAC)

### 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 107*.

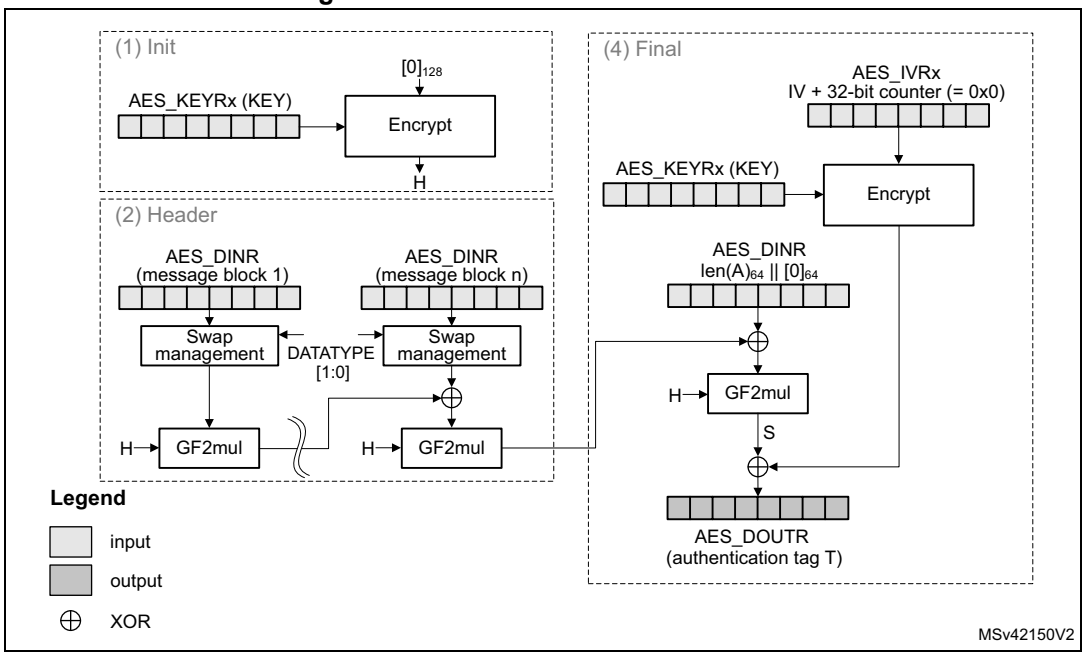
**Figure 107. Message construction in GMAC mode**




## AES GMAC processing

*Figure 108* 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 108. 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.

## 17.4.12 AES counter with CBC-MAC (CCM)

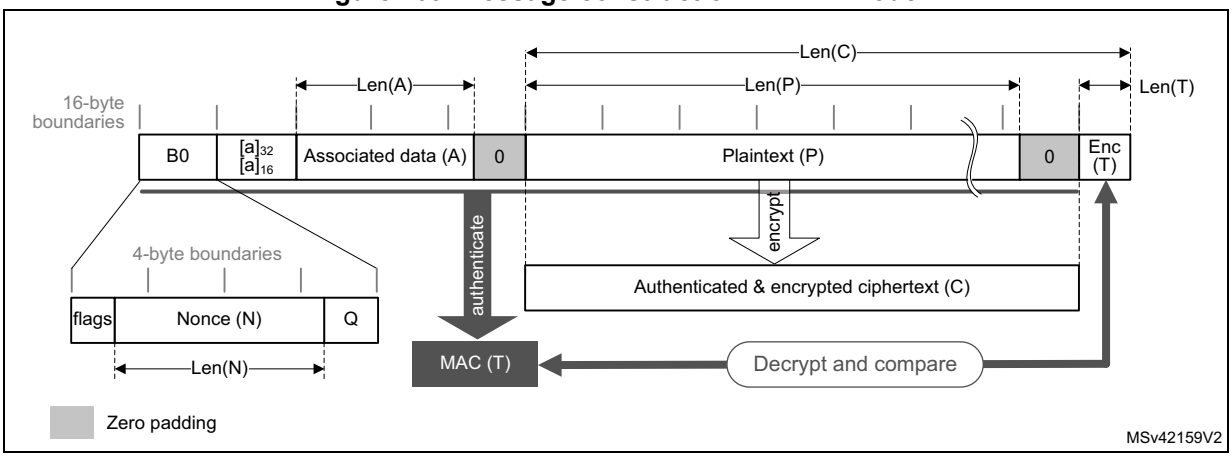
### Overview

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 109*.

**Figure 109. 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 109*). 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 109*).
- **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

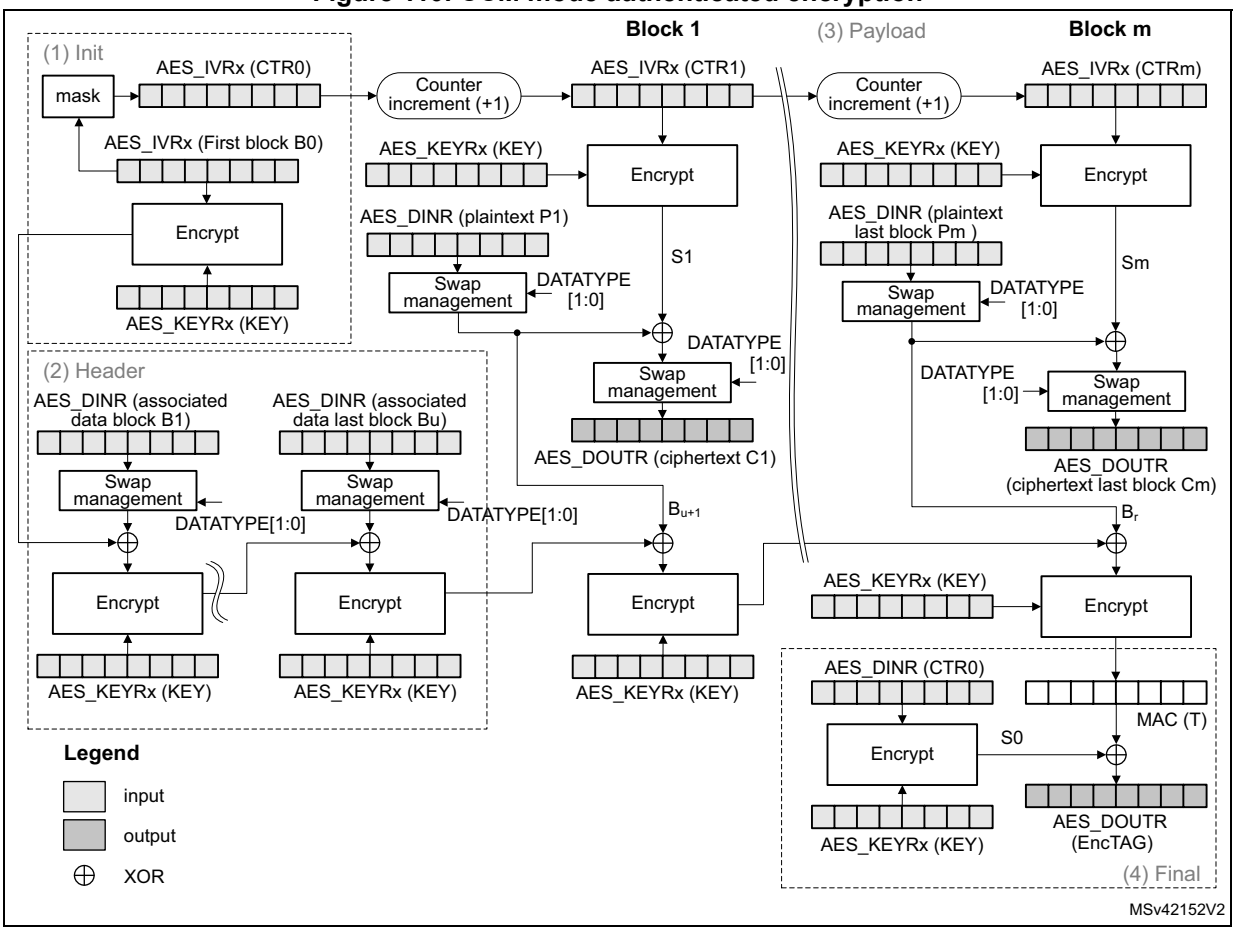
CTR1: 0710111213 141516 00000000 00000001

Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.

### CCM processing

*Figure 110* 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 110. 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 86* 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 86. 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 86*.
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 17.4.4: 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 17.4.4: AES procedure to perform a cipher operation on page 464*, 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.

## 17.4.13 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 17.4.16: AES DMA interface*.

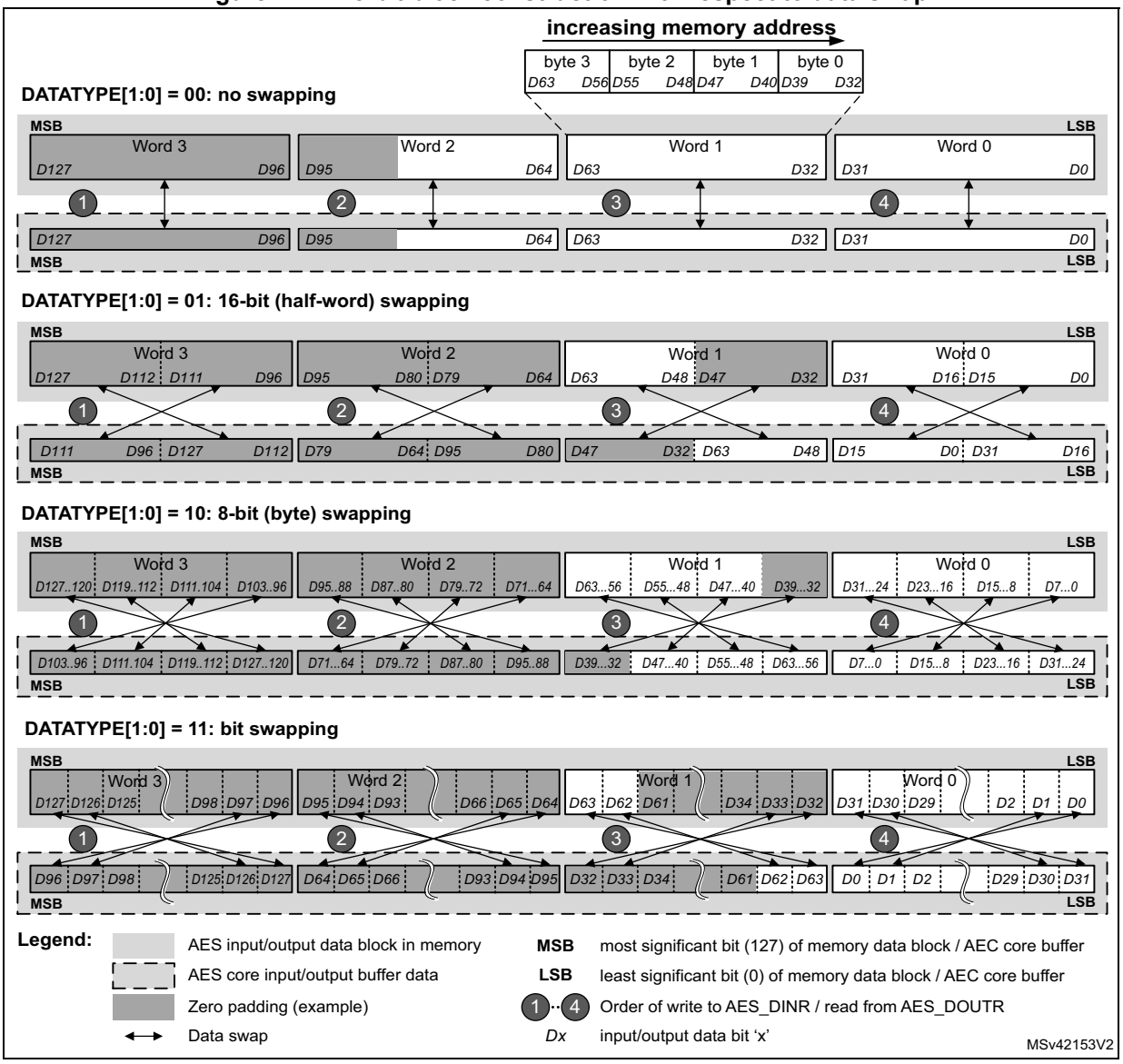
## Data swapping

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 111* 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 111. 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 111* 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

## 17.4.14 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 87*.

**Table 87. 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.

## 17.4.15 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.

### 17.4.16 AES DMA interface

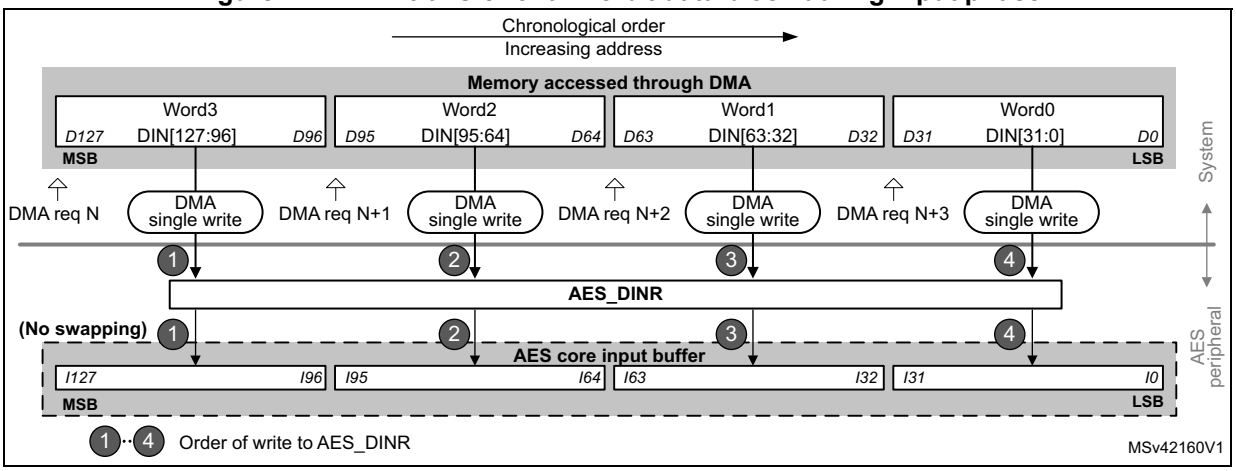
The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the AES_CR register.

#### 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 112*.

*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 17.4.4: AES procedure to perform a cipher operation.*

**Figure 112. DMA transfer of a 128-bit data block during input phase**

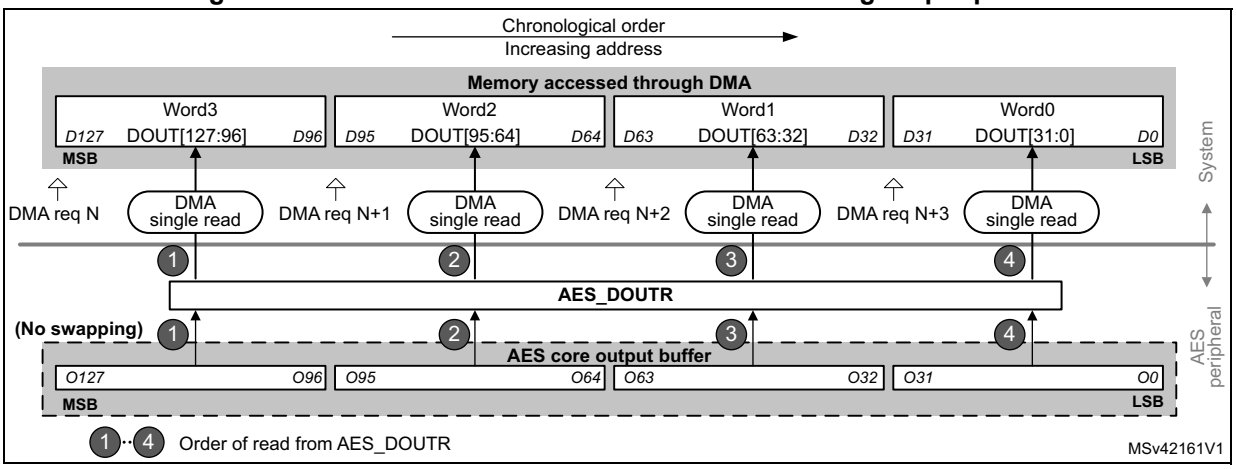



#### Data output using DMA

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 113*.

*Note: According to the message size, extra bytes might need to be discarded by application in the last block.*

**Figure 113. 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 17.4.4: AES procedure to perform a cipher operation*, subsection *Data append*, for details.

## 17.4.17 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 17.5: 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 17.5: AES interrupts*.

The WRERR flag is cleared by setting the ERRC bit of the AES_CR register.

## 17.5 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 88* gives a summary of the interrupt sources, their event flags and enable bits.

**Table 88. 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.

## 17.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

**Table 89. 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 89. 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 90. 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).

## 17.7 AES registers

### 17.7.1 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 17.4.13: 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.

### 17.7.2 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.

### 17.7.3 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 17.4.13: AES data registers and data swapping on page 488*.

### 17.7.4 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 17.4.13: AES data registers and data swapping on page 488*.

## 17.7.5 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 17.4.14: AES key registers on page 490* for more details.

### 17.7.6 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.

### 17.7.7 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.

### 17.7.8 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.

### 17.7.9 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 17.4.15: AES initialization vector registers on page 490* 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

### 17.7.10 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.

### 17.7.11 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.

### 17.7.12 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.

### 17.7.13 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.

### 17.7.14 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.

### 17.7.15 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.

### 17.7.16 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).*

### 17.7.17 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.

## 17.7.18 AES register map

**Table 91. 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 |

**Table 91. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| 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 on page 56* for the register boundary addresses.

# 18 Public key accelerator (PKA)

## 18.1 Introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(*p*) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

All needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

## 18.2 PKA main features

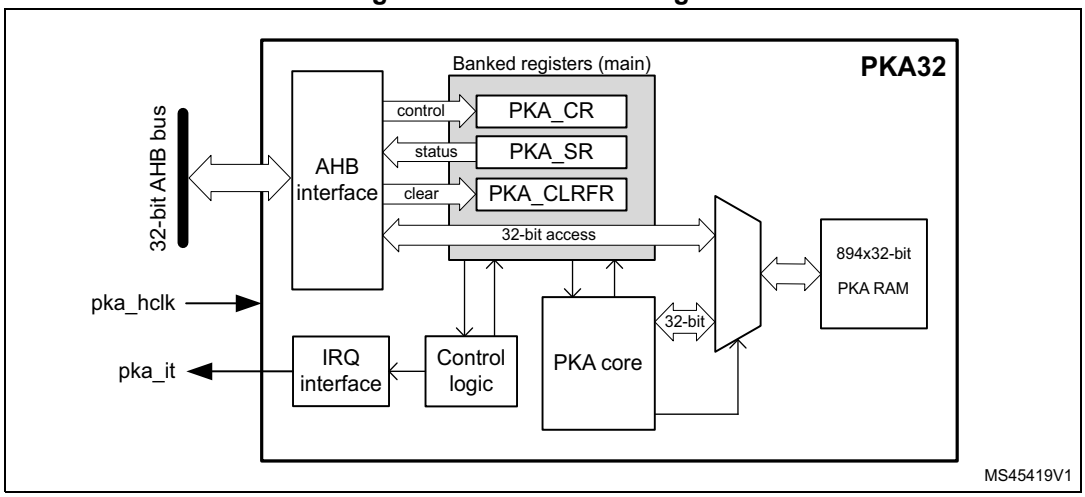
- Acceleration of RSA, DH and ECC over GF(*p*) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA Chinese Remainder Theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check
  - ECDSA signature generation and verification
- Capability to handle operands up to 3136 bits for RSA/DH and 640 bits for ECC.
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication.
- Built-in Montgomery domain inward and outward transformations.
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise, for writes, an AHB bus error is generated, and write accesses are ignored).

## 18.3 PKA functional description

### 18.3.1 PKA block diagram

*Figure 114* shows the block diagram of the public key accelerator PKA.

Figure 114. PKA block diagram




## 18.3.2 PKA internal signals

*Table 92* lists internal signals available at the IP level, not necessarily available on product bonding pads.

Table 92. Internal input/output signals

| Signal name | Signal type | Description |
|---|---|---|
| pka_hclk | Digital input | AHB bus clock |
| pka_it | Digital output | Public key accelerator IP global interrupt request |

## 18.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. The RAM receives this clock directly, the core is clocked at half the frequency.

## 18.3.4 PKA public key acceleration

### Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 3136 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information is found in *Section 18.5.1: Supported elliptic curves*.

*Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.*

A memory of 3576 bytes (894 words of 32 bits) called PKA RAM is used for providing initial data to the PKA, and for holding the results after computation is completed. Access is done though the PKA AHB interface.

### PKA operating modes

The list of operations the PKA can perform is detailed in *Table 93* and *Table 94*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register.

**Table 93. PKA integer arithmetic functions list**

| PKA_CR.MODE[5:0] | | Performed operation | Reference |
|---|---|---|---|
| Hex | Binary | | |
| 0x01 | 000001 | Montgomery parameter computation R2 mod n | *Section 18.4.2* |
| 0x0E | 001110 | Modular addition (A+B) mod n | *Section 18.4.3* |
| 0x0F | 001111 | Modular subtraction (A-B) mod n | *Section 18.4.4* |
| 0x10 | 010000 | Montgomery multiplication (AxB) mod n | *Section 18.4.5* |
| 0x00 | 000000 | Modular exponentiation $A^e$ mod n | *Section 18.4.6* |
| 0x02 | 000010 | Modular exponentiation $A^e$ mod n (fast mode) | |
| 0x08 | 001000 | Modular inversion A-1 mod n | *Section 18.4.7* |
| 0x0D | 001101 | Modular reduction A mod n | *Section 18.4.8* |
| 0x09 | 001001 | Arithmetic addition A+B | *Section 18.4.9* |
| 0x0A | 001010 | Arithmetic subtraction A-B | *Section 18.4.10* |
| 0x0B | 001011 | Arithmetic multiplication AxB | *Section 18.4.11* |
| 0x0C | 001100 | Arithmetic comparison (A=B, A>B, A<B) | *Section 18.4.12* |
| 0x07 | 000111 | RSA CRT exponentiation | *Section 18.4.13* |

**Table 94. PKA prime field (Fp) elliptic curve functions list**

| PKA_CR.MODE[5:0] | | Performed operation | Reference |
|---|---|---|---|
| Hex | Binary | | |
| 0x28 | 101000 | Point on elliptic curve Fp check | *Section 18.4.14* |
| 0x20 | 100000 | ECC scalar multiplication kP | *Section 18.4.15* |
| 0x22 | 100010 | ECC scalar multiplication kP (fast mode) | |
| 0x24 | 100100 | ECDSA sign | *Section 18.4.16* |
| 0x26 | 100110 | ECDSA verification | *Section 18.4.17* |

### Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is

supplied before starting the operation. Performance improvement is detailed in *Section 18.5.2: Computation times*.

The operations using fast mode are modular exponentiation and scalar multiplication.

## 18.3.5 Typical applications for PKA

### Introduction

The PKA can be used to accelerate a number of public key cryptographic functions. In particular:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

Specifications of the above functions are given in following publications:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

### RSA key pair

For following RSA operations a public key and a private key information are defined as below:

- Alice transmits her public key ($n$, $e$) to Bob. Numbers $n$ and $e$ are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information on above representations refer to the RSA specification.

### RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key ($n$, $e$) must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer $m$, with $0 \leq m < n$ and ($m$, $n$) being co-primes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext *m* is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\,(m_1 - m_2) \bmod p$
   d) $m = m_2 + h\,q$
3. Convert the integer message representative *m* to an encoded message EM.
4. Recover message M= DECODE(EM), where DECODE is a decoding method.

Above operations can be accelerated by PKA using *Modular exponentiation* $A^e \bmod n$ if the private key is *d*, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

*Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.*

### Elliptic curve selection

For following ECC operations curve parameters are defined as below:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 18.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (i.e. *n* x G = identity element O).

### ECDSA message signature generation

ECDSA (Elliptic Curve Digital Signature Algorithm) signature generation function principle is the following: Alice, to sign a message *m* using her private key integer $d_A$, follows the steps below.

1. Calculate e = HASH(*m*), where HASH is a cryptographic hash function.
2. Let z be the $L_n$ leftmost bits of e, where $L_n$ is the bit length of the group order *n*.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1}\,(z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (*r*, *s*).

Steps 4 to 7 are accelerated by PKA using:

- *ECDSA sign* or
- All of the operations below:
  - *ECC Fp scalar multiplication* k x P
  - *Modular reduction* A mod *n*
  - *Modular inversion* $A^{-1} \bmod n$
  - *Modular addition* and *Modular and Montgomery multiplication*

### ECDSA signature verification

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice's signature, must have a copy of her public key curve point $Q_A$.

Bob can verify that $Q_A$ is a valid curve point going through the following steps:

1. check that $Q_A$ is not equal to the identity element O
2. check that $Q_A$ is on the agreed curve
3. check that n x $Q_A$ = O.

Then Bob follows the procedure detailed below:

1. verify that *r* and *s* are integer in [1, n-1]
2. calculate *e* = HASH(*m*), where HASH is the agreed cryptographic hash function
3. let z be the $L_n$ leftmost bits of *e*
4. calculate w = $s^{-1}$ mod *n*
5. calculate $u_1$ = zw mod *n* and $u_2$ = *rw* mod *n*
6. calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. the signature is valid if $r = x_1$ (mod *n*), it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using *ECDSA verification*.

## 18.3.6 PKA procedure to perform an operation

### Enabling/disabling PKA

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed.

### Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 18.4*.

### Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set to “1”, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

*Note: When PKA is busy (BUSY = 1) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA_SR.*

#### Using precomputed Montgomery parameters (PKA fast mode)

As explained in *Section 18.3.4*, when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, *Section 18.4.5*). This is know as “fast mode”.

To manage Fast Mode usage the recommended procedure is described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in *Section 18.5.1*.
2. Program in PKA_CR register the PKA in *Montgomery parameter computation* mode (MODE=”0x1”) then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set to “1”, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.
4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in *Section 18.4*.

### 18.3.7 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the Address Error flag (ADDRERRF) is set in the PKA_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM Error Flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return zero, while writes are ignored.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see *Section 18.6* for details).

ADDRERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

## 18.4 PKA operating modes

### 18.4.1 Introduction

The various operations supported by PKA are described in the following subsections, clarifying the associated format of the input data and of the results, both stored in the PKA RAM.

The following information applies to all PKA operations.

- PKA core processes 32-bit words
- Supported operand “Size” are:
  - ROS (RSA operand size): data size is (*rsa_size*/32+1) words, with *rsa_size* equal to the chosen modulus length. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 33 words, or 1056 bits.
  - EOS (ECC operand size): data size is (*ecc_size*/32+1) words, with *ecc_size* equal to the chosen prime modulus length. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 7 words, or 224 bits.

*Note: Fractional results for above formulas are rounded up to the nearest integer since PKA core processes 32-bit words.*

*Note: The maximum ROS is 99 words (3136-bit max exponent size), while the maximum EOS is 21 words (640-bit max operand size).*

- The column indicated with Storage in the following tables indicates either the Register PKA_CR, or the address offset within the PKA, located in the PKA RAM area (starting from 0x400). To recover the physical address of an operand the application must add to the indicated offset the base address of the PKA.
- About writing parameters (“IN” direction)
  - When elements are written as input in the PKA memory, an additional word with all bits equal to zero must be added. As an example, when ECC P256 is used and when loading an input (represented on 256 bits or 8 words), an additional word is expected by the PKA and it has to be filled with zeros.
  - About endianess, for example to prepare the operation ECC Fp scalar multiplication, when application writes $x_p$ coordinate for an ECC P256 curve (EOS= 9 words) the least significant bit is to be placed in bit 0 at address offset 0x55C; the most significant bit is to be placed in bit 31 of address offset 0x578. Then, as mentioned above, word 0x00000000 should also be written at address offset 0x57C.
  - Unless indicated otherwise all operands in the tables are integers.
- About PKA outputs (“OUT” direction)
  - Unless specified in the tables PKA does not provide an error output when a wrong parameter is written by the application.

**Caution:** Validity of all input parameters to the PKA must be checked before issuing any PKA operation. Indeed, the PKA assumes that all input parameters are valid and consistent with each other.

### 18.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter ($R^2$ mod *n*) used by PKA to convert operands into the Montgomery residue system representation.

*Note: This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.*

Operation instructions for Montgomery parameter computation are summarized in *Table 95*.

**Table 95. Montgomery parameter computation**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x01 | PKA_CR | 6 bits |
| | Modulus length | (In bits, 0 ≤ value < 3136bits) | RAM@0x404 | 32 bits |
| | Modulus value n | (Odd integer only, n < $2^{3136}$) | RAM@0xD5C | ROS |
| OUT | Result: $R^2$ n | - | RAM@0x594 | |

### 18.4.3 Modular addition

Modular addition operation consists in the computation of A + B mod *n*. Operation instructions are summarized in *Table 96*.

**Table 96. Modular addition**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x0E | PKA_CR | 6 bits |
| | Operand length | (In bits, not null) | RAM@0x404 | 32 bits |
| | Operand A | (0 ≤ A < n) | RAM@0x8B4 | ROS |
| | Operand B | (0 ≤ B < n) | RAM@0xA44 | |
| | Modulus value n | ($n < 2^{3136}$) | RAM@0xD5C | |
| OUT | Result: A+B mod n | (0 ≤ result < n) | RAM@0xBD0 | |

## 18.4.4 Modular subtraction

Modular subtraction operation consists in the following computations:

- If A ≥ B result equals A - B mod *n*
- If A < B result equals A + n - B mod *n*

Operation instructions are summarized in *Table 97*.

**Table 97. Modular subtraction**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x0F | PKA_CR | 6 bits |
| | Operand length | (In bits, not null) | RAM@0x404 | 32 bits |
| | Operand A | (0 ≤ A < n) | RAM@0x8B4 | ROS |
| | Operand B | (0 ≤ B < n) | RAM@0xA44 | |
| | Modulus value n | ($n < 2^{3136}$) | RAM@0xD5C | |
| OUT | Result: A-B mod n | (0 ≤ result < n) | RAM@0xBD0 | |

## 18.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication A x B mod *n*

The method to perform above operations are described below. Note that “x” function is this operation, and A, B, C operands are in the natural domain.

1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Let's assume A is an integer in the natural domain
      Compute *r2modn* using *Montgomery parameter computation*
      Result AR= A x *r2modn* mod *n* is A in the Montgomery domain
   b) Let's assume BR is an integer in the Montgomery domain
      Result B = BR x 1 mod *n* is B in the natural domain
      Similarly, above value AR computed in a) can be converted into the natural domain by computing A = AR x 1 mod *n*
2. Simple modular multiplication A x B mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain
   c) Compute AB= AR x B mod *n*. Output is in natural domain
3. Multiple modular multiplication A x B x C mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain
   c) Compute BR = B x *r2modn* mod *n*. Output is in the Montgomery domain
   d) Compute ABR= AR x BR mod *n*. Output is in the Montgomery domain
   e) Compute CR = C x *r2modn* mod *n*. Output is in the Montgomery domain
   f) Compute ABCR= ABR x CR mod *n*. Output is in the Montgomery domain
   g) (optional) Repeat the two steps above if more operands need to be multiplied
   h) Compute ABC= ABCR x 1 mod *n* to retrieve the result in natural domain

Operation instructions for Montgomery multiplication are summarized in *Table 98*.

**Table 98. Montgomery multiplication**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x10 | PKA_CR | 6 bits |
| | Operand length | (In bits, not null) | RAM@0x404 | 32 bits |
| | Operand A | (0 ≤ A < n) | RAM@0x8B4 | ROS |
| | Operand B | (0 ≤ B < n) | RAM@0xA44 | |
| | Modulus value n | (Odd integer only, n < $2^{3136}$) | RAM@0xD5C | |
| OUT | Result: AxB mod n(1) | - | RAM@0xBD0 | |

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

### 18.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of $A^e$ mod *n*.

Operation instructions for modular exponentiation are summarized in *Table 99* (normal mode) and in *Table 100* (fast mode). Fast mode usage is explained in *Section 18.3.6*.

**Table 99. Modular exponentiation (normal mode)**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x00 | PKA_CR | 6 bits |
| IN | Exponent length | (in bits, not null) | RAM@0x400 | 32 bits |
| | Operand length | (in bits, not null) | RAM@0x404 | |
| IN/OUT | Operand A (base of exponentiation) | (0 ≤ A < n) | RAM@0xA44 | ROS |
| IN | Exponent e | (0 ≤ e < n) | RAM@0xBD0 | |
| | Modulus value n | (Odd integer only, $n < 2^{3136}$) | RAM@0xD5C | |
| OUT | Result: $A^e$ mod n | (0 ≤ result < n) | RAM@0x724 | |

**Table 100. Modular exponentiation (fast mode)**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x02 | PKA_CR | 6 bits |
| IN | Exponent length | (in bits, not null) | RAM@0x400 | 32 bits |
| | Operand length | (in bits, not null) | RAM@0x404 | |
| IN/OUT | Operand A (base of exponentiation) | (0 ≤ A < n) | RAM@0xA44 | ROS |
| IN | Exponent e | (0 ≤ e < n) | RAM@0xBD0 | |
| | Modulus value n | (Odd integer only, $n < 2^{3136}$) | RAM@0xD5C | |
| IN/OUT | Montgomery param R2 mod n | (mandatory) | RAM@0x594 | |
| OUT | Result: $A^e$ mod n | (0 ≤ result < n) | RAM@0x724 | |

### 18.4.7 Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse $A^{-1}$ mod *n*. If the modulus *n* is prime, for all values of A (1 ≤ A < *n*) modular inversion output is valid. If the modulus *n* is not prime, A has an inverse only if the largest common divisor between A and *n* is 1.

If the operand A is a divisor of the modulus *n,* the result is a multiple of a factor of *n*.

Operation instructions for modular inversion are summarized in *Table 101*.

**Table 101. Modular inversion**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x08 | PKA_CR | 6 bits |
| | Operand length | (In bits, not null) | RAM@0x404 | 32 bits |
| | Operand A | (0 ≤ A < n) | RAM@0x8B4 | ROS |
| | Modulus value n | (Odd integer only, $n < 2^{3136}$) | RAM@0xA44 | |
| OUT | Result: $A^{-1}$ mod n | 0 < result < n | RAM@0xBD0 | |

### 18.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of A divided by n. Operation instructions are summarized in *Table 102*.

**Table 102. Modular reduction**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x0D | PKA_CR | 6 bits |
| | Operand length | (In bits, not null) | RAM@0x400 | 32 bits |
| | Modulus length | (In bits, 8 < value < 3136) | RAM@0x404 | |
| | Operand A | ($0 \le A < 2n < 2^{3136}$) | RAM@0x8B4 | ROS |
| | Modulus value n | (Odd integer only, $n < 2^{3136}$) | RAM@0xA44 | |
| OUT | Result A mod n | (0 < result < n) | RAM@0xBD0 | |

### 18.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of A + B. Operation instructions are summarized in *Table 103*.

**Table 103. Arithmetic addition**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x09 | PKA_CR | 6 bits |
| | Operand length M | (In bits, not null) | RAM@0x404 | 32 bits |
| | Operand A | ($0 \le < A < 2^M$) | RAM@0x8B4 | ROS |
| | Operand B | ($0 \le < B < 2^M$) | RAM@0xA44 | |
| OUT | Result: A+B | ($0 \le$ result $< 2^{M+1}$) | RAM@0xBD0 | ROS + 1 |

### 18.4.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If $A \ge B$ result equals A - B
- If A < B and M/32 residue is >0 result equals $A + 2^{int(M/32)*32+1} - B$
- If A < B and M/32 residue is 0 result equals $A + 2^{int(M/32)*32} - B$

Operation instructions are summarized in *Table 104*.

**Table 104. Arithmetic subtraction**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x0A | PKA_CR | 6 bits |
| | Operand length M | (In bits, not null) | RAM@0x404 | 32 bits |
| | Operand A | ($0 \le < A < 2^M$) | RAM@0x8B4 | ROS |
| | Operand B | ($0 \le < B < 2^M$) | RAM@0xA44 | |
| OUT | Result: A-B | ($0 \le$ result $< 2^M$) | RAM@0xBD0 | |

### 18.4.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in *Table 105*.

**Table 105. Arithmetic multiplication**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x0B | PKA_CR | 6 bits |
| | Operand length M | (In bits, not null) | RAM@0x404 | 32 bits |
| | Operand A | $(0 \leq < A < 2^M)$ | RAM@0x8B4 | ROS |
| | Operand B | $(0 \leq < B < 2^M)$ | RAM@0xA44 | |
| OUT | Result: AxB | $(0 \leq \text{result} < 2^M)$ | RAM@0xBD0 | 2xROS |

### 18.4.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:

- If A=B then result=0x0
- If A>B then result=0x1
- If A<B then result=0x2

Operation instructions for arithmetic comparison are summarized in *Table 106*.

**Table 106. Arithmetic comparison**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x0C | PKA_CR | 6 bits |
| | Operand length M | (In bits, not null) | RAM@0x404 | 32 bits |
| | Operand A | $(0 \leq < A < 2^M)$ | RAM@0x8B4 | ROS |
| | Operand B | $(0 \leq < B < 2^M)$ | RAM@0xA44 | |
| OUT | Result A=B or A>B or A<B | 0x0, 0x1 or 0x02 | RAM@0xBD0 | 32 bits |

### 18.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries like OpenSSL RSA use the following optimization for decryption and signing based on the Chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p - 1)$
- $d_Q = d \bmod (q - 1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation $m = A^d$ (mod pq) more efficiently as follows:

- $m_1 = A^{dP}$ mod $p$
- $m_2 = A^{dQ}$ mod $p$
- $h = q_{inv} (m_1 - m_2)$ mod $p$, with $m_1 > m_2$
- $m = m_2 + hq$

Operation instructions for computing CRT exponentiation $A^d$ mod pq are summarized in *Table 107*.

**Table 107. CRT exponentiation**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x07 | PKA_CR | 6 bits |
| IN | Operand length | (in bits, not null) | RAM@0x404 | 32 bits |
| IN | Operand $d_P$ | $(0 \le d_P < 2^{M/2})$ | RAM@0x65C | ROS/2 |
| | Operand $d_Q$ | $(0 \le d_Q < 2^{M/2})$ | RAM@0xBD0 | |
| | Operand $q_{inv}$ | $(0 \le q_{inv} < 2^{M/2})$ | RAM@0x7EC | |
| | Prime $p$(1) | $(0 \le p < 2^{M/2})$ | RAM@0x97C | |
| | Prime $q$(1) | $(0 \le q < 2^{M/2})$ | RAM@0xD5C | |
| IN | Operand A | $(0 \le A < 2^{M/2})$ | RAM@0xEEC | ROS |
| OUT | Result: $A^d$ mod $pq$ | (0 ≤ result < $pq$) | RAM@0x724 | |

1. Must be different from 2.

### 18.4.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point P (x, y) satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b)$ mod $p$, where $a$ and $b$ are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in *Table 108*.

**Table 108. Point on elliptic curve Fp check**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x28 | PKA_CR | 6 bits |
| | Modulus length | (In bits, not null, 8 < value < 640) | RAM@0x404 | 32 bits |
| | Curve coefficient *a* sign | 0x0: positive<br>0x1: negative | RAM@0x408 | |
| | Curve coefficient \|a\| | (Absolute value, \|a\| < p) | RAM@0x40C | EOS |
| | Curve coefficient *b* | (\|b\| < p) | RAM@0x7FC | |
| | Curve modulus value *p* | (Odd integer prime, 0 < p < 2640) | RAM@0x460 | |
| | Point P coordinate *x* | (x < *p*) | RAM@0x55C | |
| | Point P coordinate *y* | (y < *p*) | RAM@0x5B0 | |
| OUT | Result: P on curve | 0x0: point on curve<br>Not 0x0: point not on curve | RAM@0x400 | 32 bits |

## 18.4.15 ECC Fp scalar multiplication

This operation consists in the computation of a k x P ($x_P$, $y_P$), where P is a point on a curve over prime fields and “x” is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in *Table 109* (normal mode) and *Table 110* (fast mode). Fast mode usage is explained in *Section 18.3.6*.

**Table 109. ECC Fp scalar multiplication**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x20 | PKA_CR | 6 bits |
| IN | Scalar multiplier k length | (In bits, not null, 8 < value < 640) | RAM@0x400 | 32 bits |
| | Modulus length | (In bits, not null, 8 < value < 640) | RAM@0x404 | |
| | Curve coefficient *a* sign | 0x0: positive<br>0x1: negative | RAM@0x408 | |
| IN | Curve coefficient \|a\| | (Absolute value, \|a\| < p) | RAM@0x40C | EOS |
| | Curve modulus value *p* | (Odd integer prime, 0 < p < $2^{640}$) | RAM@0x460 | |
| | Scalar multiplier k | (0 ≤ k < $2^{640}$) | RAM@0x508 | |
| | Point P coordinate $x_P$ | (x < *p*) | RAM@0x55C | |
| | Point P coordinate $y_P$ | (y < *p*) | RAM@0x5B0 | |
| OUT | Result: k x P coordinate *x* | (result < *p*) | RAM@0x55C | |
| | Result: k x P coordinate *y* | (result < *p*) | RAM@0x5B0 | |

**Table 110. ECC Fp scalar multiplication (Fast Mode)**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x22 | PKA_CR | 6 bits |
| IN | Scalar multiplier k length | (In bits, not null, 8 < value < 640) | RAM@0x400 | 32 bits |
| | Modulus length | (In bits, not null, 8 < value < 640) | RAM@0x404 | |
| | Curve coefficient *a* sign | 0x0: positive<br>0x1: negative | RAM@0x408 | |
| IN | Curve coefficient \|a\| | (Absolute value, \|a\| < p) | RAM@0x40C | EOS |
| | Curve modulus value *p* | (Odd integer prime, $0 < p < 2^{640}$) | RAM@0x460 | |
| | Scalar multiplier k | ($0 \le k < 2^{640}$) | RAM@0x508 | |
| | Point P coordinate $x_P$ | (x < *p*) | RAM@0x55C | |
| | Point P coordinate $y_P$ | (y < *p*) | RAM@0x5B0 | |
| IN | Montgomery parameter $R^2$ mod *p* | (mandatory) | RAM@0x4B4 | |
| OUT | Result: k x P coordinate *x* | (result < *p*) | RAM@0x55C | |
| | Result: k x P coordinate *y* | (result < *p*) | RAM@0x5B0 | |

When performing this operation following special cases should be noted:

- For k = 0, this function returns a point at infinity, that is (0, 0) if curve parameter b is nonzero and (0, 1) otherwise. For k different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation should be carried out.
- For k < 0 (i.e. a negative scalar multiplication is required) multiplier absolute value k = |-k| should be provided to the PKA. After the computation completion, the formula -P = (x, -y) can be used to compute the y coordinate of the effective final result (the x coordinate remains the same).

### 18.4.16 ECDSA sign

ECDSA signing operation (outlined in *Section 18.3.5*) is summarized in *Table 111* (input parameters) and in *Table 112* (output parameters).

The application should check if the output error is equal to zero, if it is different from zero a new k should be generated and the ECDSA sign operation should be repeated.

**Table 111. ECDSA sign - Inputs**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x24 | PKA_CR | 6 bits |
| | Curve prime order n length | (in bits, not null) | RAM@0x400 | 32 bits |
| | Curve modulus p length | (in bits, 8 < value < 640) | RAM@0x404 | |
| | Curve coefficient a sign | 0x0: positive<br>0x1: negative | RAM@0x408 | |
| | Curve coefficient \|a\| | (Absolute value, \|a\| < p) | RAM@0x40C | EOS |
| | Curve modulus value *p* | (Odd integer prime, $0 < p < 2^{640}$) | RAM@0x460 | |
| | Integer k(1) | ($0 \le k < 2^{640}$) | RAM@0x508 | |
| | Curve base point G coordinate x | (x < *p*) | RAM@0x55C | |
| | Curve base point G coordinate y | (y < *p*) | RAM@0x5B0 | |
| | Hash of message z | (z < 2M) | RAM@0xDE8 | |
| | Private key d | (positive integer) | RAM@0xE3C | |
| | Curve prime order n | (integer prime) | RAM@0xE94 | |

1. This integer is usually a cryptographically secure random number, but in some cases k could be deterministically generated.

**Table 112. ECDSA sign - Outputs**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| OUT | Signature part *r* | (0 < *r* < n) | RAM@0x700 | EOS |
| | Signature part *s* | (0 < *s* < n) | RAM@0x754 | |
| ERROR | Result of signature | – 0x0: no error<br>– 0x1: signature part r is equal to 0<br>– 0x2: signature part s is equal to 0 | RAM@0xEE8 | 32 bits |

*Note: If error output is different from zero the content of the PKA memory should be cleared to avoid leaking information about the private key.*

### Extended ECDSA support

PKA also supports Extended ECDSA signature, for which the inputs and the outputs have the same ECDSA signature (*Table 111* and *Table 112*, respectively), with the addition of the coordinates of the point kG. This extra output is defined in *Table 113*.

**Table 113. Extended ECDSA sign (extra outputs)**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| OUT | Curve point kG coordinate $x_1$ | $(0 \le x_1 < p)$ | RAM@0x103C | EOS |
| | Curve point kG coordinate $y_1$ | $(0 \le y_1 < p)$ | RAM@0x1090 | |

## 18.4.17 ECDSA verification

ECDSA verification operation (outlined in *Section 18.3.5*) is summarized in *Table 114* (input parameters) and *Table 115* (output parameters).

The epplication should check if the output error is equal to zero, if it is different from zero, the signature is not verified.

**Table 114. ECDSA verification (inputs)**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| IN | MODE | 0x26 | PKA_CR | 6 bits |
| | Curve prime order n length | (In bits, not null) | RAM@0x404 | 32 bits |
| | Curve modulus p length | (In bits, not null, 8 < value < 640) | RAM@0x4B4 | |
| | Curve coefficient a sign | 0x0: positive<br>0x1: negative | RAM@0x45C | |
| | Curve coefficient \|a\| | (Absolute value, \|a\| < p) | RAM@0x460 | EOS |
| | Curve modulus value *p* | (Odd integer prime, $0 < p < 2^{640}$) | RAM@0x4B8 | |
| | Curve base point G coordinate x | (x < *p*) | RAM@0x5E8 | |
| | Curve base point G coordinate y | (y < *p*) | RAM@0x63C | |
| | Public-key curve point Q coordinate $x_Q$ | ($x_Q$ < *p*) | RAM@0xF40 | |
| | Public-key curve point Q coordinate $y_Q$ | ($y_Q$ < *p*) | RAM@0xF94 | |
| | Signature part *r* | (0 < *r* < n) | RAM@0x1098 | |
| | Signature part *s* | (0 < *s* < n) | RAM@0xA44 | |
| | Hash of message *z* | ($z < 2^M$) | RAM@0xFE8 | |
| | Curve prime order *n* | (integer prime) | RAM@0xD5C | |

**Table 115. ECDSA verification (outputs)**

| Parameters with direction | | Value (Note) | Storage | Size |
|---|---|---|---|---|
| OUT | Result: ECDSA verify | 0x0: valid signature<br>Not 0x0: invalid signature | RAM@0x5B0 | 32 bits |

## 18.5 Example of configurations and processing times

### 18.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curvescan be described with a short Weierstrass equation $y^2 = x^3 + ax + b \pmod{p}$.

*Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA. The maximum supported operand size for ECC operations is 640 bits.*

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field GF(*p*)
- the (usually prime) integer *n*, the order of the group generated by G, defined below
- the base point of the curve G, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. **a defined as *p*-|*a*|** in the finite field GF(*p*), for example **p-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

*Table 116* summarizes the family of curves supported by PKA for ECC operations.

**Table 116. Family of supported curves for ECC operations**

| Curve name | Standard | Reference |
|---|---|---|
| P-192 | NIST | *Digital Signature Standard (DSS)*, NIST FIPS 186-4 |
| P-224 | | |
| P-256 | | |
| P-384 | | |
| P-521 | | |

**Table 116. Family of supported curves for ECC operations (continued)**

| Curve name | Standard | Reference | |
|---|---|---|---|
| brainpoolP224r1, brainpoolP224t1 | IETF | – *Brainpool Elliptic Curves*, IETF RFC 5639<br>– *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932 | https://tools.ietf.org |
| brainpoolP256r1, brainpoolP256t1 | | | |
| brainpoolP320r1, brainpoolP320t1 | | | |
| brainpoolP384r1, brainpoolP384t1 | | | |
| brainpoolP512r1, brainpoolP512t1 | | | |
| secp192k1, secp192r1 | SEC | *Standards for Efficient Cryptography SEC 2 curves* | https://www.secg.org |
| secp224k1, secp224r1 | | | |
| secp256k1, secp256r1 | | | |
| secp384r1 | | | |
| secp521r1 | | | |
| Recommended curve parameters for public key cryptographic algorithm SM2 | OSCCA | – *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010<br>– *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018 | |

### 18.5.2 Computation times

The following tables summarize the PKA computation times, expressed in clock cycles.

**Table 117. Modular exponentiation computation times**

| Exponent length (in bits) | Mode | Modulus length (in bits) | | |
|---|---|---|---|---|
| | | 1024 | 2048 | 3072 |
| 3 | Normal | 304000 | 814000 | 1728000 |
| | Fast | 46000 | 164000 | 356000 |
| 17 | Normal | 326000 | 896000 | 1910000 |
| | Fast | 68000 | 246000 | 534000 |
| $2^{16} + 1$ | Normal | 416000 | 1222000 | 2616000 |
| | Fast | 158000 | 572000 | 1244000 |
| 1024 | Normal | 11664000 | - | - |
| | Fast | 11280000 | - | - |
| | CRT(1) | 3546000 | - | - |
| 2048 | Normal | - | 83834000 | - |
| | Fast | - | 82046000 | - |
| | CRT(1) | - | 23468000 | - |
| 3072 | Normal | - | - | 274954000 |
| | Fast | - | - | 273522000 |
| | CRT(1) | - | - | 73378000 |

1. CRT stands for chinese remainder theorem optimization (MODE bitfield = 0x07).

**Table 118. ECC scalar multiplication computation times(1)**

| Mode | Modulus length (in bits) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 160 | 192 | 256 | 320 | 384 | 512 | 521 |
| Normal | 1634000 | 2500000 | 4924000 | 8508000 | 13642000 | 28890000 | 33160000 |
| Fast | 1630000 | 2494000 | 4916000 | 8494000 | 13614000 | 28842000 | 33158000 |

1. These times depend on the number of “1”s included in the scalar parameter.

**Table 119. ECDSA signature average computation times(1) (2)**

| Modulus length (in bits) | | | | | | |
|---|---|---|---|---|---|---|
| 160 | 192 | 256 | 320 | 384 | 512 | 521 |
| 1760000 | 2664000 | 5249000 | 9016000 | 14596000 | 30618000 | 35540000 |

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

**Table 120. ECDSA verification average computation times**

| Modulus length (in bits) | | | | | | |
|---|---|---|---|---|---|---|
| **160** | **192** | **256** | **320** | **384** | **512** | **521** |
| 3500000 | 5350000 | 10498000 | 18126000 | 29118000 | 61346000 | 71588000 |

**Table 121. Point on elliptic curve Fp check average computation times**

| Modulus length (in bits) | | | | | |
|---|---|---|---|---|---|
| **160** | **192** | **256** | **320** | **384** | **512** |
| 10800 | 14200 | 20400 | 31000 | 49600 | 82400 |

**Table 122. Montgomery parameters average computation times(1)**

| Modulus length (in bits) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **160** | **192** | **256** | **320** | **384** | **512** | **521** | **1024** | **2048** | **3072** |
| 4518 | 7846 | 11848 | 14902 | 21682 | 35012 | 64000 | 119536 | 466146 | 1104642 |

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

## 18.6 PKA interrupts

There are three individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. access to unmapped address (ADDRERRF), see *Section 18.3.7*
2. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 18.3.7*
3. PKA end of operation (PROCENDF)

The three interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

*Table 123* gives a summary of the available features.

**Table 123. PKA interrupt requests**

| Interrupt acronym | Interrupt event | Event flag | Enable control bit | Interrupt clear method |
|---|---|---|---|---|
| PKA | Access to unmapped address error | ADDRERRF | ADDRERRIE | Set ADDRERRFC bit |
| | PKA RAM access error | RAMERRF | RAMERRIE | Set RAMERRFC bit |
| | PKA end of operation | PROCENDF | PROCENDIE | Set PROCENDFC bit |

## 18.7 PKA registers

### 18.7.1 PKA control register (PKA_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. | ADDR ERRIE | RAM ERRIE | Res. | PROC ENDIE | Res. |
| | | | | | | | | | | | rw | rw | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | MODE[5:0] | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | START | EN |
| | | rw | rw | rw | rw | rw | rw | | | | | | | rw | rw |

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

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code
000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
100000: Montgomery parameter computation then ECC scalar multiplication
100010: ECC scalar multiplication only (Montgomery parameter must be loaded first)
100100: ECDSA sign
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
Others: Reserved

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

Bit 1 **START**: start the operation

Writing 1 to this bit starts the operation which is selected by MODE[5:0], using the operands and data already written to the PKA RAM. This bit is always read as 0.

*Note: START is ignored if PKA is busy.*

Bit 0 **EN**: PKA enable.

0: Disable PKA

1: Enable PKA

*Note: When EN=0 PKA RAM can still be accessed by the application.*

### 18.7.2 PKA status register (PKA_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. | ADDR ERRF | RAM ERRF | Res. | PROC ENDF | BUSY |
| | | | | | | | | | | | 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |

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

Bit 20 **ADDRERRF**: Address error flag

0: No Address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA End of Operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: PKA operation is in progress

This bit is set to 1 whenever START bit in the PKA_CR is set. It is automatically cleared when the computation is complete, meaning that PKA RAM can be safely accessed and a new operation can be started.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode the peripheral is busy for a couple of cycles, then it aborts automatically the operation and go back to ready (BUSY bit is set to 0).

Bits 15:0 Reserved, must be kept at reset value.

### 18.7.3 PKA clear flag register (PKA_CLRFR)

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. | ADDR ERRFC | RAM ERRFC | Res. | PROC ENDFC | Res. |
| | | | | | | | | | | | w | w | | w | |

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

Bit 20 **ADDRERRFC**: Clear Address error flag
0: No action
1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
0: No action
1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA End of Operation flag
0: No action
1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

*Note: Reading PKA_CLRFR returns all 0s.*

### 18.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 3576 bytes (max word offset: 0x11F4).

### 18.7.5 PKA register map

**Table 124. PKA register map and reset values**

| Offset | Register name | 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 | PKA_CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ADDRERRIE | RAMERRIE | Res. | PROCENDIE | Res. | Res. | Res. | MODE[5:0] | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | START | EN |
| | Reset value | | | | | | | | | | | | 0 | 0 | | 0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 |
| 0x004 | PKA_SR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ADDRERRF | RAMERRF | Res. | PROCENDF | BUSY | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | 0 | 0 | | 0 | 0 | | | | | | | | | | | | | | | | |
| 0x008 | PKA_CLRFR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ADDRERRFC | RAMERRFC | Res. | PROCENDFC | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | 0 | 0 | | 0 | | | | | | | | | | | | | | | | | |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 19 Advanced-control timer (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

## 19.1 TIM1 introduction

The advanced-control timer (TIM1) 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).

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.

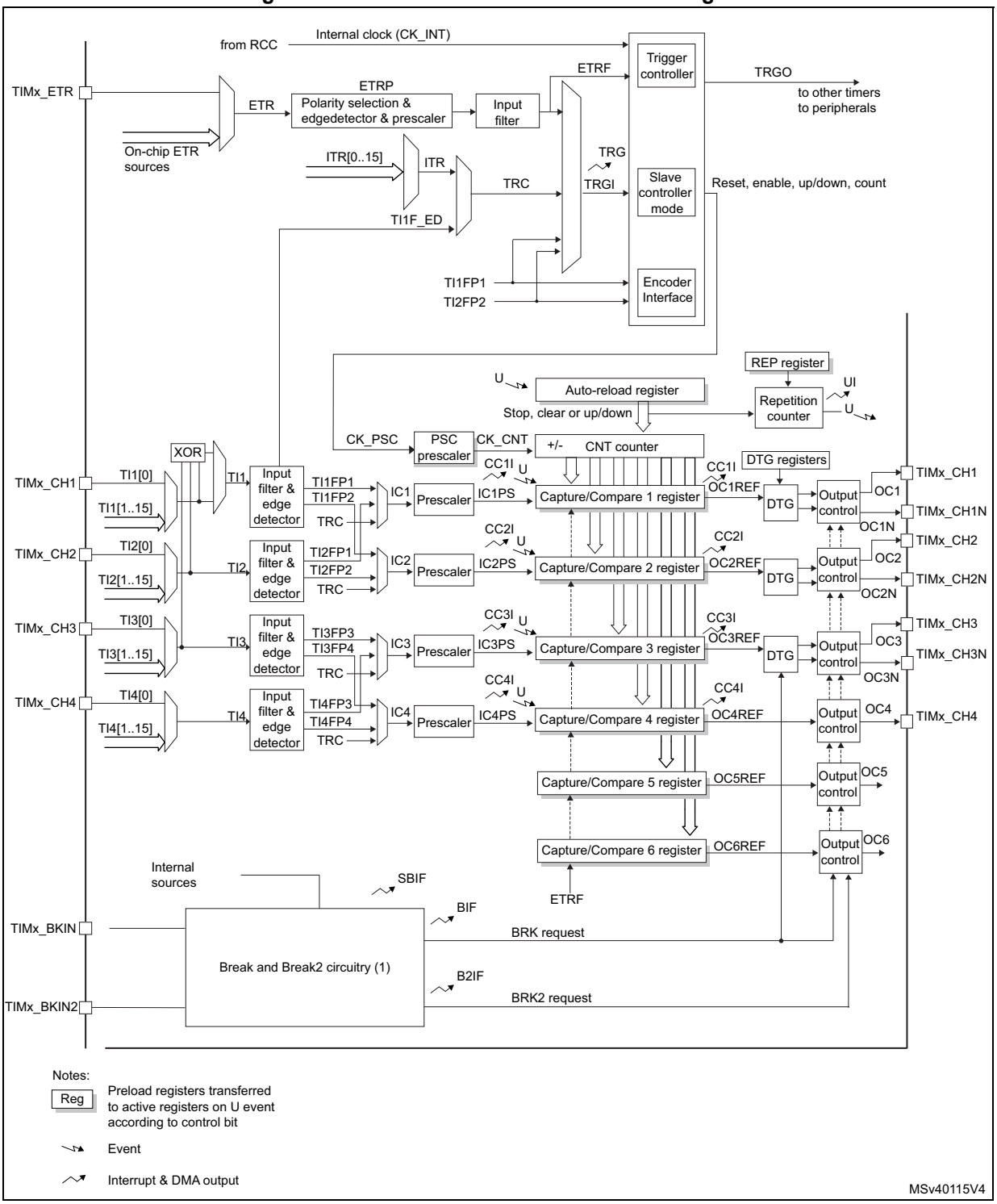
The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 19.3.26: Timer synchronization*.

## 19.2 TIM1 main features

TIM1 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 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs 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.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- 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 and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

**Figure 115. Advanced-control timer block diagram**




1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 7.2.11: Clock security system (CSS) on HSE*
   - A PVD output
   - SRAM parity error signal
   - CPU1 Cortex®-M4 LOCKUP (Hardfault) output.

## 19.3 TIM1 functional description

### 19.3.1 Time-base unit

The main block of the programmable advanced-control 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 116* and *Figure 117* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 116. Counter timing diagram with prescaler division change from 1 to 2**

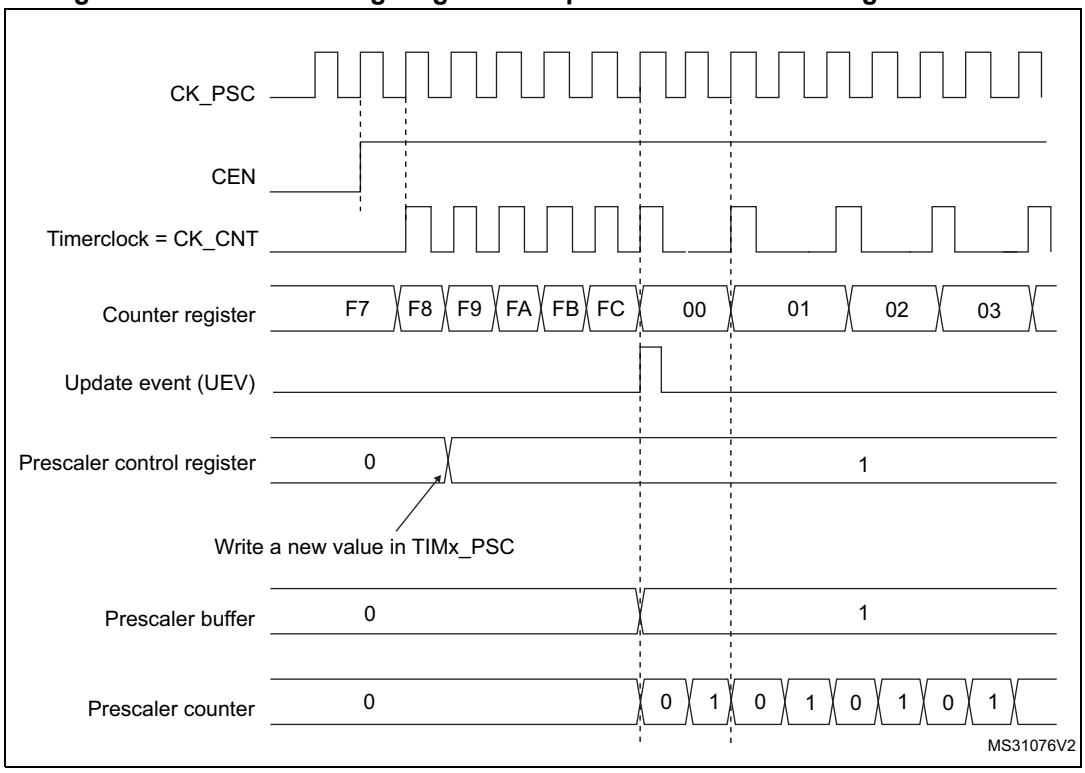
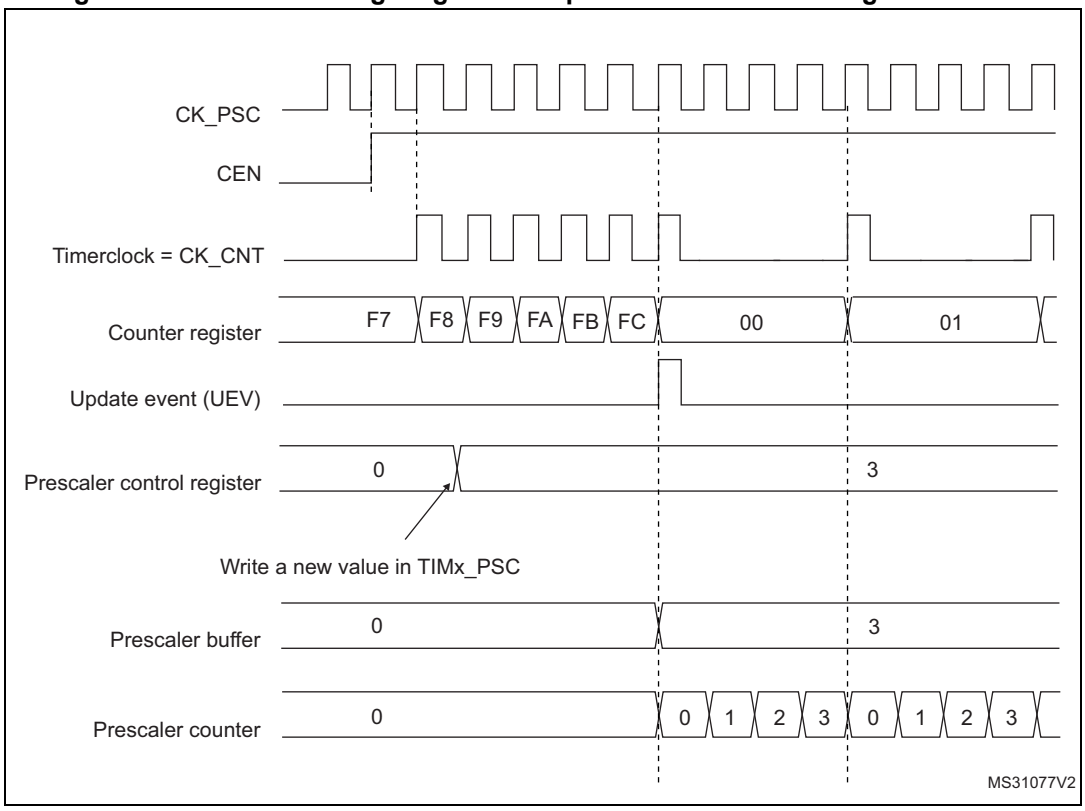



**Figure 117. Counter timing diagram with prescaler division change from 1 to 4**



## 19.3.2 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 118. Counter timing diagram, internal clock divided by 1**

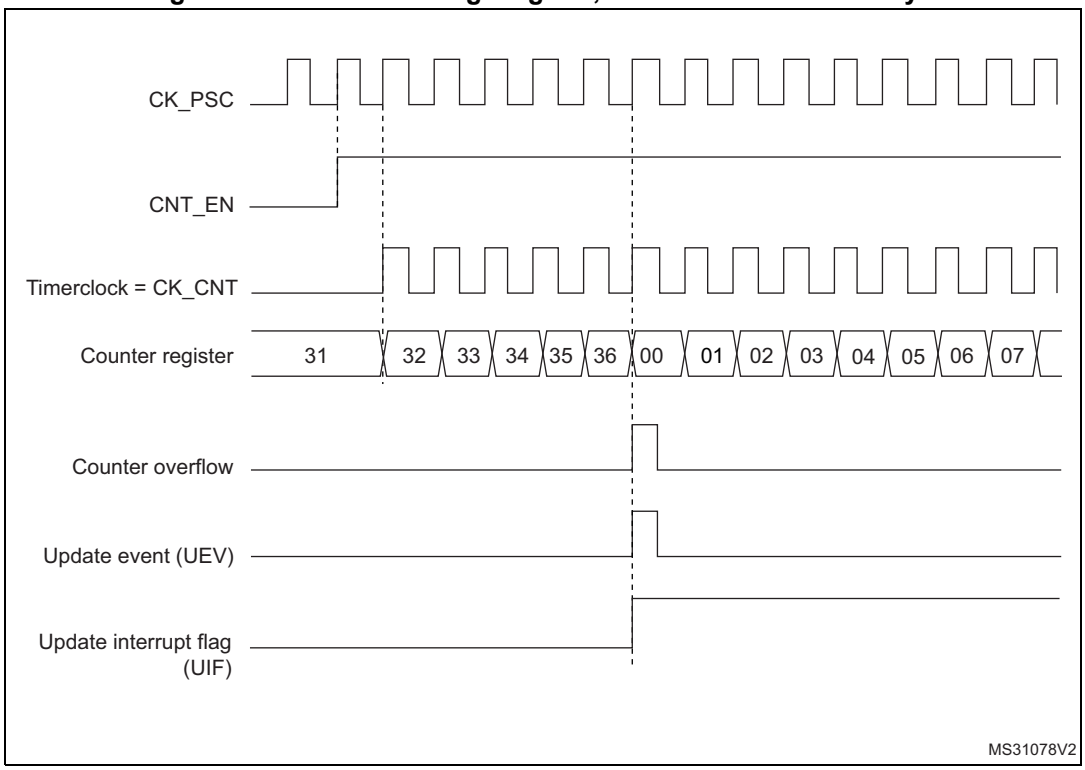



**Figure 119. Counter timing diagram, internal clock divided by 2**

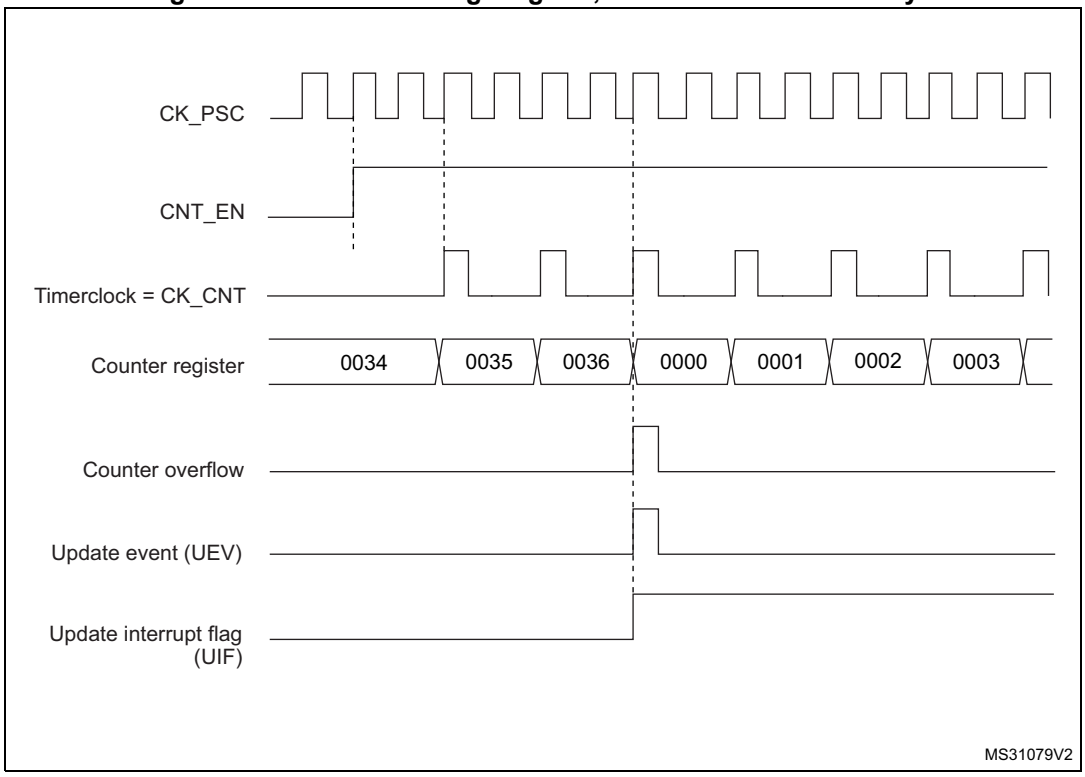


**Figure 120. Counter timing diagram, internal clock divided by 4**

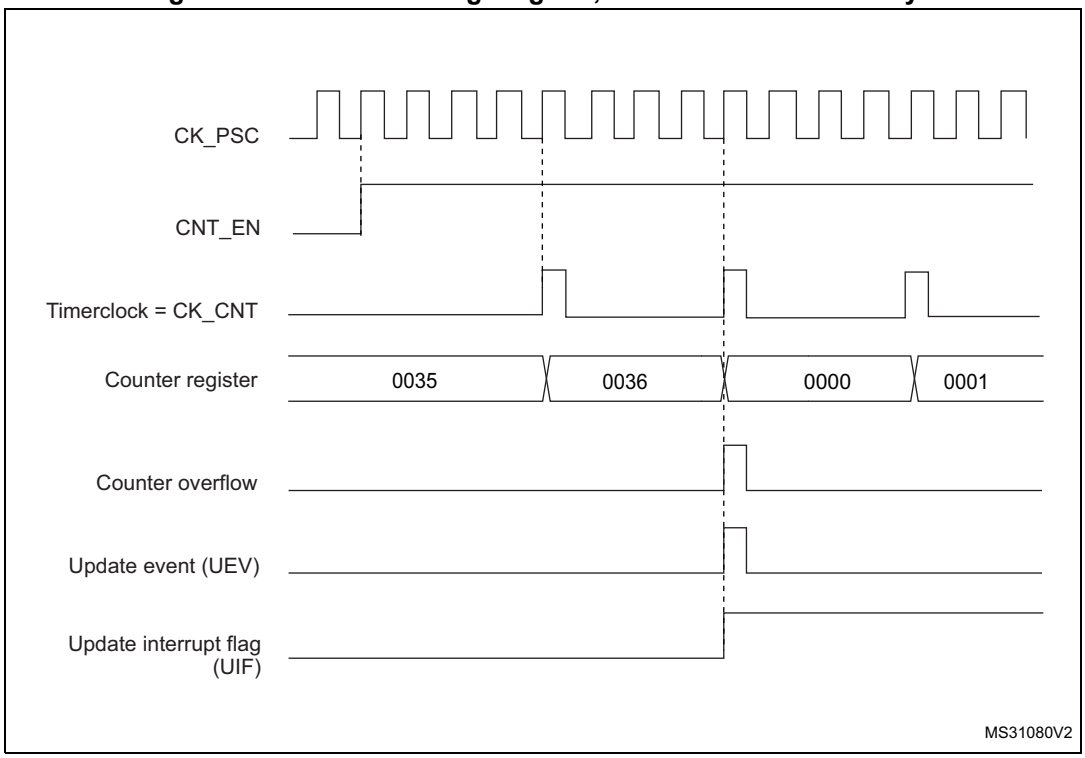



**Figure 121. Counter timing diagram, internal clock divided by N**

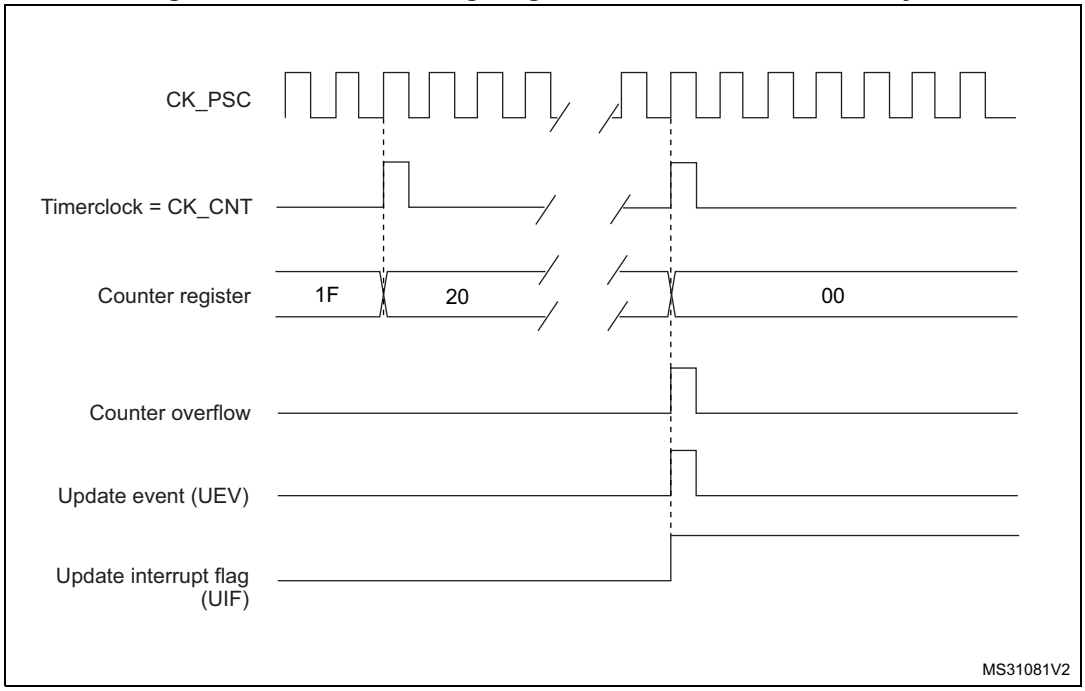


**Figure 122. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)**

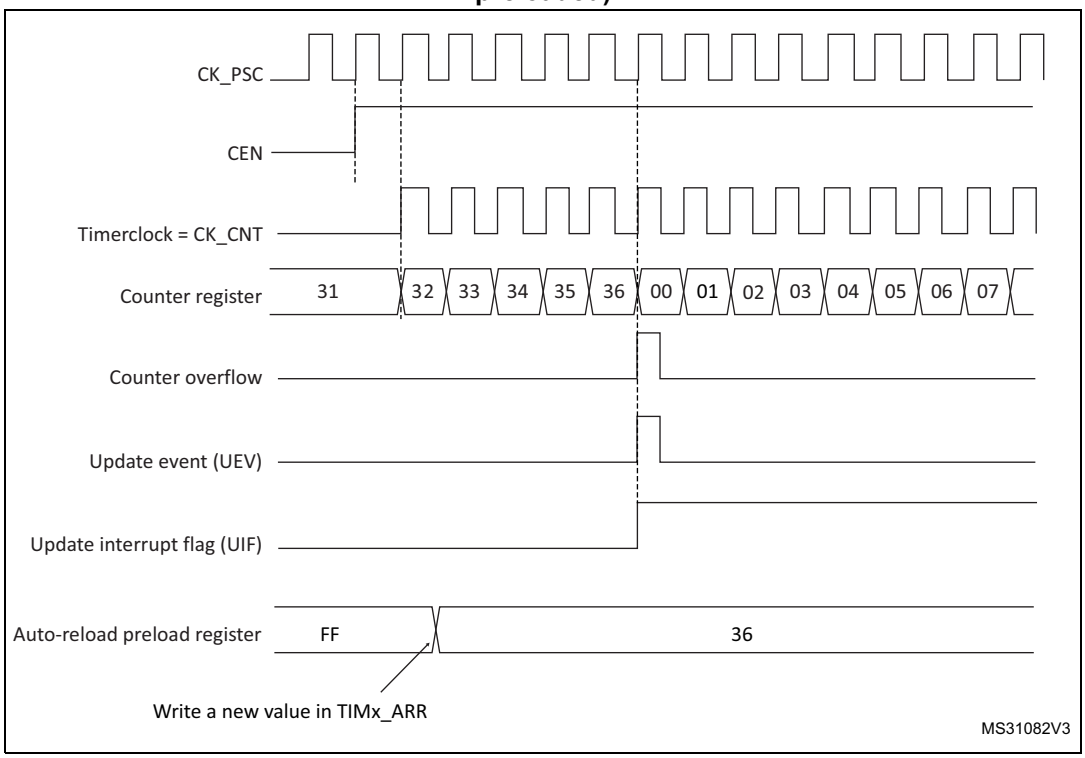
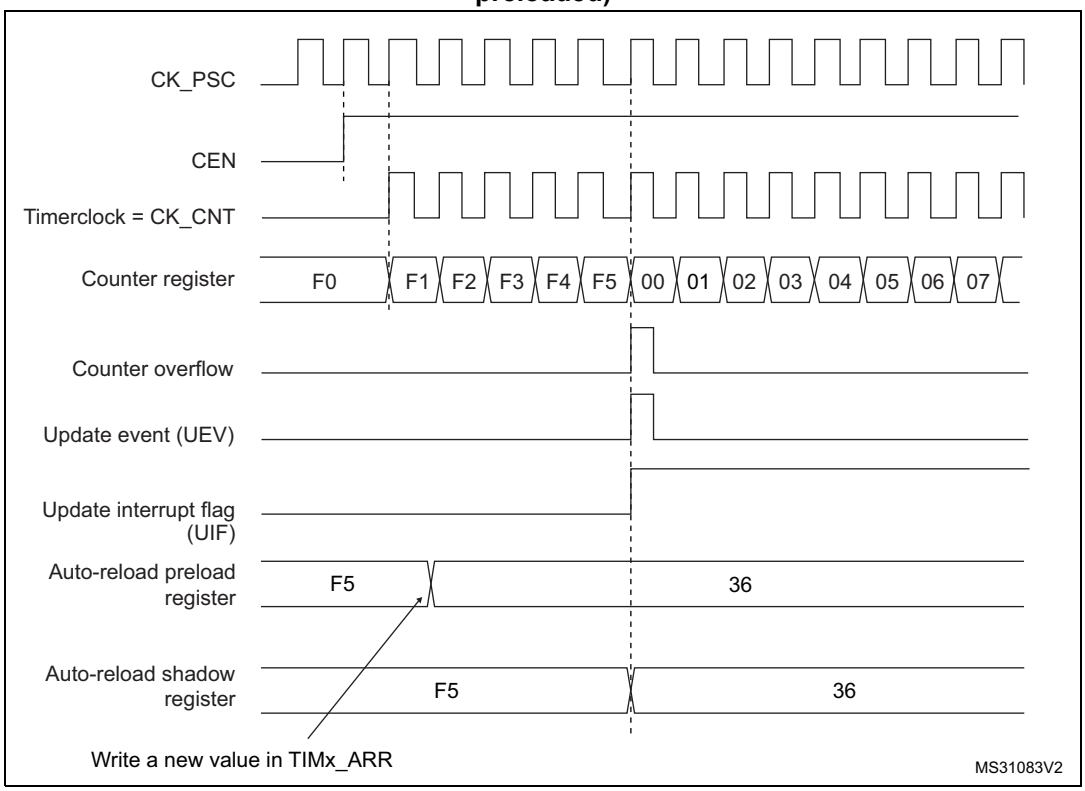



**Figure 123. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)**



### 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 124. Counter timing diagram, internal clock divided by 1**

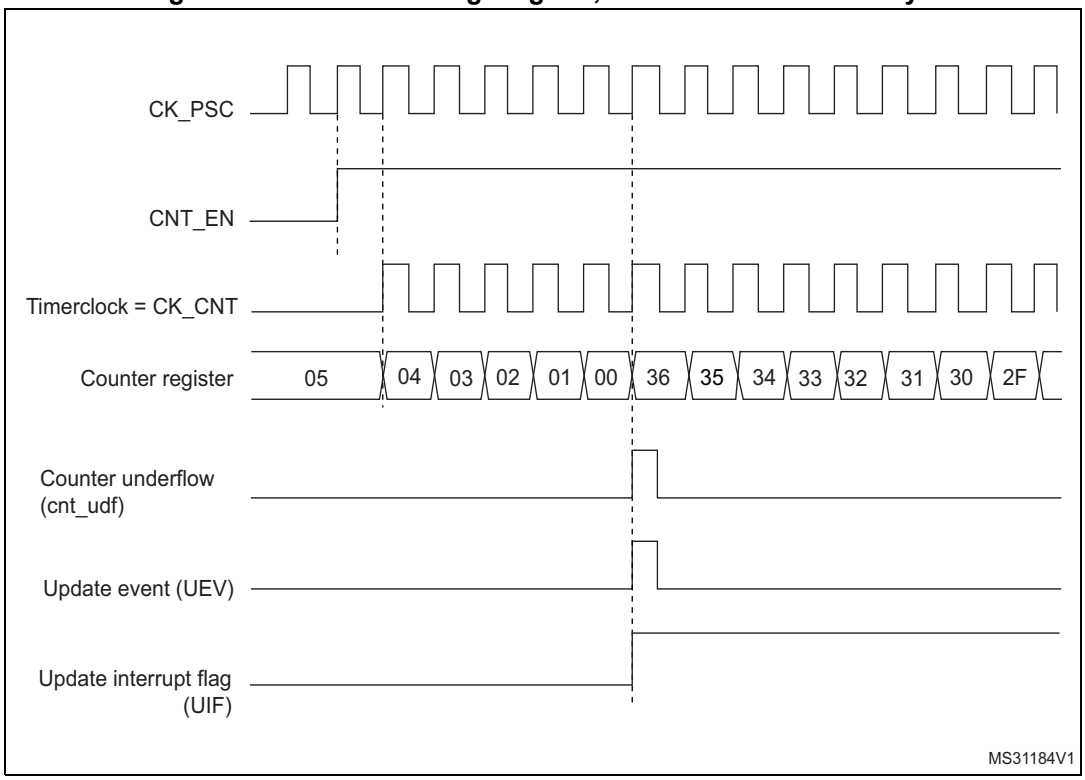



**Figure 125. Counter timing diagram, internal clock divided by 2**

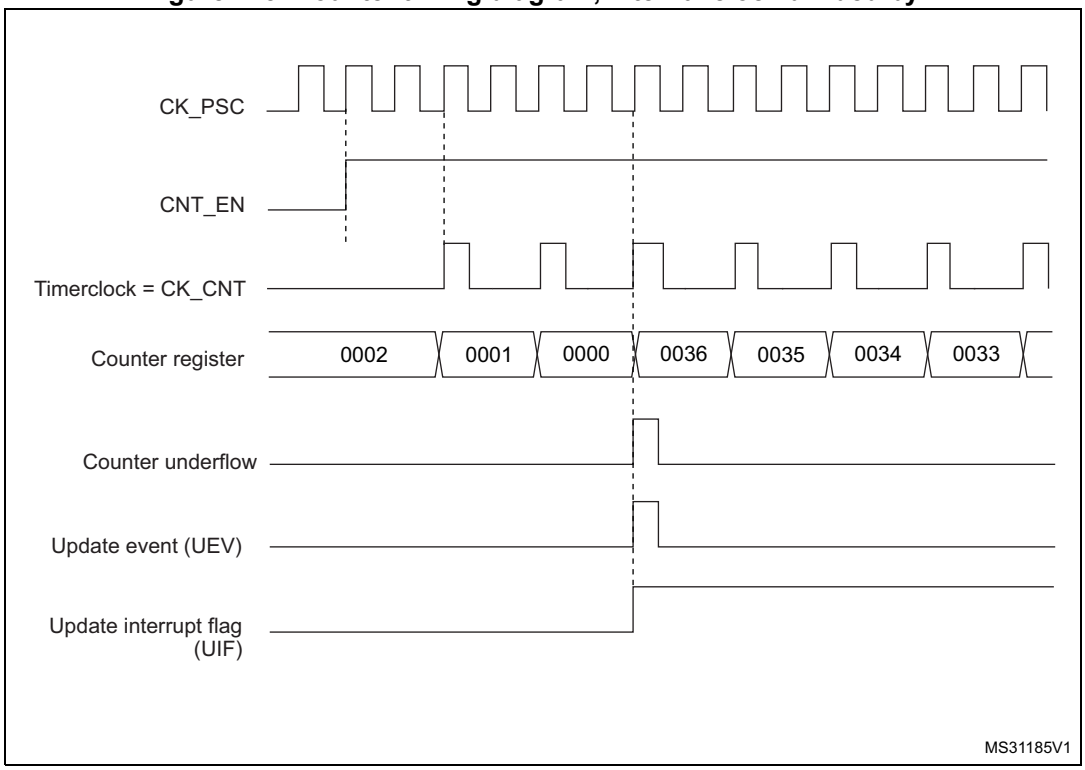


**Figure 126. Counter timing diagram, internal clock divided by 4**

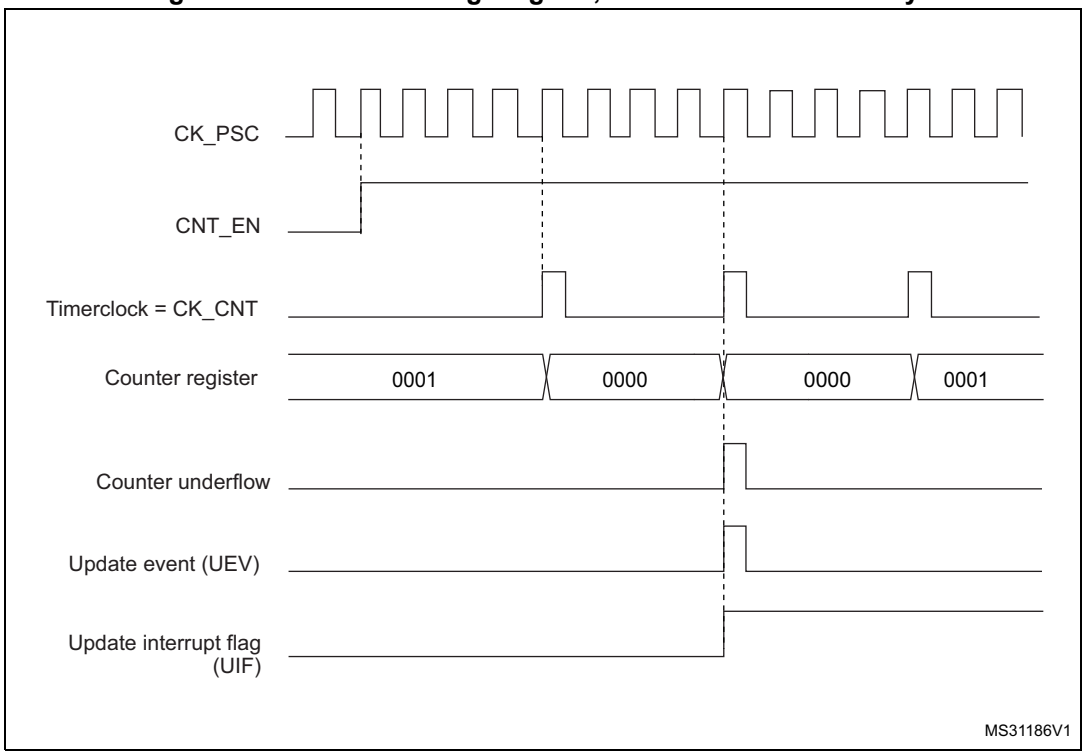



**Figure 127. Counter timing diagram, internal clock divided by N**

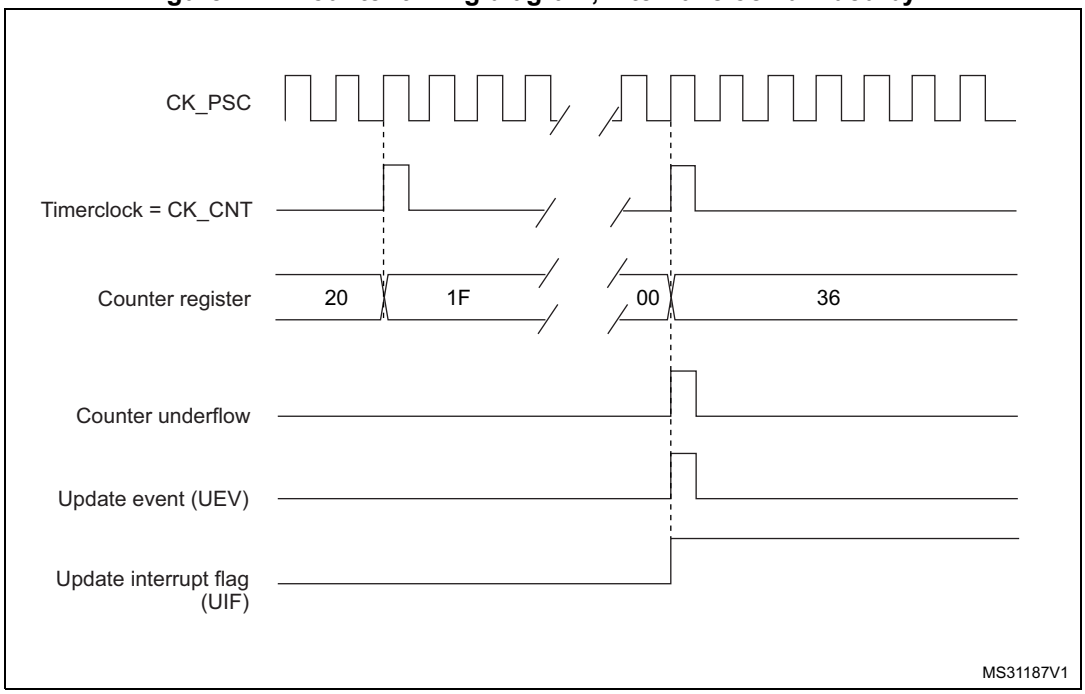
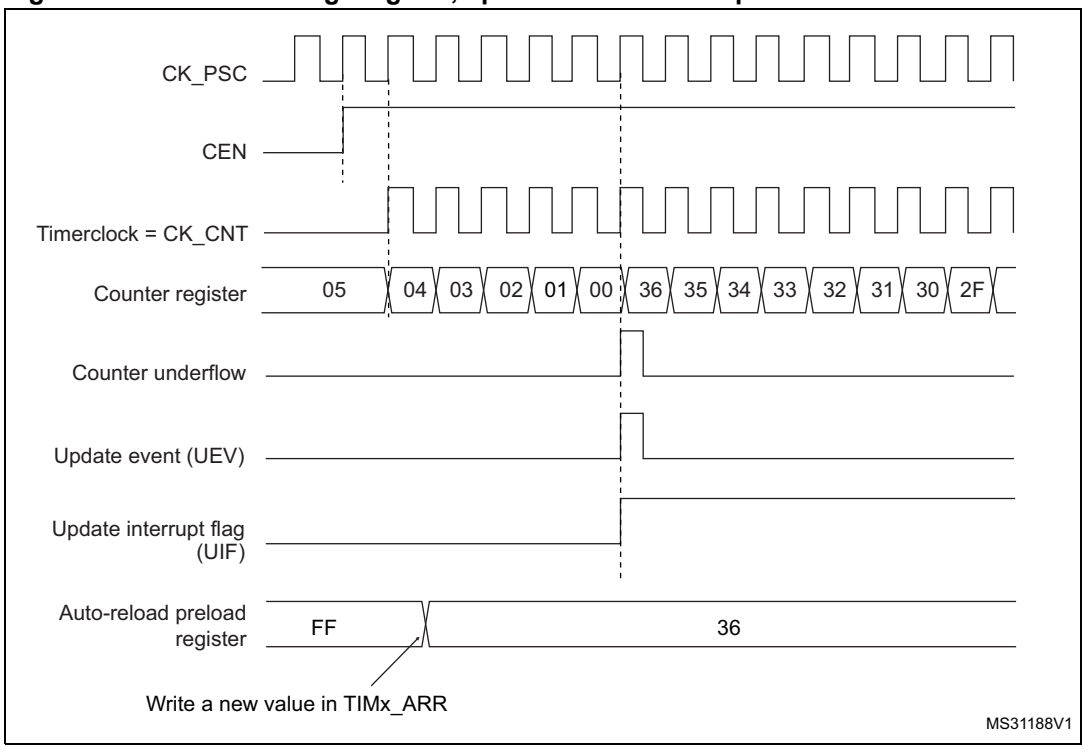


**Figure 128. Counter timing diagram, update event when repetition counter is not used**




## 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

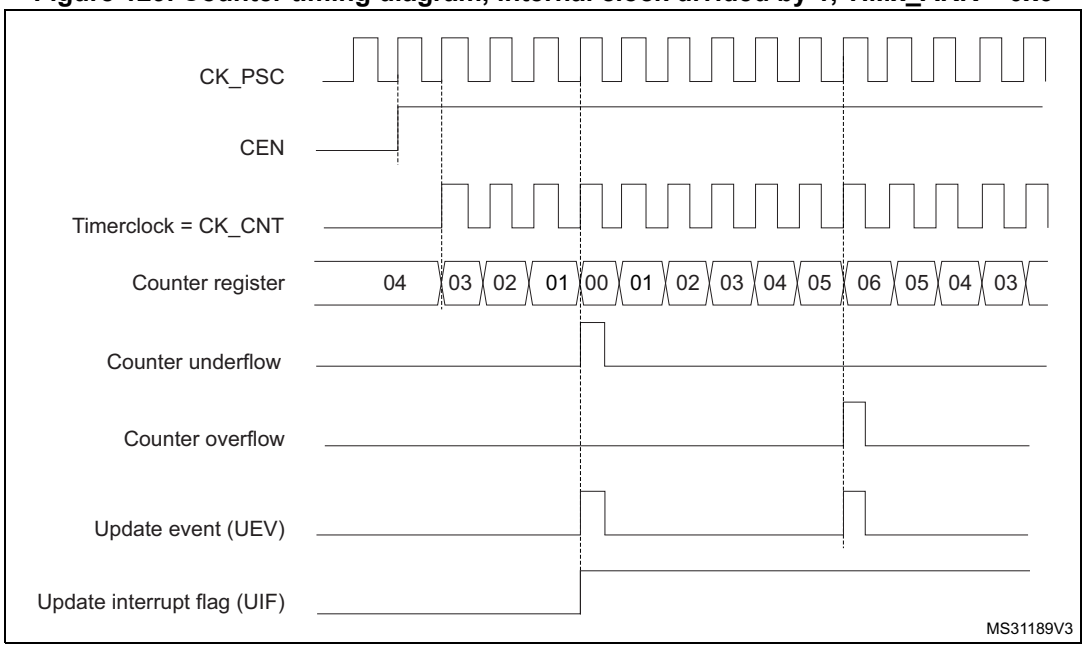
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

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 129. 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 19.4: TIM1 registers*).

**Figure 130. Counter timing diagram, internal clock divided by 2**

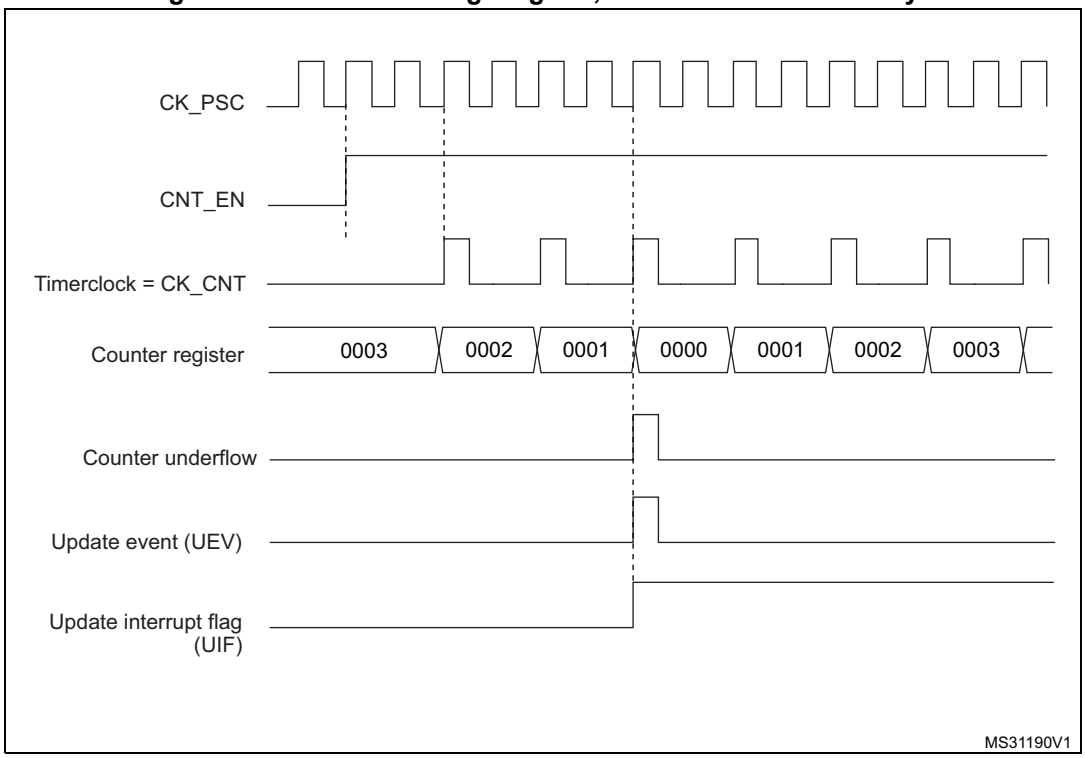



**Figure 131. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36**

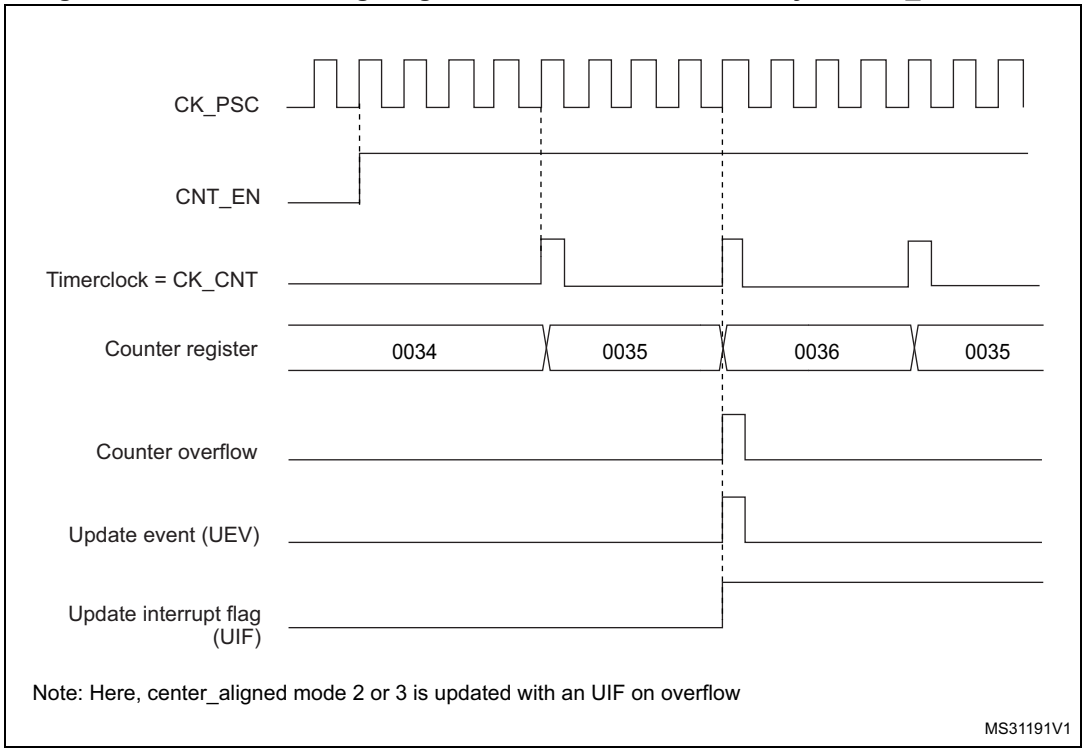


**Figure 132. Counter timing diagram, internal clock divided by N**

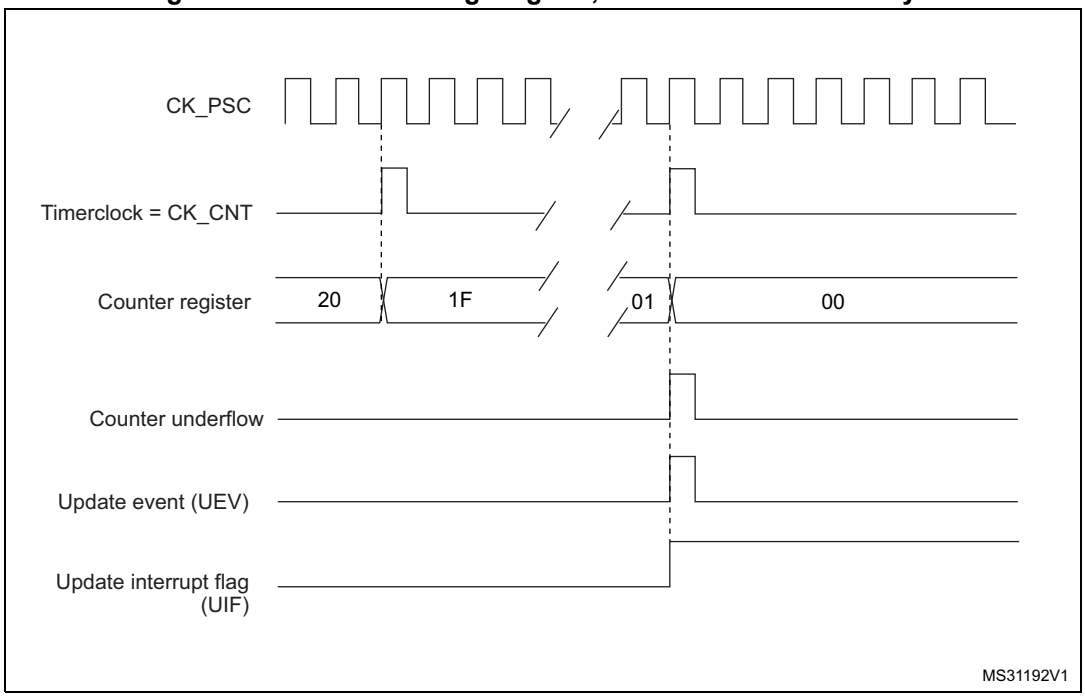



**Figure 133. Counter timing diagram, update event with ARPE=1 (counter underflow)**

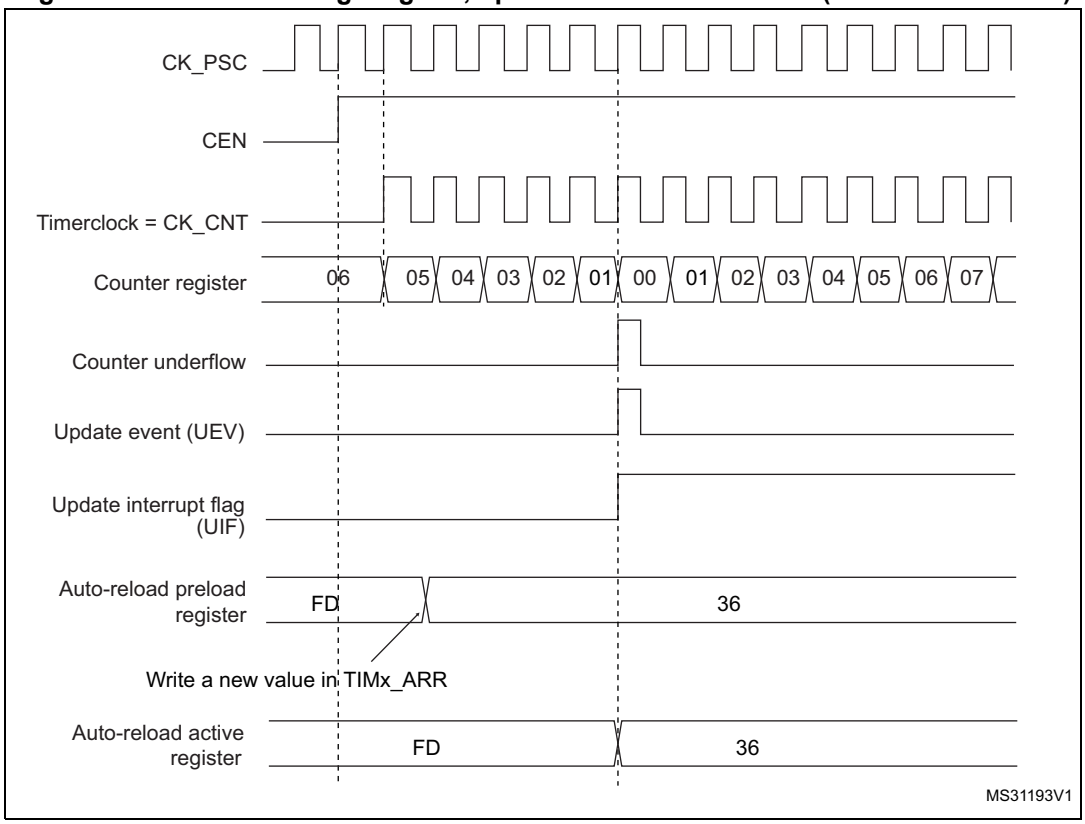
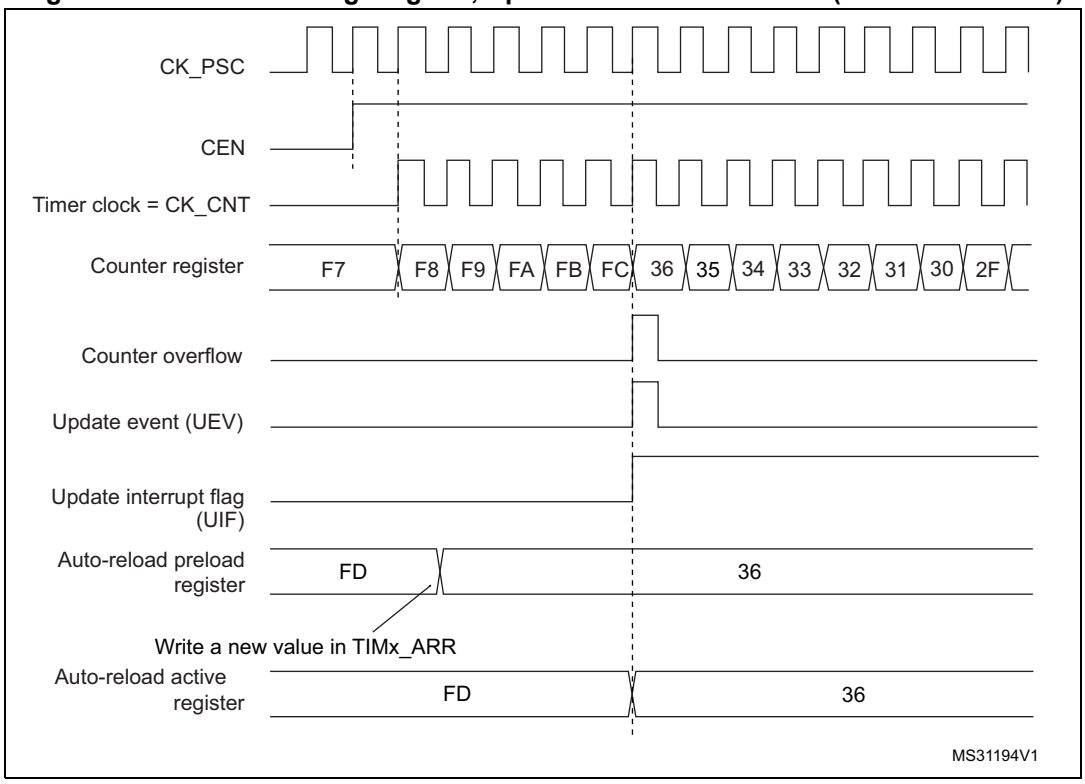


**Figure 134. Counter timing diagram, Update event with ARPE=1 (counter overflow)**




## 19.3.3 Repetition counter

*Section 19.3.1: 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:

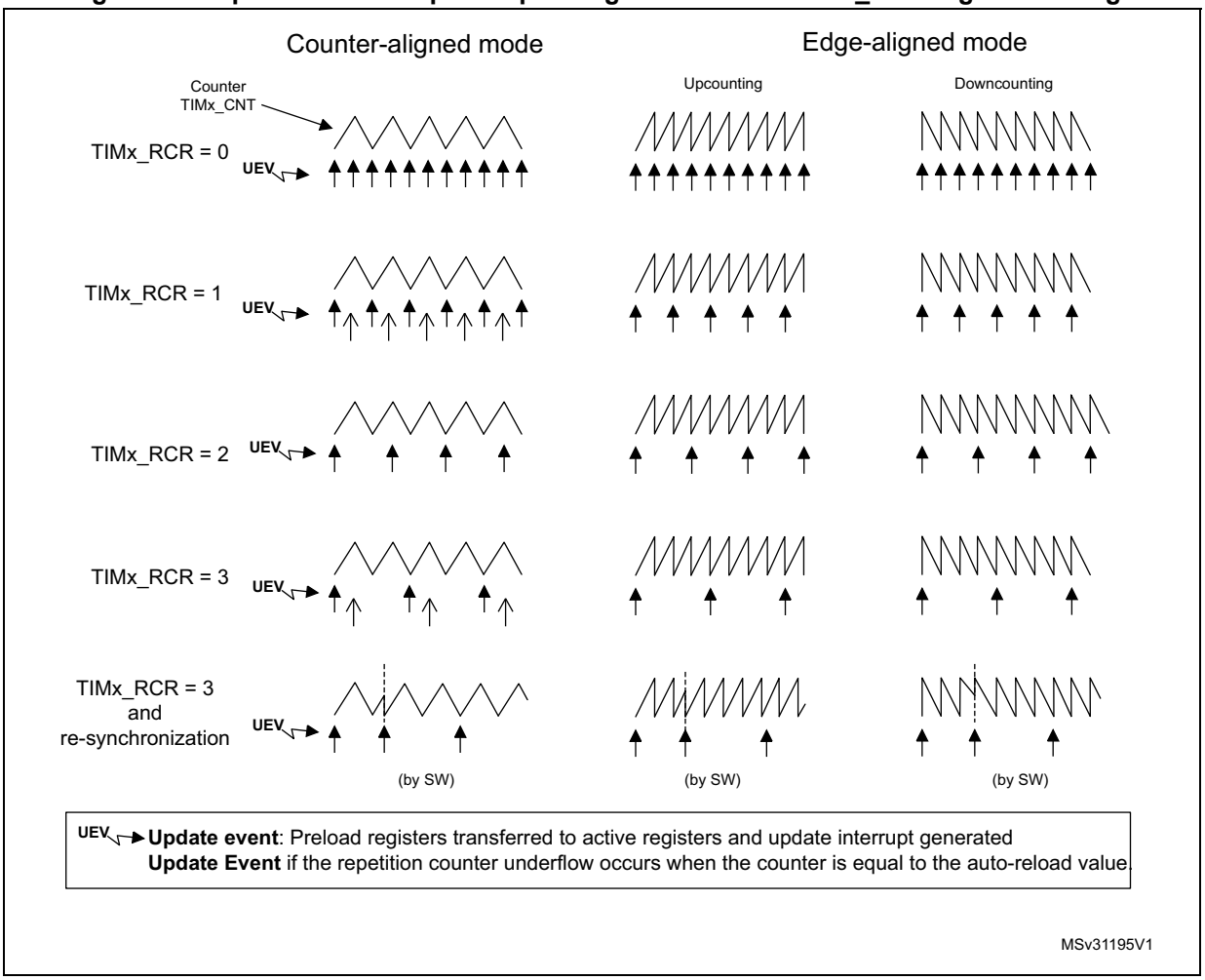
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice 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 135*). 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 underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

**Figure 135. Update rate examples depending on mode and TIMx_RCR register settings**



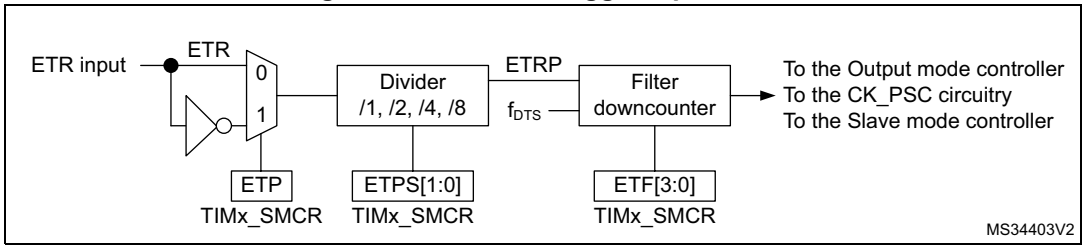
### 19.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see *Section 19.3.5*)
- trigger for the slave mode (see *Section 19.3.26*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 19.3.7*)

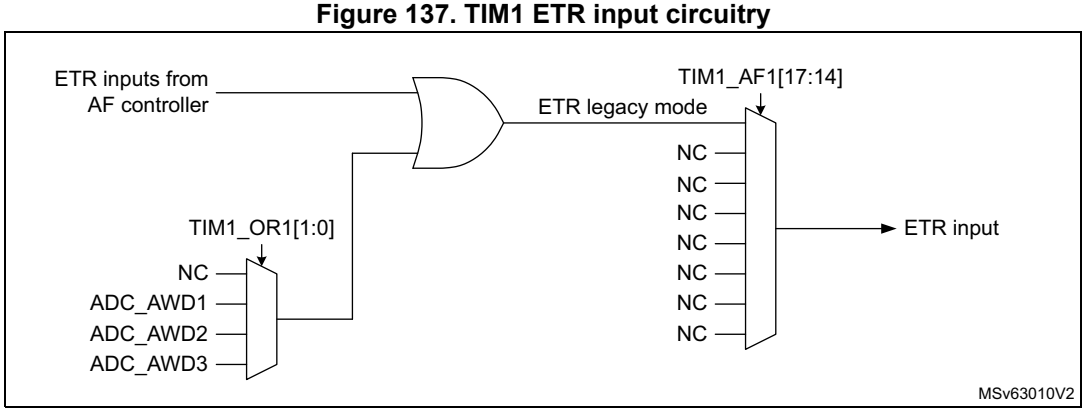
*Figure 136* below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

**Figure 136. External trigger input block**




The ETR input comes from multiple sources: input pins (default configuration) and analog watchdogs. The selection is done with the ETRSEL[3:0] and the TIM1_OR1[1:0] bitfields.

**Figure 137. TIM1 ETR input circuitry**



### 19.3.5 Clock selection

The counter clock can be provided by the following clock sources:

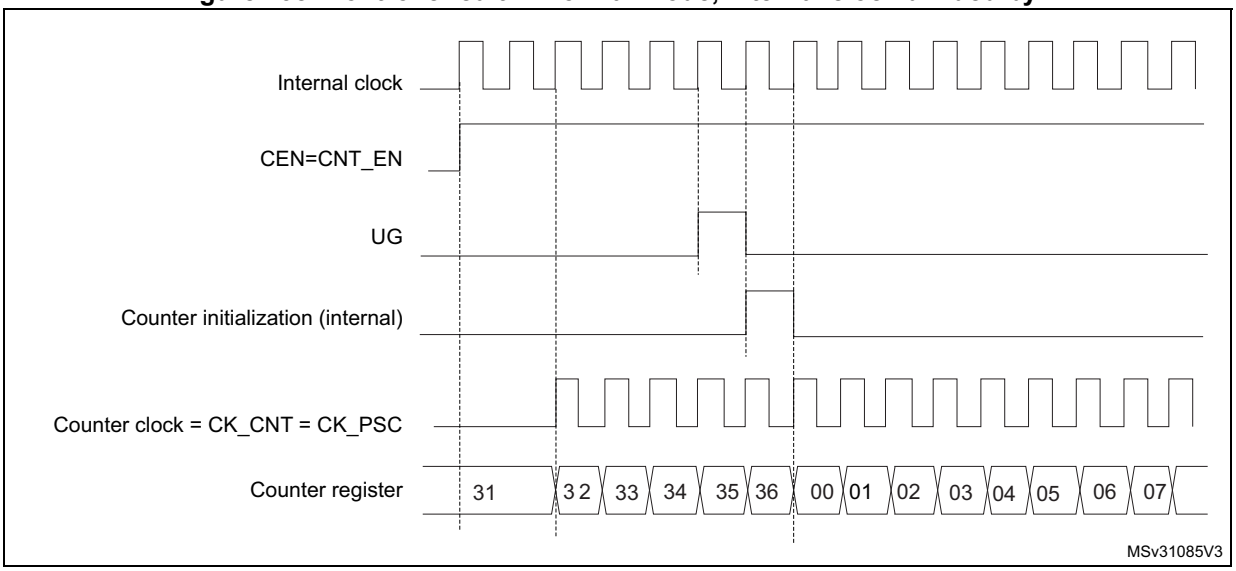
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

#### 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 138* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

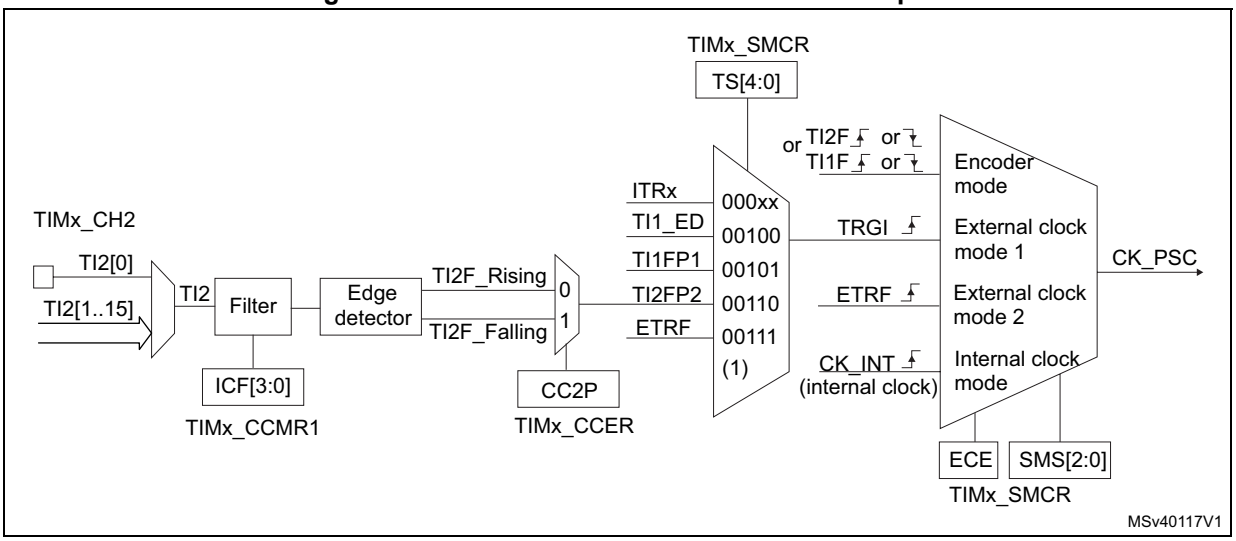
**Figure 138. Control circuit in normal mode, internal clock divided by 1**




#### 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 139. TI2 external clock connection example**




1. Codes ranging from 01000 to 11111 are reserved

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

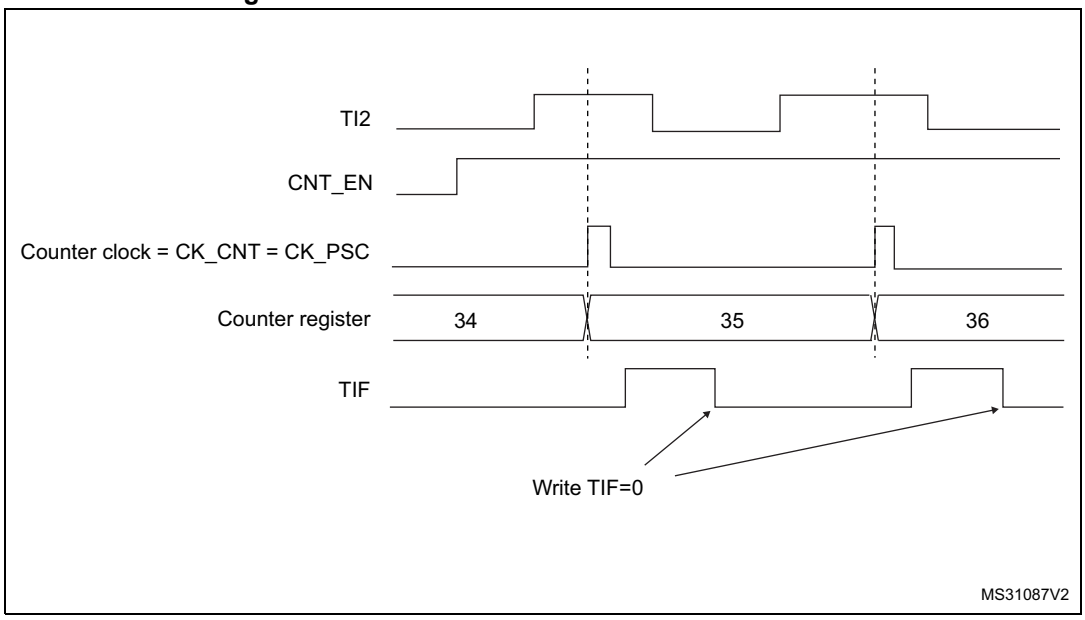
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 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=00110 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 the user does not 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 140. Control circuit in external clock mode 1**



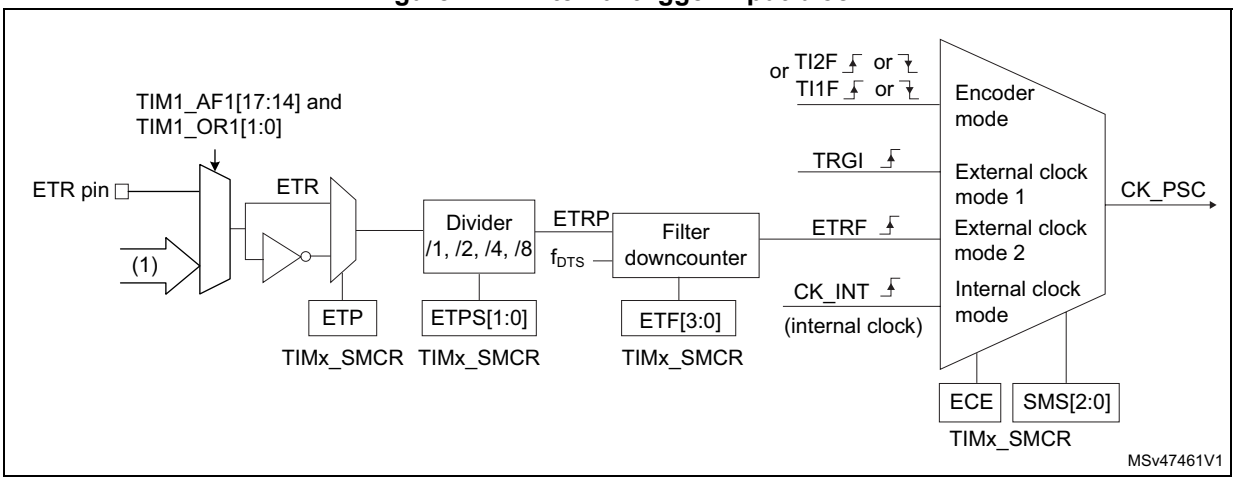

## External clock source mode 2

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 141* gives an overview of the external trigger input block.

**Figure 141. External trigger input block**




1. Refer to *Figure 137: TIM1 ETR input circuitry*.

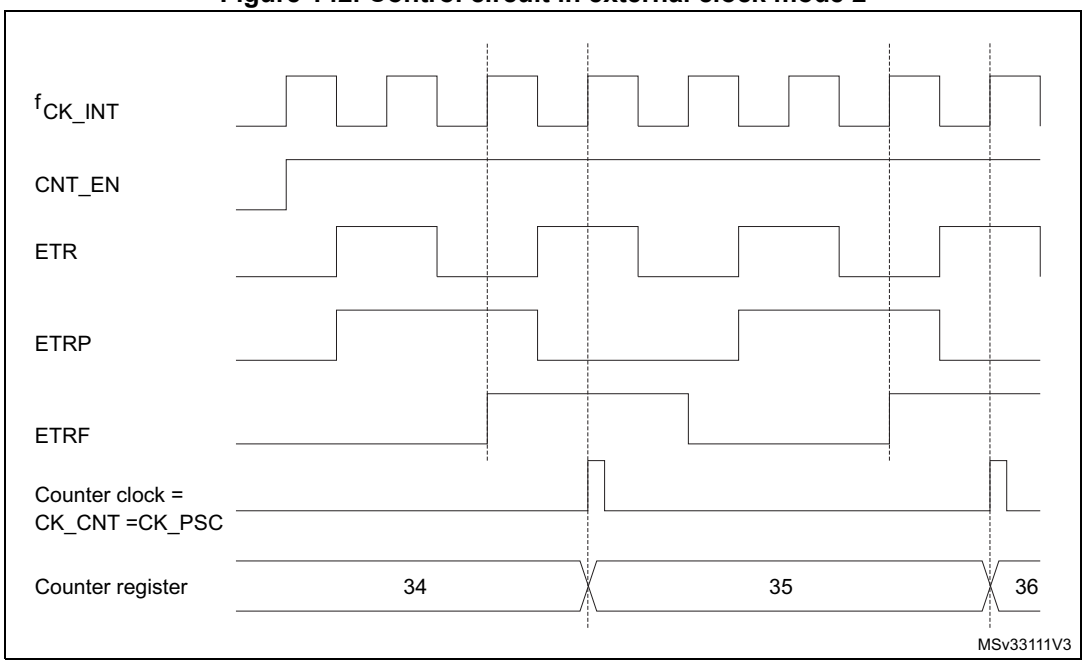
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. 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. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by proper ETPS prescaler setting.

**Figure 142. Control circuit in external clock mode 2**



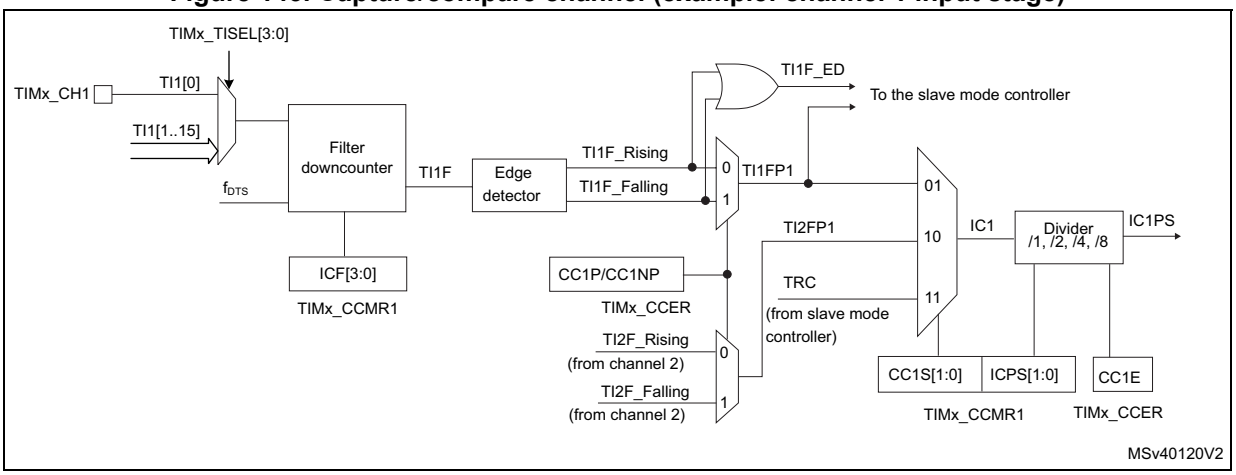
### 19.3.6 Capture/compare channels

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 143* to *Figure 146* 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 143. 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 144. Capture/compare channel 1 main circuit**

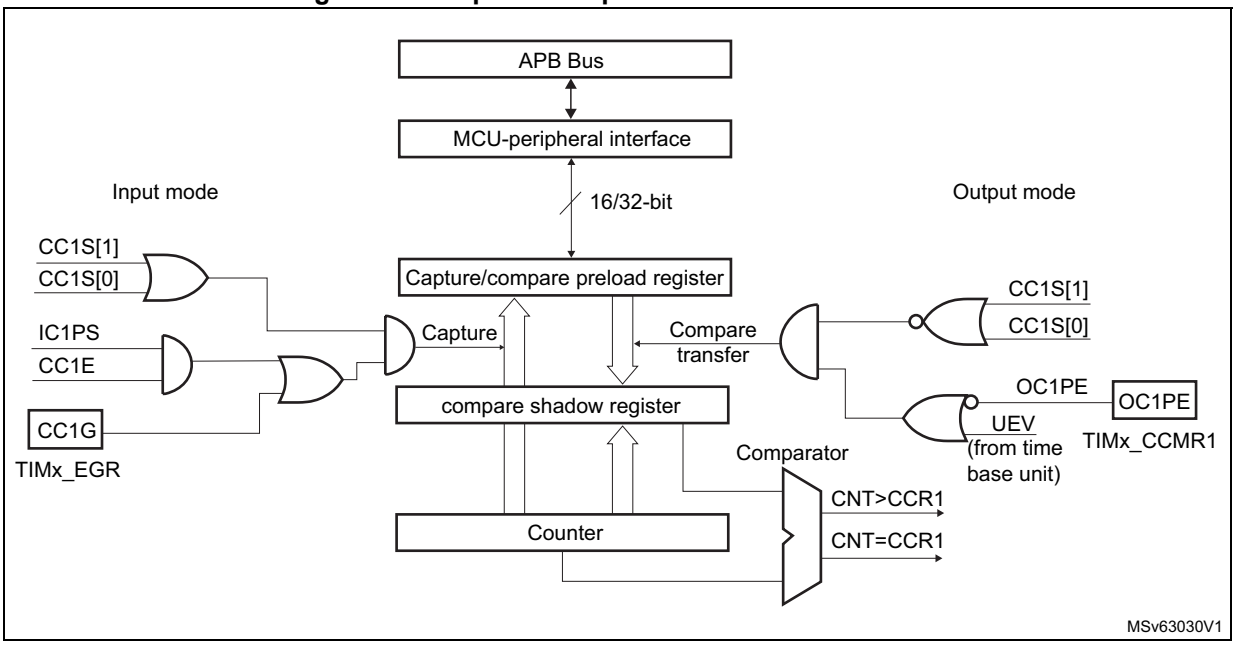
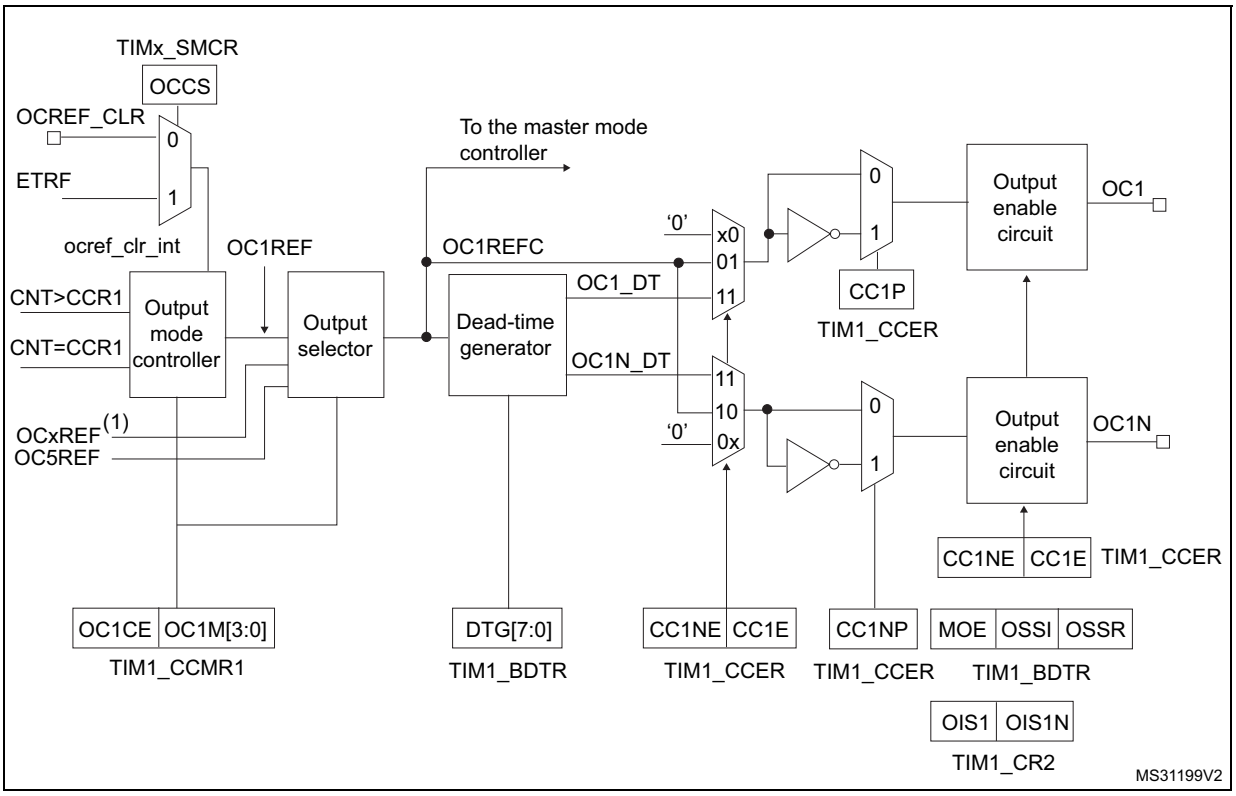


**Figure 145. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)**




1. OCxREF, where x is the rank of the complementary channel

**Figure 146. Output stage of capture/compare channel (channel 4)**

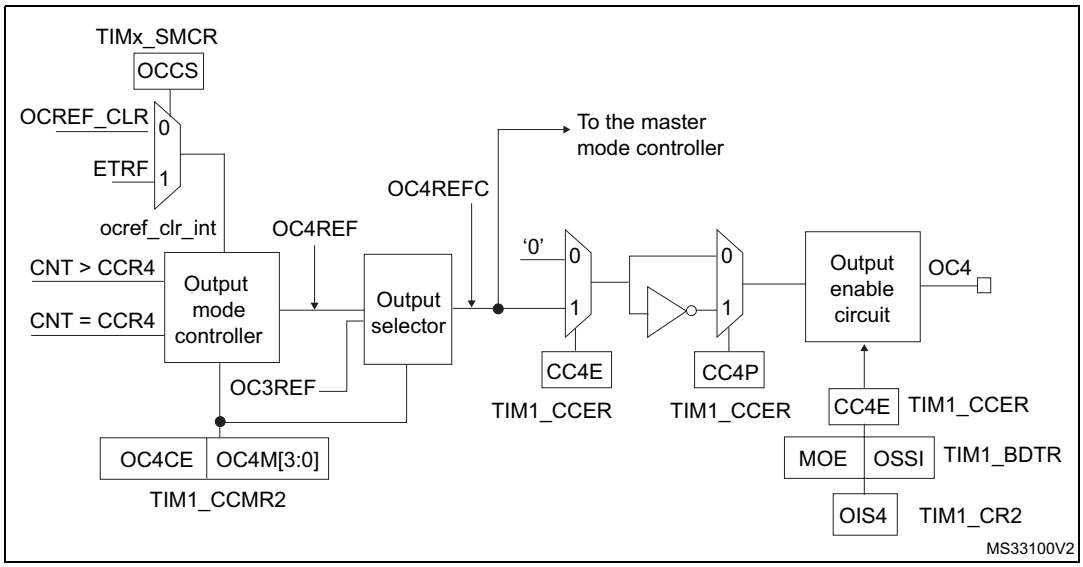
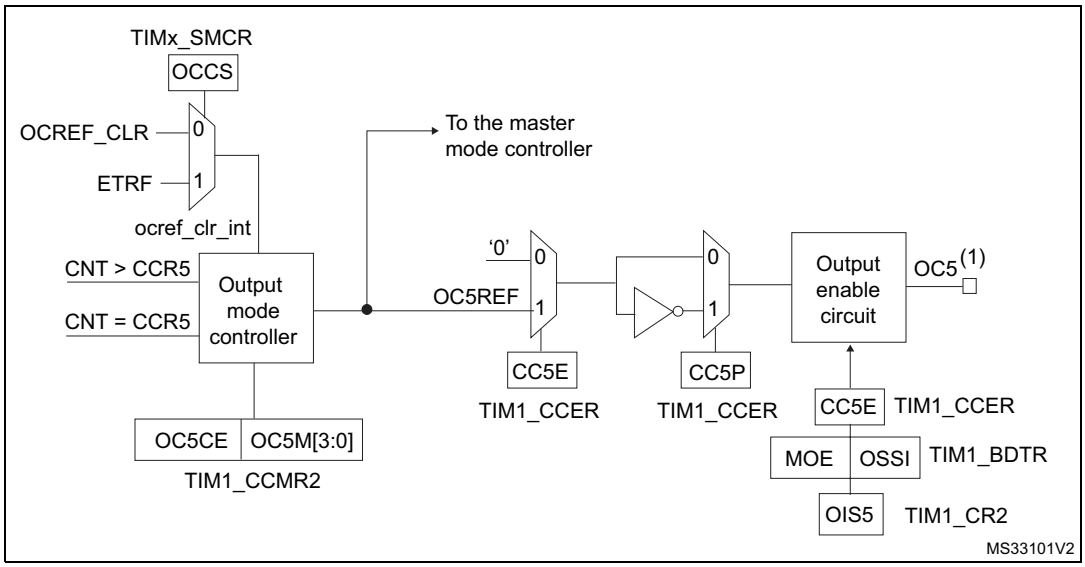


**Figure 147. Output stage of capture/compare channel (channel 5, idem ch. 6)**




1. Not available externally.

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.

### 19.3.7 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 written with '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 proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. 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.
3. Program the appropriate input filter duration in relation with the signal connected 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.

4. 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).
5. 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).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. 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.*

### 19.3.8 PWM input mode

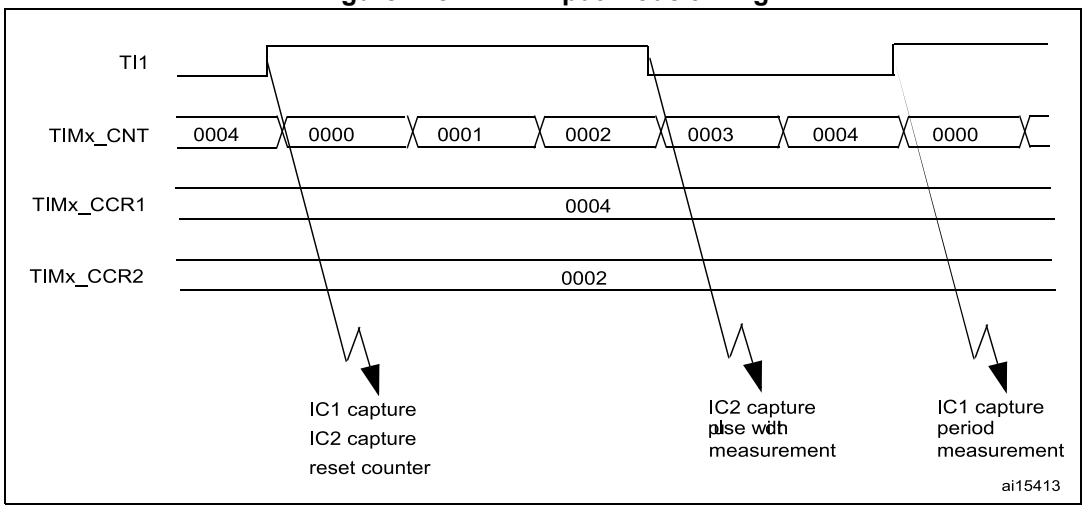
This mode is a particular case of input capture mode. The procedure is the same except:

- 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, the user 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):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. 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).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. 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).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

**Figure 148. PWM input mode timing**




## 19.3.9 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, user just needs 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.

### 19.3.10 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, while Channel 5 and 6 are only available inside the device (for instance, for compound waveform generation or for ADC triggering).

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.

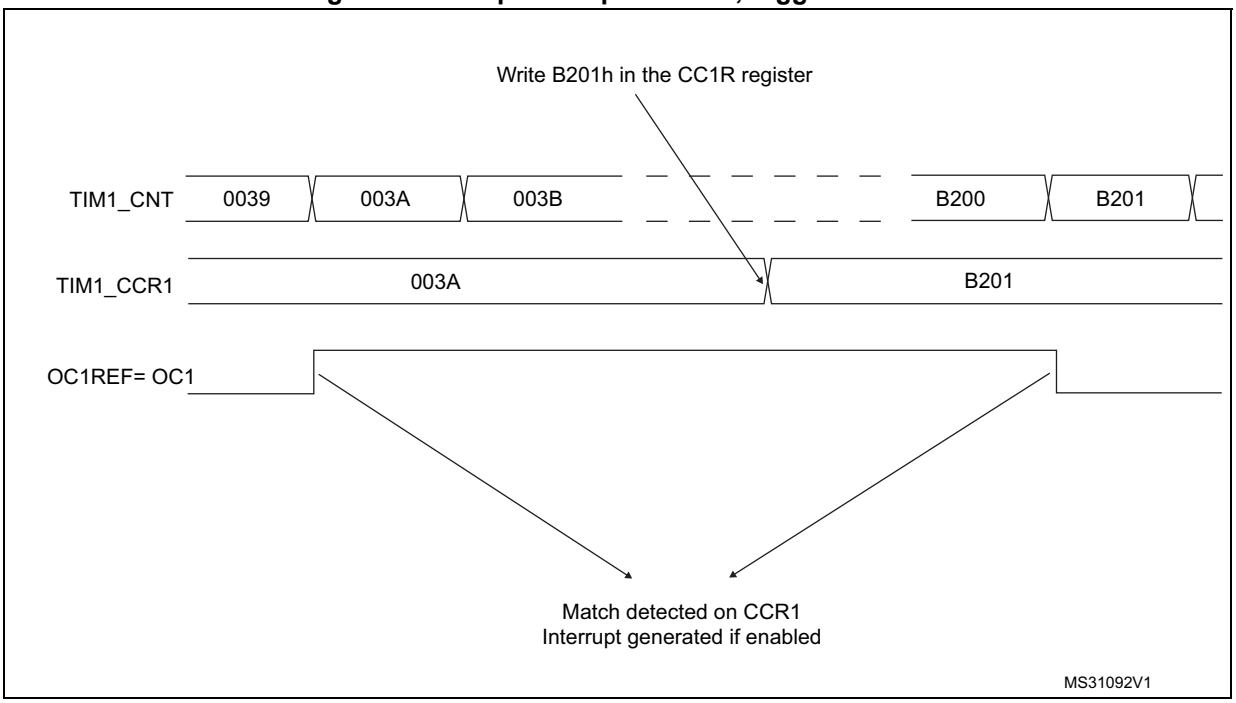
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

#### 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 149*.

**Figure 149. Output compare mode, toggle on OC1**




## 19.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated 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. The corresponding preload register must be enabled 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, all registers must be initialized 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.

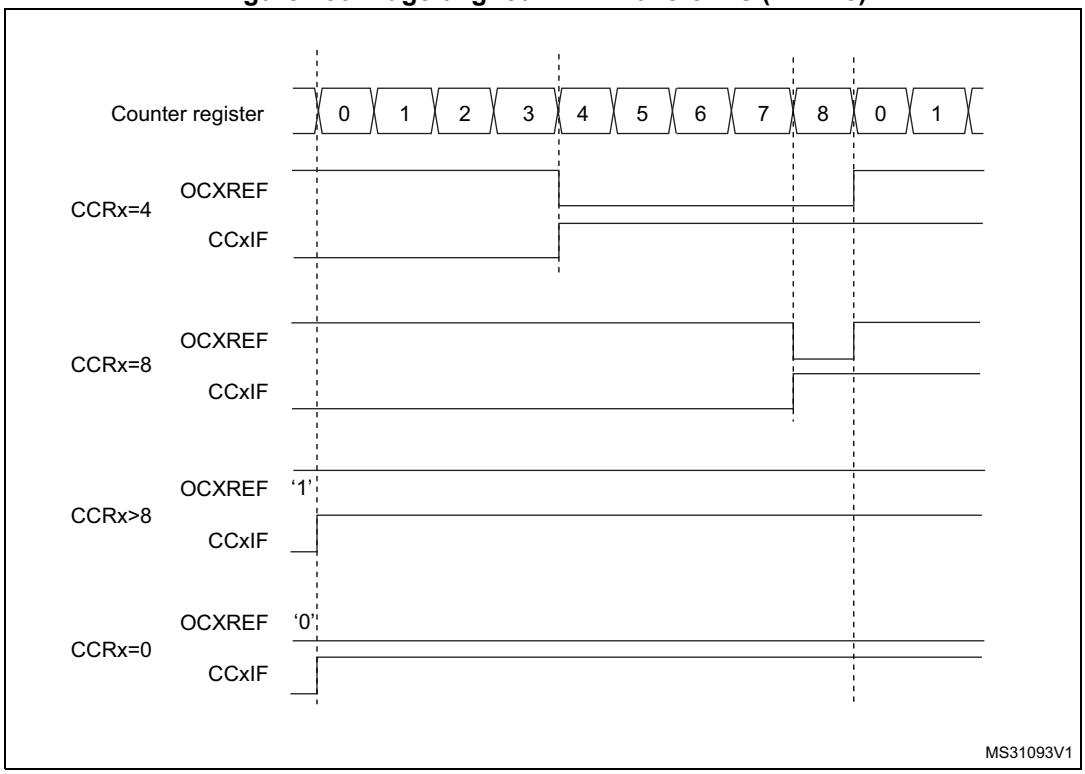
### PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 537*.

  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 150* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

**Figure 150. 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 541*

  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

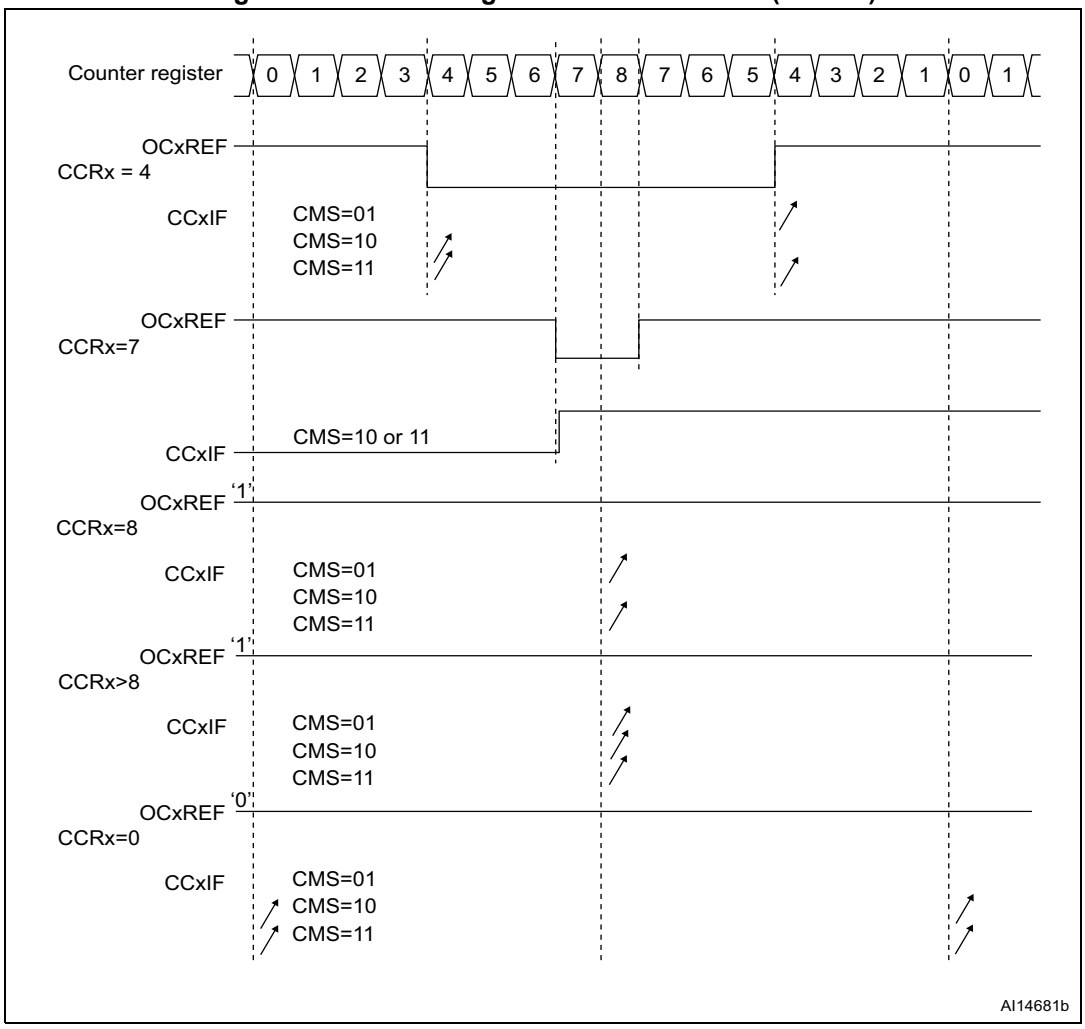
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the 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 544*.

*Figure 151* 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 151. 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 a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written 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.

### 19.3.12 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

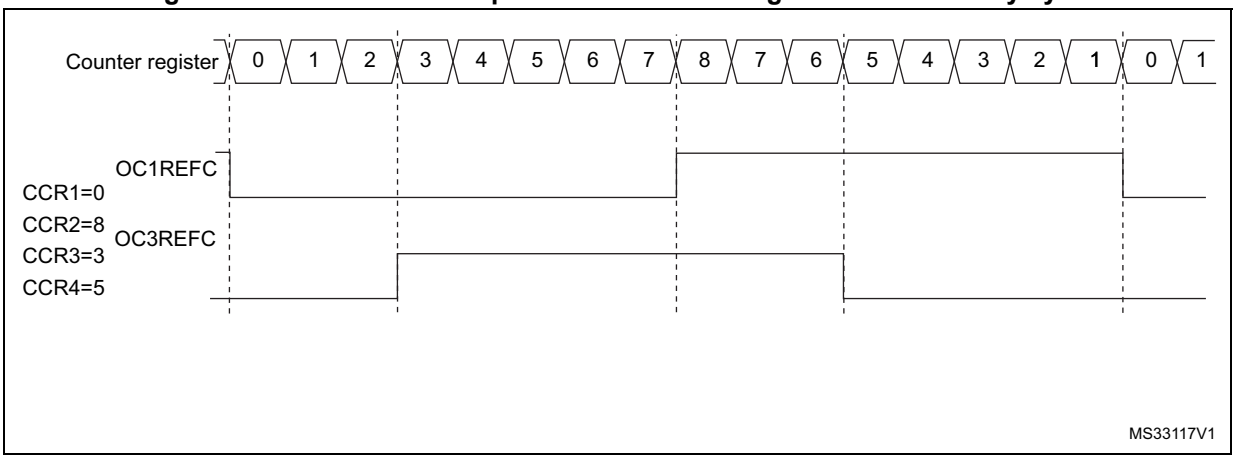
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

*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 152* 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 152. Generation of 2 phase-shifted PWM signals with 50% duty cycle**




## 19.3.13 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.

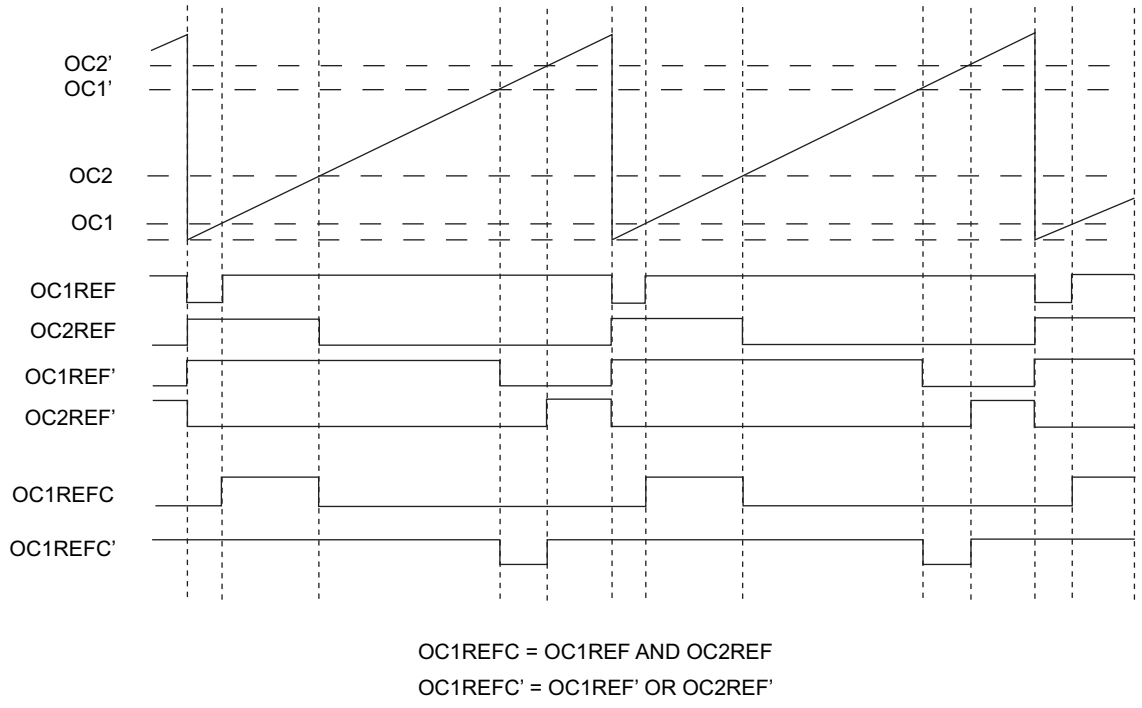
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

*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 153* 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 153. Combined PWM mode on channel 1 and 3**



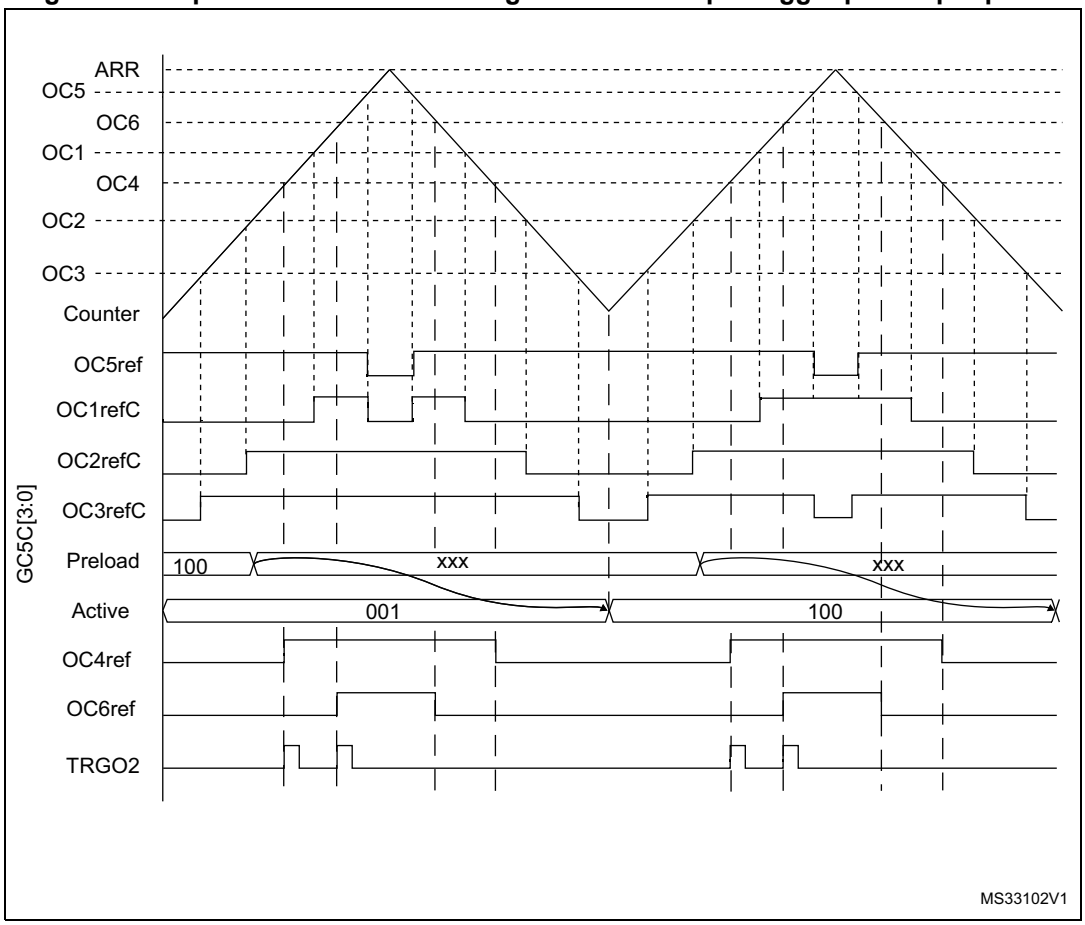

## 19.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, OC1REFC is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, OC2REFC is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, OC3REFC is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

**Figure 154. 3-phase combined PWM signals with multiple trigger pulses per period**




The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 19.3.27: ADC synchronization* for more details.

## 19.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected 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 129: Output control bits for complementary OCx and OCxN channels with break feature on page 613* 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 155. Complementary output with dead-time insertion**

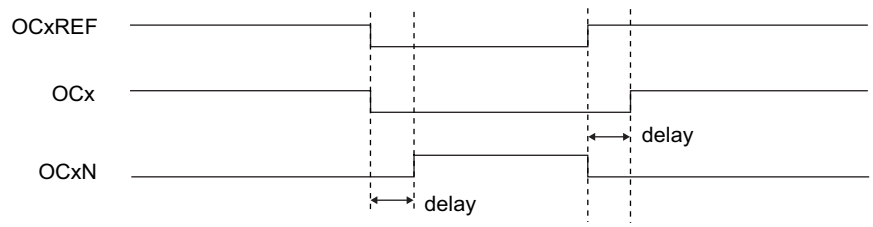



**Figure 156. Dead-time waveforms with delay greater than the negative pulse**

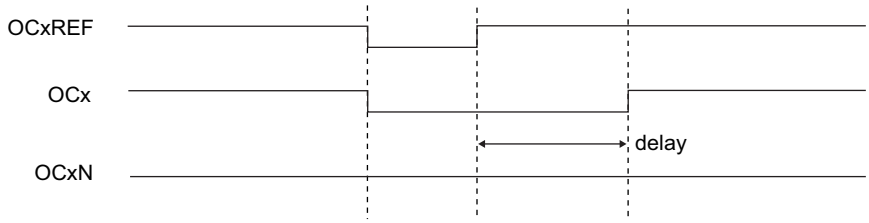
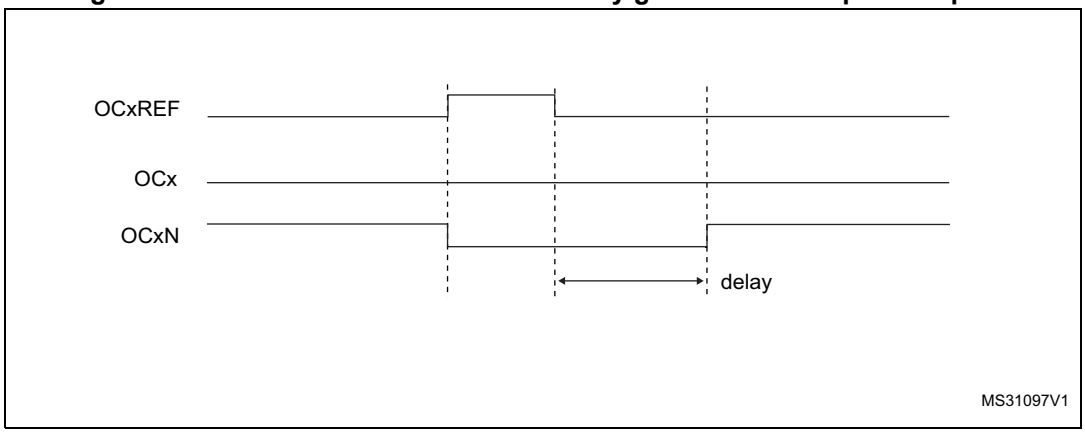


**Figure 157. Dead-time waveforms with delay greater than the positive pulse**




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 19.4.20: TIM1 break and dead-time register (TIM1_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 a specific waveform to be sent (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.*

## 19.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 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. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows the outputs to be enabled/disabled by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 129: Output control bits for complementary OCx and OCxN channels with break feature on page 613* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKE/BK2E and BKP/BK2P can be modified at the same time. When the BKE/BK2E and BKP/BK2P 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 MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 and TIMx_AF2 registers.

The sources for break (BRK) channel are:

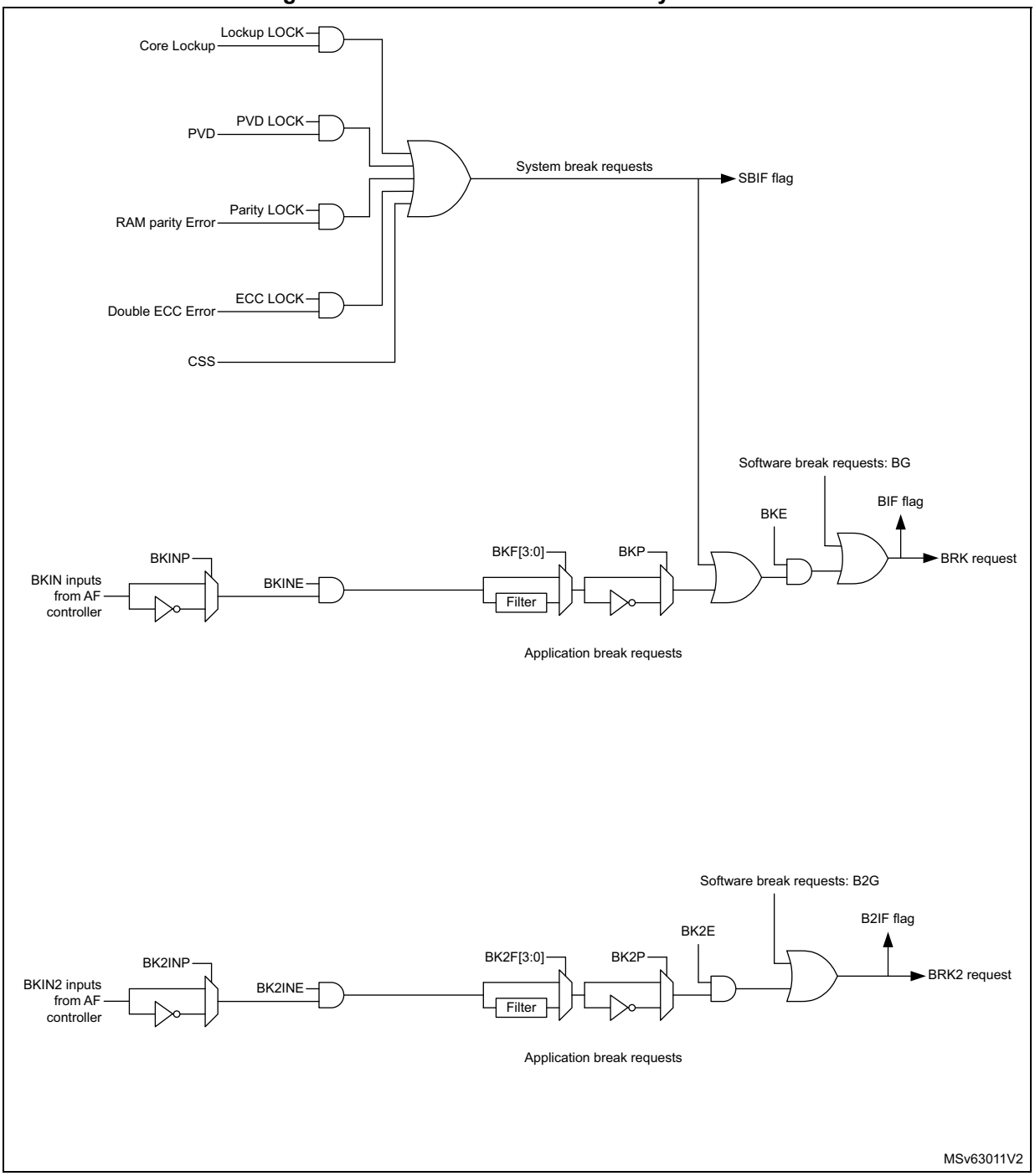
- An external source connected to one of the BKIN pin (as per selection done in the SYSCFG_CFGR2 register), with polarity selection and optional digital filtering
- An internal source:
  - the CPU1 Cortex®-M4 LOCKUP output
  - the PVD output
  - the SRAM parity error signal
  - a flash memory ECC double error detection
  - a clock failure event generated by the CSS detector

The sources for break2 (BRK2) is an external source connected to one of the BKIN pin (as per selection done in the SYSCFG_CFGR2 register), with polarity selection and optional digital filtering

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.

All sources are ORed before entering the timer BRK or BRK2 inputs, as per *Figure 158* below.

**Figure 158. Break and Break2 circuitry overview**




*Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.*

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled 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 GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in 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 slightly longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE 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). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to ‘1’ again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

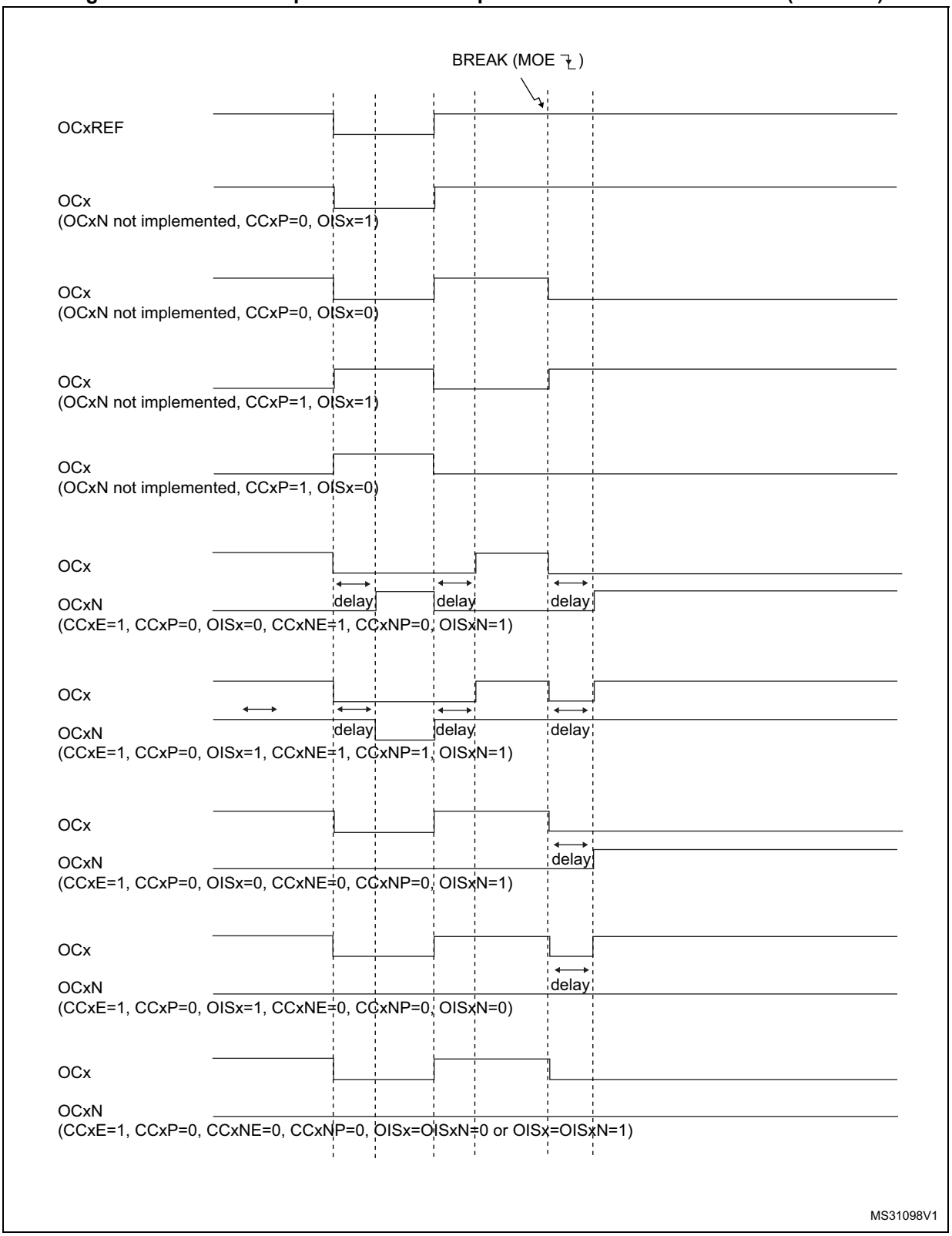
*Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value.*
*If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.*

*Note: The break inputs are active 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 and B2IF cannot be cleared.*

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 the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 19.4.20: TIM1 break and dead-time register (TIM1_BDTR)*. The LOCK bits can be written only once after an MCU reset.

*Figure 159* shows an example of behavior of the outputs in response to a break.

**Figure 159. Various output behavior in response to a break event on BRK (OSSI = 1)**



The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in *Table 125*.

*Note: BRK2 must only be used with OSSR = OSSI = 1.*

**Table 125. Behavior of timer outputs versus BRK/BRK2 inputs**

| BRK | BRK2 | Timer outputs state | Typical use case | |
|---|---|---|---|---|
| | | | OCxN output (low side switches) | OCx output (high side switches) |
| Active | X | – Inactive then forced output state (after a deadtime)<br>– Outputs disabled if OSSI = 0 (control taken over by GPIO logic) | ON after deadtime insertion | OFF |
| Inactive | Active | Inactive | OFF | OFF |

*Figure 160* gives an example of OCx and OCxN output behavior in case of active signals on BRK and BRK2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

**Figure 160. PWM output state following BRK and BRK2 pins assertion (OSSI=1)**

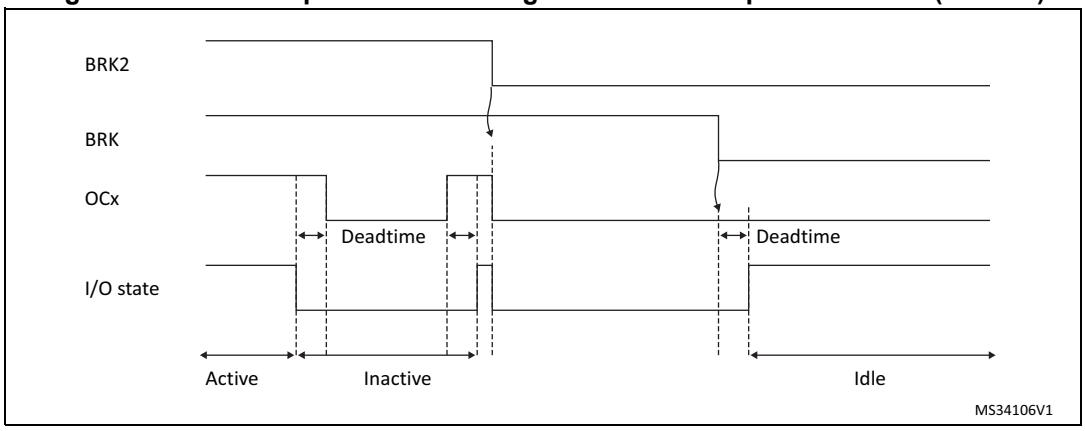
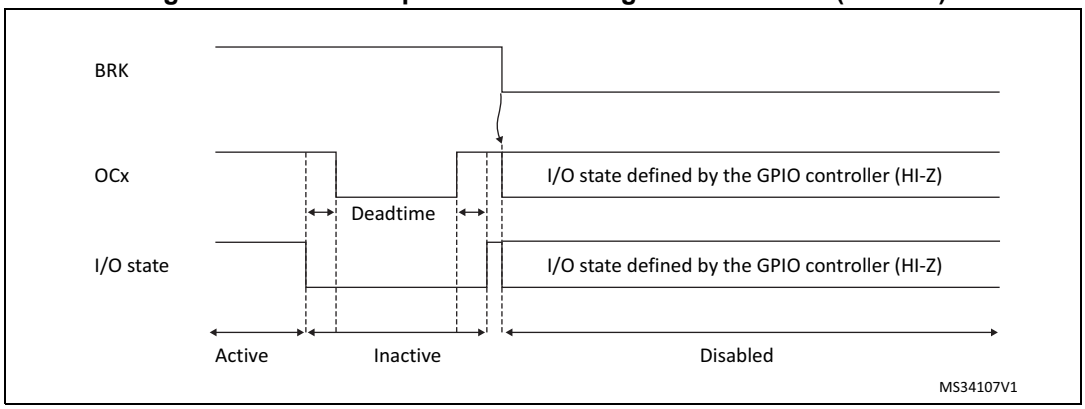


**Figure 161. PWM output state following BRK assertion (OSSI=0)**




## 19.3.17 Bidirectional break inputs

The TIM1 are featuring bidirectional break I/Os, as represented on *Figure 162*.

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 and break2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the break and break2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or 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 events (BG and B2G) also cause 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 (BK(2)E = 1). When a software break event is generated with BK(2)E = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break(2) 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 (BK2DSRM) 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 (BK2DSRM) 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 BK(2)DSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 126*)

**Table 126. Break protection disarming conditions**

| MOE | BKDIR (BK2DIR) | BKDSRM (BK2DSRM) | 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

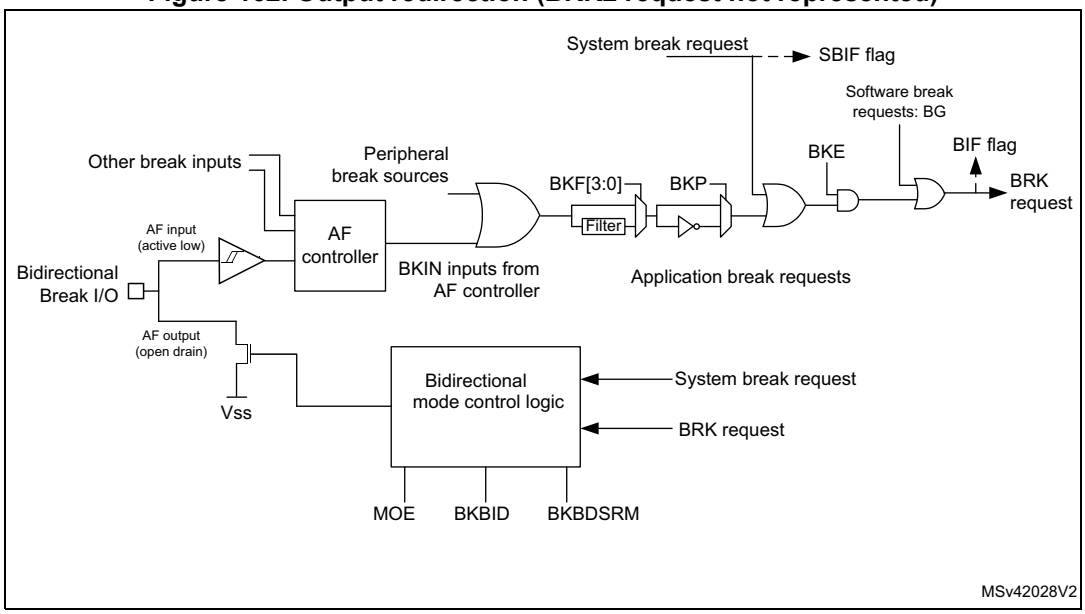
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) 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 162. Output redirection (BRK2 request not represented)**



### 19.3.18 Clearing the OCxREF signal on an external event

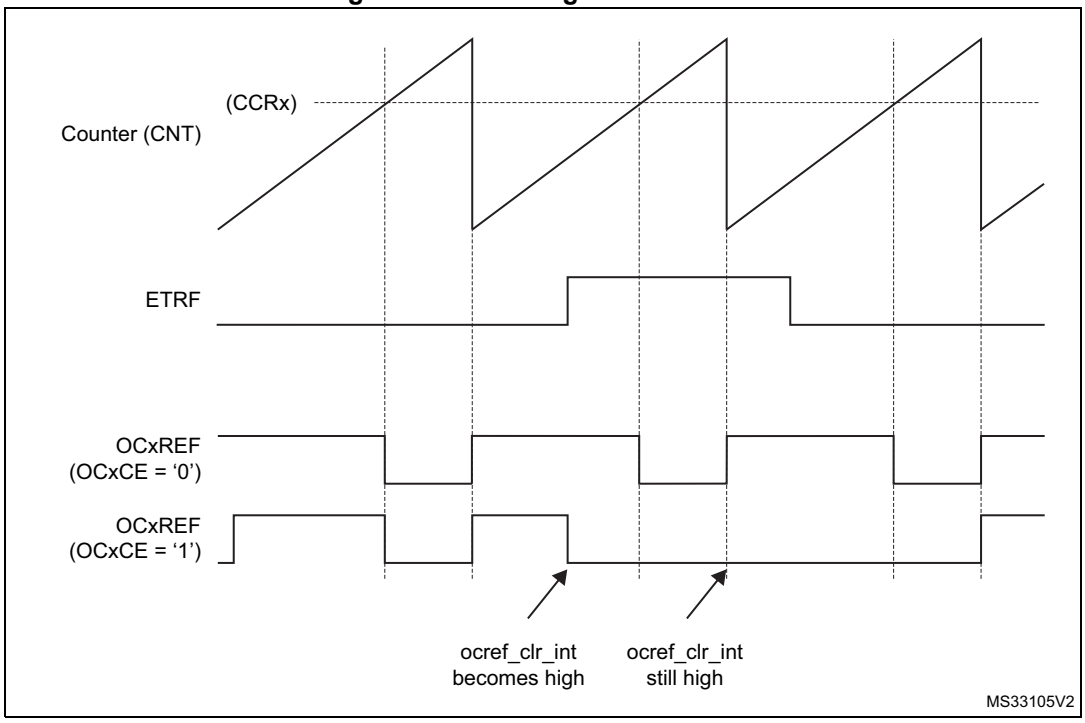
The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. ocref_clr_int input can be selected between the OCREF_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

When ETRF is chosen, ETR must be configured as follows:

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 163* 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 163. 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.*

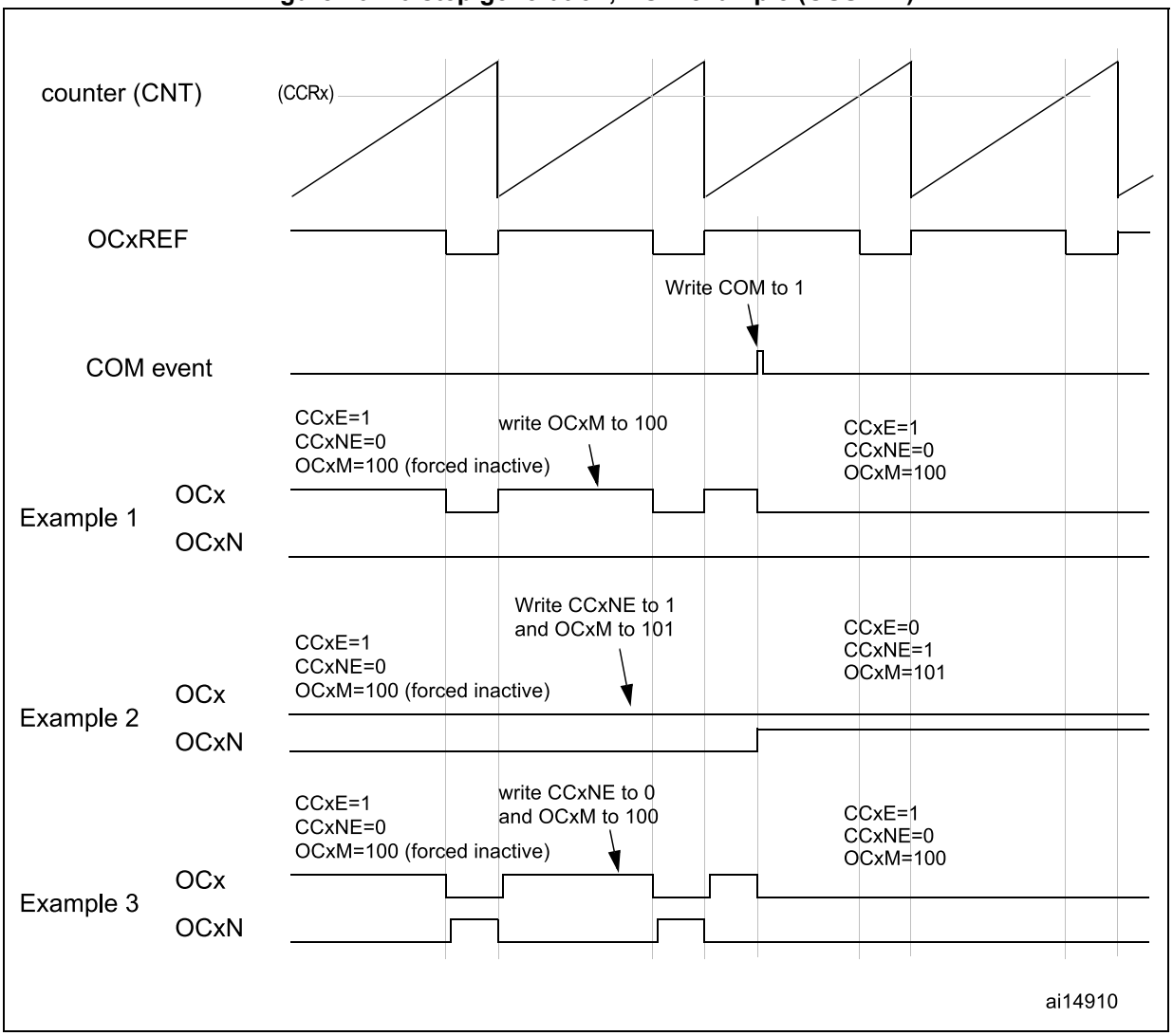
### 19.3.19 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 164* describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

**Figure 164. 6-step generation, COM example (OSSR=1)**



### 19.3.20 One-pulse mode

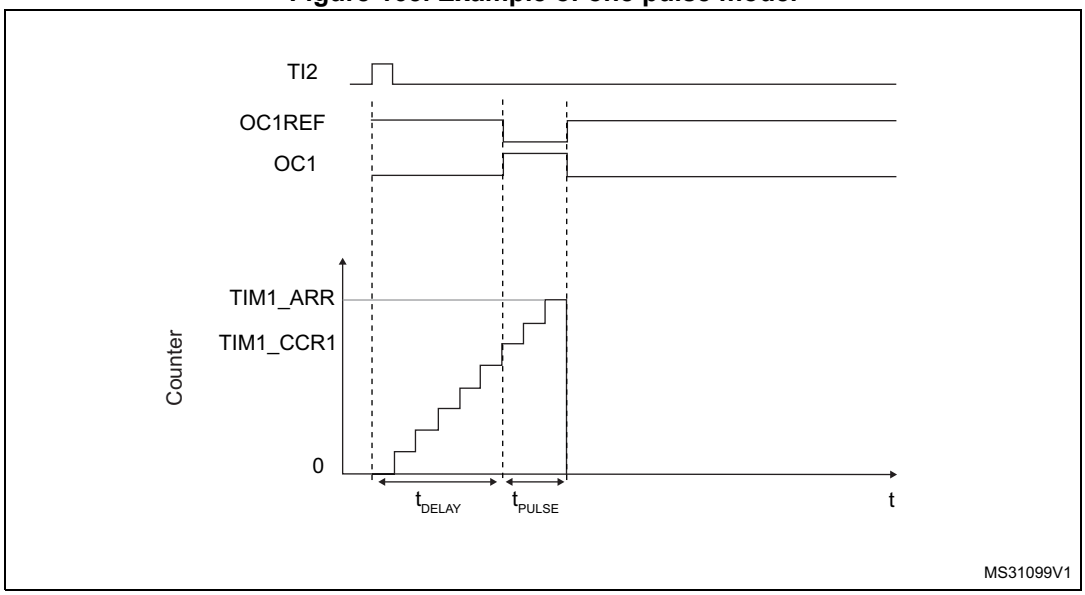
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected 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 165. Example of one pulse mode.**




For example one 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:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. 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 one 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 PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has 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.

Since only 1 pulse (Single mode) is needed, a 1 must be written 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 one wants to output a waveform with the minimum delay, the OCxFE bit can be set 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.

### 19.3.21 Retriggerable one pulse mode

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 19.3.20*:

- 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

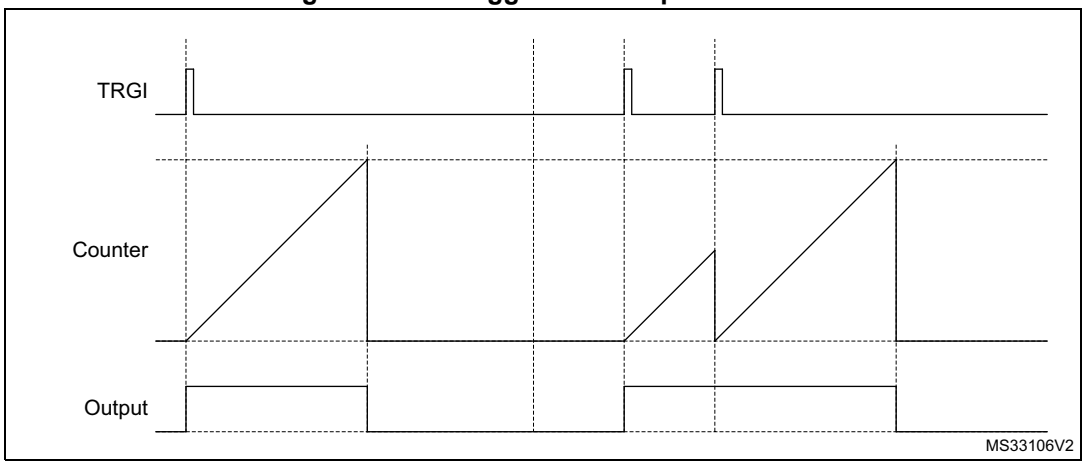
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for retriggerable OPM mode 1 or 2.

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: 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.*

*This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.*

**Figure 166. Retriggerable one pulse mode**




## 19.3.22 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, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to *Table 127*. 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 the TIMx_ARR must be configured before starting. In the same way, the capture, compare, 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 do not switch at the same time.

**Table 127. 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 167* 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_CCMR1 register, TI2FP2 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 167. Example of counter operation in encoder interface mode.**

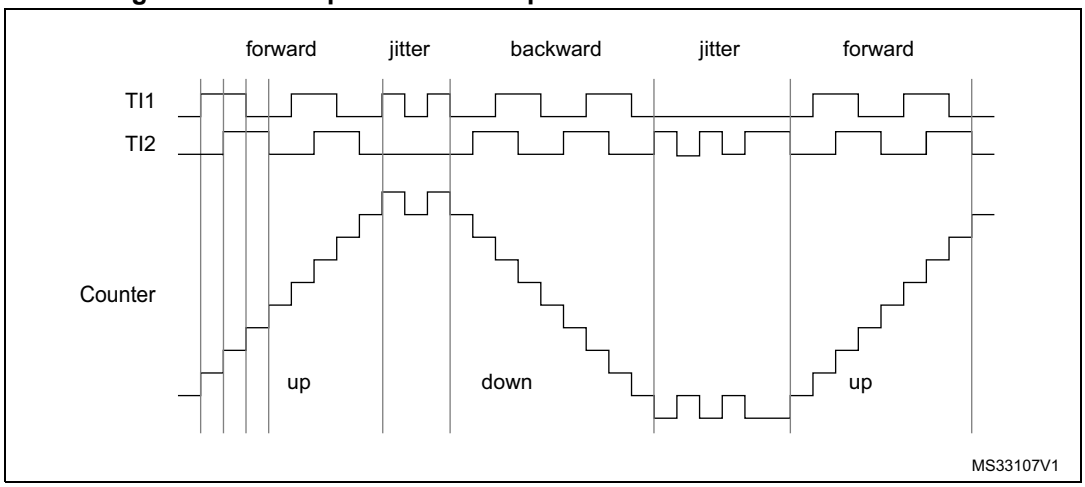
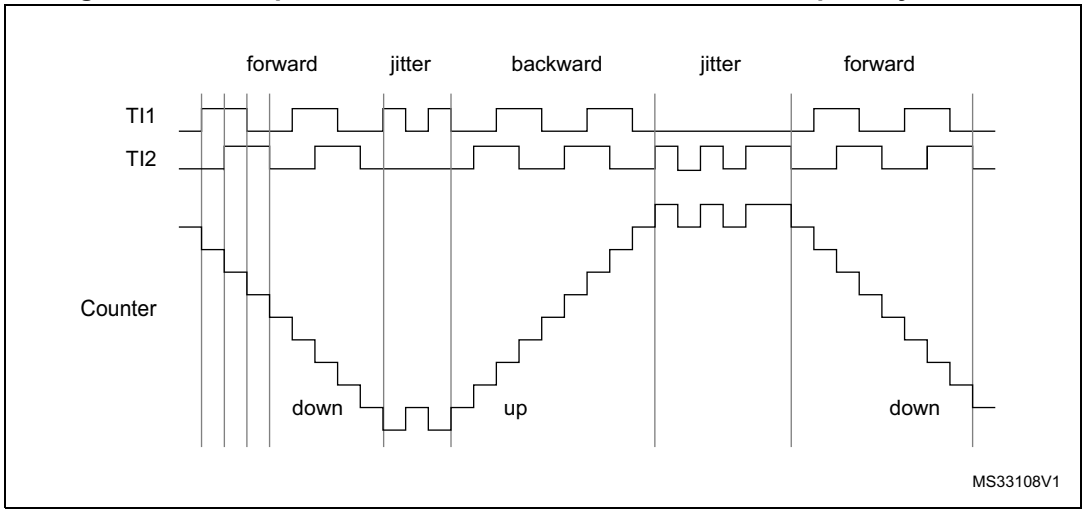


*Figure 168* gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

**Figure 168. 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. Dynamic information can be obtained (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. This can be done 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). when available, it is also possible to read its value through a DMA request.

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).

### 19.3.23 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).

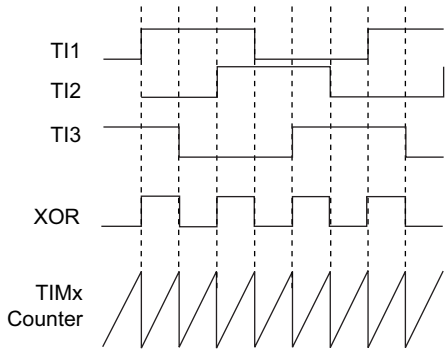
There is no latency between the UIF and UIFCPY flags assertion.

### 19.3.24 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 three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

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 169* below.

**Figure 169. Measuring time interval between edges on 3 signals**




### 19.3.25 Interfacing with Hall sensors

This is done using the advanced-control timer (TIM1) to generate PWM signals to drive the motor and another timer TIMx (TIM2) referred to as “interfacing timer” in *Figure 170*. The “interfacing timer” captures the 3 timer input pins (CC1, CC2, CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (See *Figure 143: Capture/compare channel (example: channel 1 input stage) on page 555*). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1) through the TRGO output.

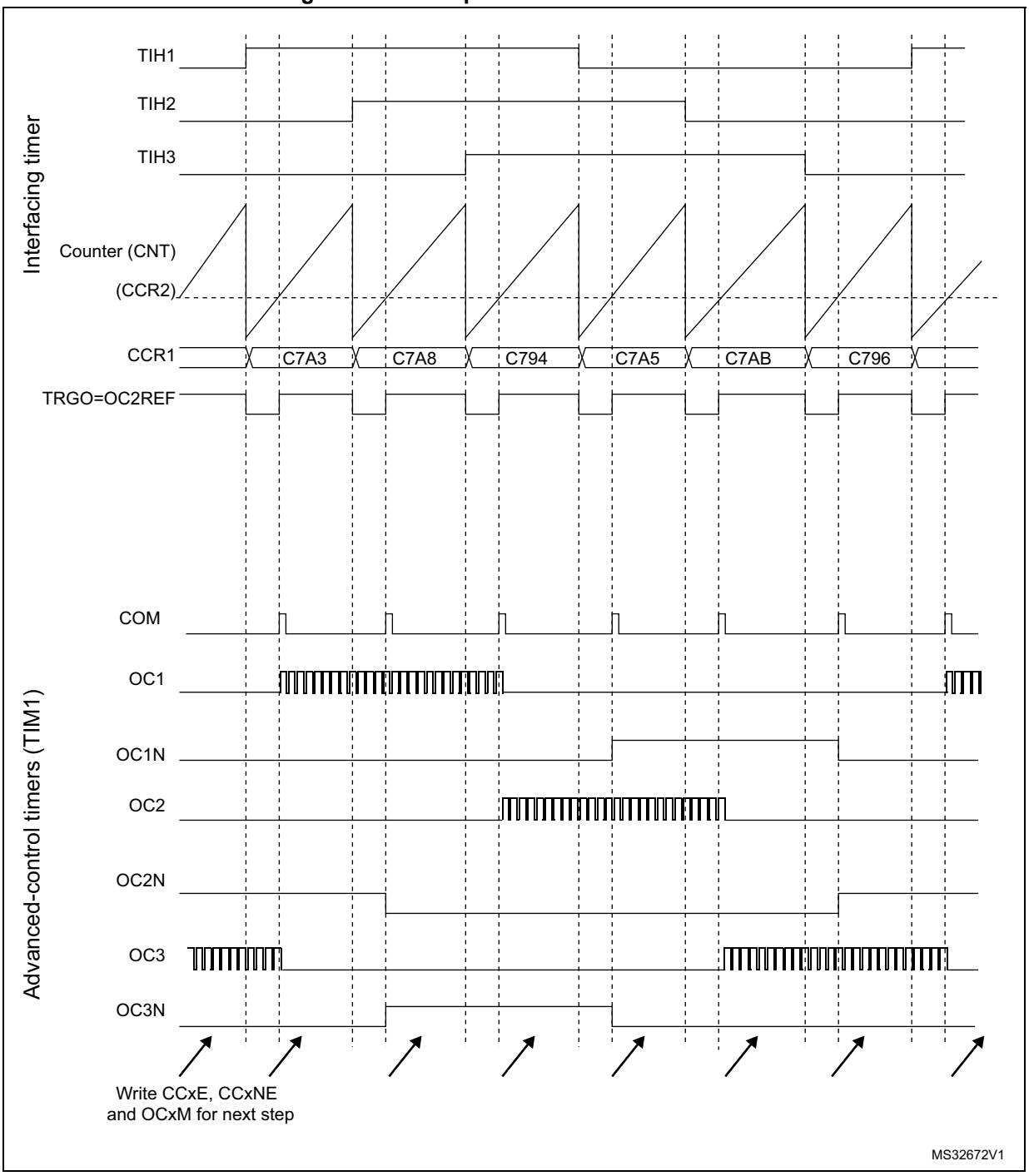
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘01’. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The *Figure 170* describes this example.

**Figure 170. Example of Hall sensor interface**



### 19.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 20.3.19: Timer synchronization* for details. They can be synchronized 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.

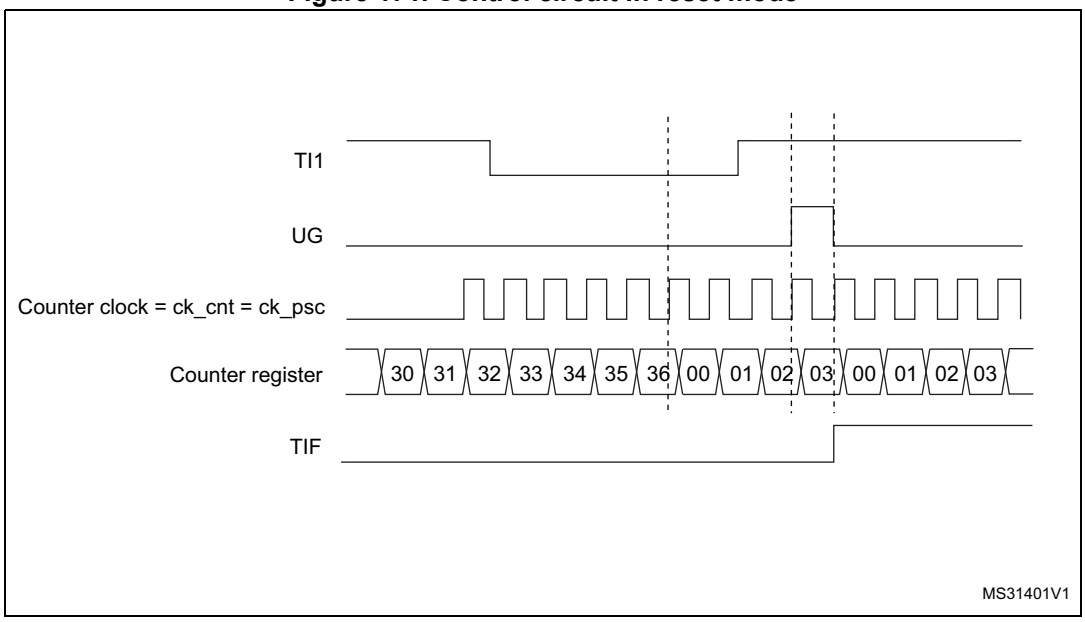
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- 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).
- 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.
- 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, or a DMA request can be sent if enabled (depending on the TIE and TDE bits 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 171. Control circuit in reset mode**



### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

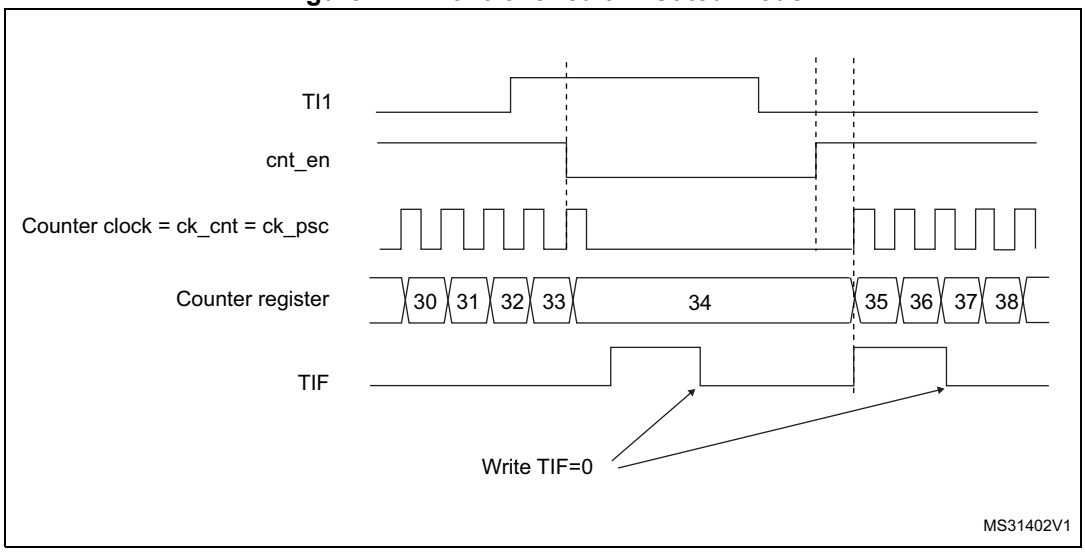
In the following example, the upcounter counts only when TI1 input is low:

- 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).
- 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.
- 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 172. Control circuit in Gated mode**




### 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:

- 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. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1

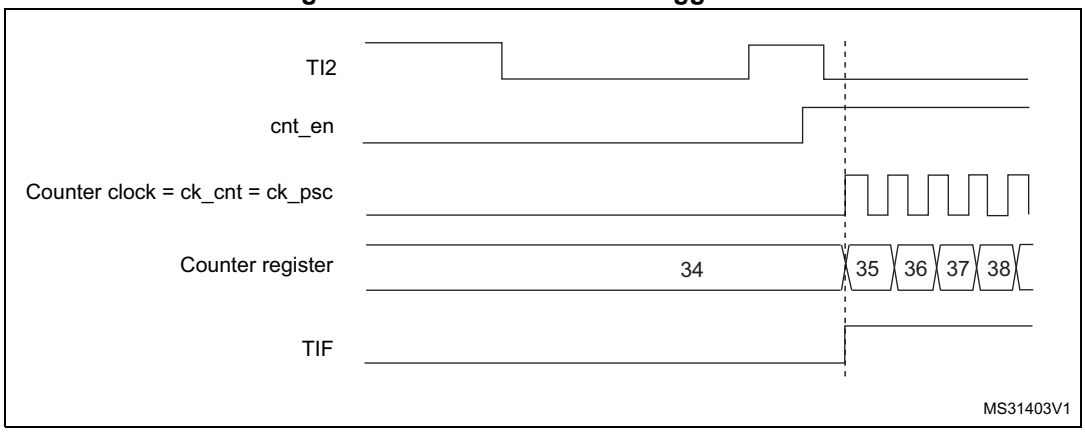
register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- 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 173. 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.

### 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 (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.

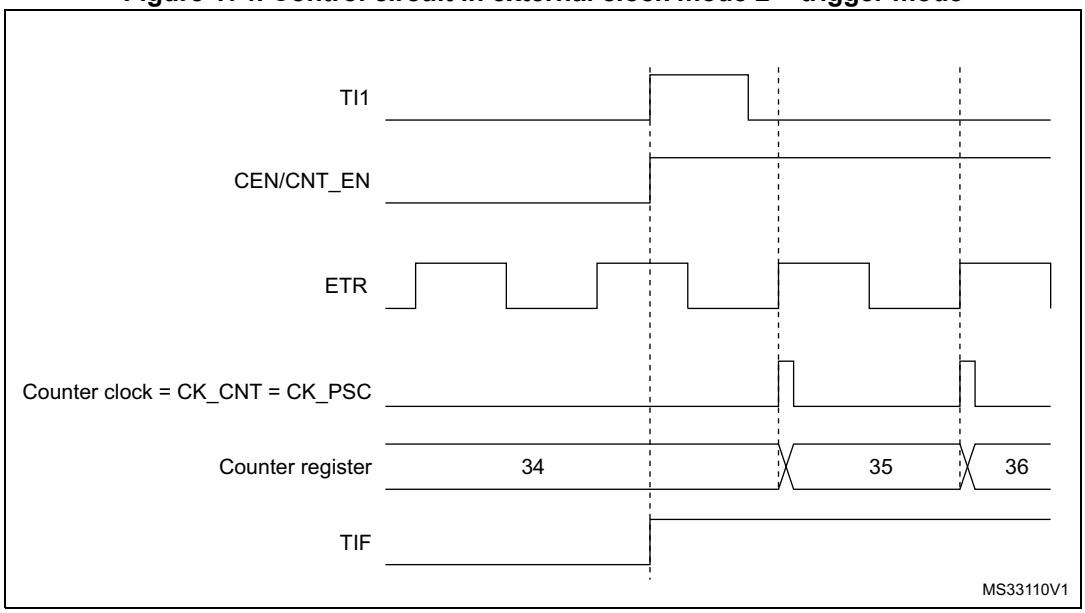
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

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=00101 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 174. Control circuit in external clock mode 2 + trigger mode**




*Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals 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.*

### 19.3.27 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 154 on page 567*.

*Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals 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: The clock of the ADC 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 timer.*

### 19.3.28 DMA burst mode

The TIMx timers have 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 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.*

### 19.3.29 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIM1_STOP configuration bit in DBGMCU module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

## 19.4 TIM1 registers

Refer to for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 19.4.1 TIM1 control register 1 (TIM1_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. | UIFREMAP | 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, 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 (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

*Note: $t_{DTS} = 1/f_{DTS}$, $t_{CK_INT} = 1/f_{CK_INT}$.*

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: Switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) is not allowed*

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.*

### 19.4.2 TIM1 control register 2 (TIM1_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. | MMS2[3:0] | | | | Res. | OIS6 | Res. | OIS5 |
| | | | | | | | | rw | rw | rw | rw | | rw | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | OIS4 | OIS3N | OIS3 | OIS2N | OIS2 | OIS1N | OIS1 | TI1S | MMS[2:0] | | | CCDS | CCUS | Res. | CCPC |
| | 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 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). 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 AND between the 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 TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).

0100: **Compare** - OC1REFC signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REFC signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REFC signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REFC signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REFC signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REFC signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REFC rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REFC rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REFC or OC6REFC rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REFC rising or OC6REFC falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REFC or OC6REFC rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REFC rising or OC6REFC falling edges generate pulses on TRGO2

*Note: The clock of the slave timer or ADC 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 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output Idle state 6 (OC6 output)

Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output Idle state 5 (OC5 output)

Refer to OIS1 bit

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)

Refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)

Refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)
Refer to OIS1 bit

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_BDTR 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_BDTR register).*

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 **MMS[2:0]**: Master mode selection
These bits allow selected 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 AND 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** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

*Note: The clock of the slave timer or ADC 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 **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 a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

*Note: This bit acts only on channels that have a complementary output.*

### 19.4.3 TIM1 slave mode control register (TIM1_SMCR)

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. | 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, must be kept at reset value.

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

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: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).*

*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 00111).*

*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 $f_{CK_INT}$ 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 consecutive 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, 6, 5, 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 *Table 128: TIM1 internal trigger connection on page 599* 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
1: OCREF_CLR_INT is connected to ETRF

Bits 16, 2, 1, 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.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 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.

Codes above 1000: 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 (timer, ADC, ...) receiving the TRGO or the TRGO2 signals 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 128. TIM1 internal trigger connection**

| Slave TIM | ITR0 (TS = 00000) | ITR1 (TS = 00001) | ITR2 (TS = 00010) | ITR3 (TS = 00011) |
|---|---|---|---|---|
| TIM1 | - | TIM2 | - | TIM17 OC1 |

### 19.4.4 TIM1 DMA/interrupt enable register (TIM1_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 | COMDE | CC4DE | CC3DE | CC2DE | CC1DE | UDE | BIE | TIE | COMIE | CC4IE | CC3IE | CC2IE | CC1IE | UIE |
| | 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 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM 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 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

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

### 19.4.5 TIM1 status register (TIM1_SR)

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. | CC6IF | CC5IF |
| | | | | | | | | | | | | | | rc_w0 | rc_w0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | SBIF | CC4OF | CC3OF | CC2OF | CC1OF | B2IF | BIF | TIF | COMIF | 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 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

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

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description (Note: Channel 6 can only be configured as output)

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description (Note: Channel 5 can only be configured as output)

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

Bit 13 **SBIF**: System Break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

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 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

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. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG 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 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

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

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

**If channel CC1 is configured as output**: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

**If channel CC1 is configured as input**: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

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 19.4.3: TIM1 slave mode control register (TIM1_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

## 19.4.6 TIM1 event generation register (TIM1_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. | B2G | BG | TG | COMG | CC4G | CC3G | CC2G | CC1G | UG |
| | | | | | | | w | w | w | w | w | w | w | w | w |

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 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 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

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 CCPC bit is set, it allows CCxE, CCxNE and OCxM bits to be updated.

*Note: This bit acts only on channels having a complementary output.*

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. The prescaler internal counter is also cleared (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).

### 19.4.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Input capture 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IC2F[3:0] | | | | IC2PSC[1:0] | | CC2S[1:0] | | IC1F[3:0] | | | | IC1PSC[1:0] | | CC1S[1:0] | |
| 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 15:12 **IC2F[3:0]**: Input capture 2 filter
Refer to IC1F[3:0] description.

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler
Refer to IC1PSC[1:0] description.

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 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 consecutive 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[1:0]**: 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[1:0]**: 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).*

### 19.4.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Output compare mode:**

| 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] |
| | | | | | | | rw | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OC2 CE | OC2M[2:0] | | | OC2 PE | OC2 FE | CC2S[1:0] | | OC1 CE | OC1M[2:0] | | | OC1 PE | OC1 FE | CC1S[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output Compare 2 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode
Refer to OC1M[3:0] description.

Bit 11 **OC2PE**: Output Compare 2 preload enable
Refer to OC1PE description.

Bit 10 **OC2FE**: Output Compare 2 fast enable
Refer to OC1FE description.

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
0: OC1Ref is not affected by the ocref_clr_int signal
1: OC1Ref is cleared as soon as a High level is detected on ocref_clr_int signal (OCREF_CLR input or ETRF input)

Bits 16, 6:4 **OC1M[3:0]**: 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 can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

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: Retriggerable 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: Retriggerable 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: 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).*

*Note: 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.*

*Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.*

*Note: The OC1M[3] bit is not contiguous, located in bit 16.*

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: 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).*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

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[1:0]**: 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).*

## 19.4.9 TIM1 capture/compare mode register 2 (TIM1_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Input capture 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IC4F[3:0] | | | | IC4PSC[1:0] | | CC4S[1:0] | | IC3F[3:0] | | | | IC3PSC[1:0] | | CC3S[1:0] | |
| 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 15:12 **IC4F[3:0]**: Input capture 4 filter
Refer to IC1F[3:0] description.

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
Refer to IC1PSC[1:0] description.

Bits 9:8 **CC4S[1:0]**: 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[3:0]**: Input capture 3 filter
Refer to IC1F[3:0] description.

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler
Refer to IC1PSC[1:0] description.

Bits 1:0 **CC3S[1:0]**: 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).*

### 19.4.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Output compare mode**

| 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] |
| | | | | | | | 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] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] description.

Bit 11 **OC4PE**: Output compare 4 preload enable
Refer to OC1PE description.

Bit 10 **OC4FE**: Output compare 4 fast enable
Refer to OC1FE description.

Bits 9:8 **CC4S[1:0]**: 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
Refer to OC1CE description.

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M[3:0] description.

Bit 3 **OC3PE**: Output compare 3 preload enable
Refer to OC1PE description.

Bit 2 **OC3FE**: Output compare 3 fast enable
Refer to OC1FE description.

Bits 1:0 **CC3S[1:0]**: 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).*

### 19.4.11 TIM1 capture/compare enable register (TIM1_CCER)

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. | CC6P | CC6E | Res. | Res. | CC5P | CC5E |
| | | | | | | | | | | rw | rw | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CC4NP | Res. | CC4P | CC4E | CC3NP | CC3NE | CC3P | CC3E | CC2NP | CC2NE | CC2P | CC2E | CC1NP | CC1NE | CC1P | CC1E |
| rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/Compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

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 **CC3NP**: Capture/Compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable
Refer to CC1NE description

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 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable
Refer to CC1NE description

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 **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 and TI2FP1. Refer to CC1P description.

*Note: 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" (channel configured as output).*

*On channels having 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.

*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 1 **CC1P**: Capture/Compare 1 output polarity
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: 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).
CC1NP=0, CC1P=1: 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).
CC1NP=1, CC1P=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), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: The configuration is reserved, it must not be used.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

*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

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**When CC1 channel is configured as output**, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 129* for details.

*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 129. Output control bits for complementary OCx and OCxN channels with break feature**

| Control bits | | | | | Output states(1) | |
|---|---|---|---|---|---|---|
| **MOE bit** | **OSSI bit** | **OSSR bit** | **CCxE bit** | **CCxNE bit** | **OCx output state** | **OCxN output state** |
| 1 | X | X | 0 | 0 | Output disabled (not driven by the timer: Hi-Z)<br>OCx=0, OCxN=0 | |
| | | 0 | 0 | 1 | Output disabled (not driven by the timer: Hi-Z)<br>OCx=0 | OCxREF + Polarity<br>OCxN = OCxREF xor CCxNP |
| | | 0 | 1 | 0 | OCxREF + Polarity<br>OCx=OCxREF xor CCxP | Output Disabled (not driven by the timer: Hi-Z)<br>OCxN=0 |
| | | X | 1 | 1 | OCREF + Polarity + dead-time | Complementary to OCREF (not OCREF) + Polarity + dead-time |
| | | 1 | 0 | 1 | Off-State (output enabled with inactive state)<br>OCx=CCxP | OCxREF + Polarity<br>OCxN = OCxREF x or CCxNP |
| | | 1 | 1 | 0 | OCxREF + Polarity<br>OCx=OCxREF xor CCxP | Off-State (output enabled with inactive state)<br>OCxN=CCxNP |
| 0 | 0 | X | X | X | Output disabled (not driven by the timer: Hi-Z). | |
| | 1 | | 0 | 0 | | |
| | | | 0 | 1 | Off-State (output enabled with inactive state)<br>Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered).<br>Then (this is valid only if BRK is triggered), 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 (may cause a short circuit when driving switches in half-bridge configuration).<br>**Note**: BRK2 can only be used if OSSI = OSSR = 1. | |
| | | | 1 | 0 | | |
| | | | 1 | 1 | | |

1. When both outputs of a channel are not used (control taken over by GPIO), 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 the GPIO registers.*

### 19.4.12 TIM1 counter (TIM1_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

| 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 the TIMxCR1 is reset, bit 31 is reserved and read at 0.

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

Bits 15:0 **CNT[15:0]**: Counter value

### 19.4.13 TIM1 prescaler (TIM1_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”).

### 19.4.14 TIM1 auto-reload register (TIM1_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]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 19.3.1: Time-base unit on page 535* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 19.4.15 TIM1 repetition counter register (TIM1_RCR)

Address offset: 0x30

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REP[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:0 **REP[15:0]**: 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.

### 19.4.16 TIM1 capture/compare register 1 (TIM1_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**: CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

### 19.4.17 TIM1 capture/compare register 2 (TIM1_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_CCMR1 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 signaled on OC2 output.

**If channel CC2 is configured as input**: CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

### 19.4.18 TIM1 capture/compare register 3 (TIM1_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_CCMR2 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). The TIMx_CCR3 register is read-only and cannot be programmed.

### 19.4.19 TIM1 capture/compare register 4 (TIM1_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_CCMR2 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 signalled on OC4 output.

**If channel CC4 is configured as input**: CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

### 19.4.20 TIM1 break and dead-time register (TIM1_BDTR)

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. | BK2BID | BKBID | BK2 DSRM | BK DSRM | BK2P | BK2E | BK2F[3:0] | | | | BKF[3:0] | | | |
| | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 bits BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.*

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

Bit 29 **BK2BID**: Break2 bidirectional

Refer to BKBID description

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 **BK2DSRM**: Break2 Disarm

Refer to BKDSRM description

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 (open-drain 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.*

Bit 25 **BK2P**: Break 2 polarity

0: Break input BRK2 is active low
1: Break input BRK2 is active high

*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 24 **BK2E**: Break 2 enable

*Note: The  must only be used with OSSR = OSSI = 1.*

*Note: This bit cannot be modified when 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.*

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK2 acts asynchronously
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

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously
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

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). 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: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 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 19.4.11: TIM1 capture/compare enable register (TIM1_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 none of the break inputs BRK and BRK2 is 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: This bit can not 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 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk_acth and BRK sources, as per *Figure 158: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

*Note: This bit cannot be modified when 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 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having 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 19.4.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO 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 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 19.4.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

*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 and B*K2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0]* bits in TIMx_BDTR 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]) x $t_{DTG}$ with $t_{DTG} = 2 \times t_{DTS}$.
DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x $t_{DTG}$ with $t_{DTG} = 8 \times t_{DTS}$.
DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x $t_{DTG}$ with $t_{DTG} = 16 \times t_{DTS}$.
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).*

## 19.4.21 TIM1 DMA control register (TIM1_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 vector 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

**Example:** Let us consider the following transfer: DBL = 7 bytes & DBA = TIMx_CR1.

– If DBL = 7 bytes and DBA = TIMx_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

– If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.
– If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector 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: TIMx_SMCR,
...

### 19.4.22 TIM1 DMA address for full transfer (TIM1_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMAB[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 |
| DMAB[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **DMAB[31:0]**: 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).

## 19.4.23 TIM1 option register 1 (TIM1_OR1)

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. | TI1_ RMP | Res. | Res. | TIM1_ETR_ ADC_RMP[1:0] | |
| | | | | | | | | | | | rw | | | rw | rw |

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

Bit 4 **TI1_RMP**: Input Capture 1 remap

0: TIM1 input capture 1 is connected to I/O

Others: Reserved

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **TIM1_ETR_ADC_RMP[1:0]**: TIM1_ETR_ADC remapping capability

00: TIM1_ETR is not connected to ADC AWDx (must be selected when the ETR comes from the ETR input pin)

01: TIM1_ETR is connected to ADC AWD1

10: TIM1_ETR is connected to ADC AWD2

11: TIM1_ETR is connected to ADC AWD3

*Note: ADC AWDx sources are 'ORed' with the TIM1_ETR input signals. When ADC AWDx is used, it is necessary to make sure that the corresponding TIM1_ETR input pin is not enabled in the alternate function controller.*

## 19.4.24 TIM1 capture/compare mode register 3 (TIM1_CCMR3)

Address offset: 0x54

Reset value: 0x0000 0000

The channels 5 and 6 can only be configured in output.

**Output compare mode:**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | OC6M[3] | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OC5M[3] |
| | | | | | | | rw | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OC6 CE | OC6M[2:0] | | | OC6 PE | OC6FE | Res. | Res. | OC5 CE | OC5M[2:0] | | | OC5PE | OC5FE | Res. | Res. |
| rw | rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | | |

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable
Refer to OC1CE description.

Bits 24, 14, 13, 12 **OC6M[3:0]**: Output compare 6 mode
Refer to OC1M description.

Bit 11 **OC6PE**: Output compare 6 preload enable
Refer to OC1PE description.

Bit 10 **OC6FE**: Output compare 6 fast enable
Refer to OC1FE description.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable
Refer to OC1CE description.

Bits 16, 6, 5, 4 **OC5M[3:0]**: Output compare 5 mode
Refer to OC1M description.

Bit 3 **OC5PE**: Output compare 5 preload enable
Refer to OC1PE description.

Bit 2 **OC5FE**: Output compare 5 fast enable
Refer to OC1FE description.

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

### 19.4.25 TIM1 capture/compare register 5 (TIM1_CCR5)

Address offset: 0x58

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GC5C3 | GC5C2 | GC5C1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| rw | rw | rw | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR5[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **GC5C3**: Group Channel 5 and Channel 3

Distortion on Channel 3 output:
0: No effect of OC5REF on OC3REFC
1: OC3REFC is the logical AND of OC3REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).

*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 30 **GC5C2**: Group Channel 5 and Channel 2

Distortion on Channel 2 output:
0: No effect of OC5REF on OC2REFC
1: OC2REFC is the logical AND of OC2REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 29 **GC5C1**: Group Channel 5 and Channel 1

Distortion on Channel 1 output:
0: No effect of OC5REF on OC1REFC5
1: OC1REFC is the logical AND of OC1REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

*Note: it is also possible to apply this distortion on combined PWM signals.*

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value

CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 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 OC5 output.

### 19.4.26 TIM1 capture/compare register 6 (TIM1_CCR6)

Address offset: 0x5C

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR6[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 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 OC6 output.

### 19.4.27 TIM1 alternate function option register 1 (TIM1_AF1)

Address offset: 0x60

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. | ETRSEL[3:2] | |
| | | | | | | | | | | | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETRSEL[1:0] | | Res. | Res. | Res. | Res. | BKINP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BKINE |
| rw | rw | | | | | rw | | | | | | | | | rw |

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

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.

0000: ETR legacy mode

Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:10 Reserved, must be kept at reset value.

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 polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: BKIN input polarity is inverted (active high if BKP=0, active low if BKP=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other 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).*

*Note: Refer to Figure 137: TIM1 ETR input circuitry and to Figure 158: Break and Break2 circuitry overview.*

### 19.4.28 TIM1 Alternate function register 2 (TIM1_AF2)

Address offset: 0x64

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. | BK2 INP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BK2INE |
| | | | | | | rw | | | | | | | | | rw |

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

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: BKIN2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **BK2INE**: BRK2 BKIN input enable

This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.

0: BKIN2 input disabled

1: BKIN2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

*Note: Refer to Figure 158: Break and Break2 circuitry overview.*

### 19.4.29 TIM1 timer input selection register (TIM1_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

| 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: TIM1_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: TIM1_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: TIM1_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: TIM1_CH1 input
Others: Reserved

## 19.4.30 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 130. TIM1 register map and reset values**

| Offset | Register name | 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 | **TIM1_CR1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | UIFREMAP | 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 | **TIM1_CR2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MMS2[3:0] | | | | Res. | OIS6 | Res. | OIS5 | Res. | OIS4 | OIS3N | OIS3 | OIS2N | OIS2 | OIS1N | OIS1 | TI1S | MMS [2:0] | | | CCDS | CCUS | Res. | CCPC |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | | 0 | | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 |
| 0x08 | **TIM1_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 | **TIM1_DIER** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TDE | COMDE | CC4DE | CC3DE | CC2DE | CC1DE | UDE | BIE | TIE | COMIE | CC4IE | CC3IE | CC2IE | CC1IE | UIE |
| | Reset value | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10 | **TIM1_SR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CC6IF | CC5IF | Res. | Res. | SBIF | CC4OF | CC3OF | CC2OF | CC1OF | B2IF | BIF | TIF | COMIF | CC4IF | CC3IF | CC2IF | CC1IF | UIF |
| | Reset value | | | | | | | | | | | | | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x14 | **TIM1_EGR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | B2G | BG | TG | COMG | CC4G | CC3G | CC2G | CC1G | UG |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x18 | **TIM1_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 |
| | **TIM1_CCMR1** Input Capture mode | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | IC2F[3:0] | | | | IC2PSC [1:0] | | CC2S [1:0] | | IC1F[3:0] | | | | IC1PSC [1:0] | | CC1S [1:0] | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1C | **TIM1_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 |
| | **TIM1_CCMR2** Input Capture mode | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | IC4F[3:0] | | | | IC4PSC [1:0] | | CC4S [1:0] | | IC3F[3:0] | | | | IC3PSC [1:0] | | CC3S [1:0] | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x20 | **TIM1_CCER** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CC6P | CC6E | Res. | Res. | CC5P | CC5E | CC4NP | Res. | CC4P | CC4E | CC3NP | CC3NE | CC3P | CC3E | CC2NP | CC2NE | CC2P | CC2E | CC1NP | CC1NE | CC1P | CC1E |
| | Reset value | | | | | | | | | | | 0 | 0 | | | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 130. TIM1 register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x24 | TIM1_CNT | UIFCP | 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 | TIM1_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 | TIM1_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 | TIM1_RCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REP[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x34 | TIM1_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 | TIM1_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 | TIM1_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 | TIM1_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 |
| 0x44 | TIM1_BDTR | Res. | Res. | BK2BID | BKBID | BK2DSRM | BKDSRM | BK2P | BK2E | BK2F[3:0] | | | | BKF[3:0] | | | | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x48 | TIM1_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 | TIM1_DMAR | DMAB[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x50 | TIM1_OR1 | Res. | Res. | Res. | 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 | Res. | Res. | TIM1_ETR_ADC_RMP | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | 0 | 0 |

**Table 130. TIM1 register map and reset values (continued)**

| Offset | Register name | 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 | **TIM1_CCMR3** Output Compare mode | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OC6M[3] | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OC5M[3] | OC6CE | OC6M [2:0] | | | OC6PE | OC6FE | Res. | Res. | OC5CE | OC5M [2:0] | | | OC5PE | OC5FE | Res. | Res. |
| | Reset value | | | | | | | | 0 | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 0x58 | **TIM1_CCR5** | GC5C3 | GC5C2 | GC5C1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CCR5[15:0] | | | | | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x5C | **TIM1_CCR6** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CCR6[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x60 | **TIM1_AF1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res ETRSEL [3:0] | | | | Res. | Res. | Res. | Res. | BKINP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BKINE |
| | Reset value | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | | | | | 0 | | | | | | | | | 1 |
| 0x64 | **TIM1_AF2** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BK2INP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BK2INE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | | | | | 1 |
| 0x68 | **TIM1_TISEL** | Res. | Res. | Res. | Res. | TI4SEL[3:0] | | | | Res. | Res. | Res. | Res. | TI3SEL[3:0] | | | | Res. | 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 on page 56* for the register boundary addresses.

# 20 General-purpose timer (TIM2)

## 20.1 TIM2 introduction

The general-purpose timer TIM2 consists of a 32-bit auto-reload counter driven by a programmable prescaler.

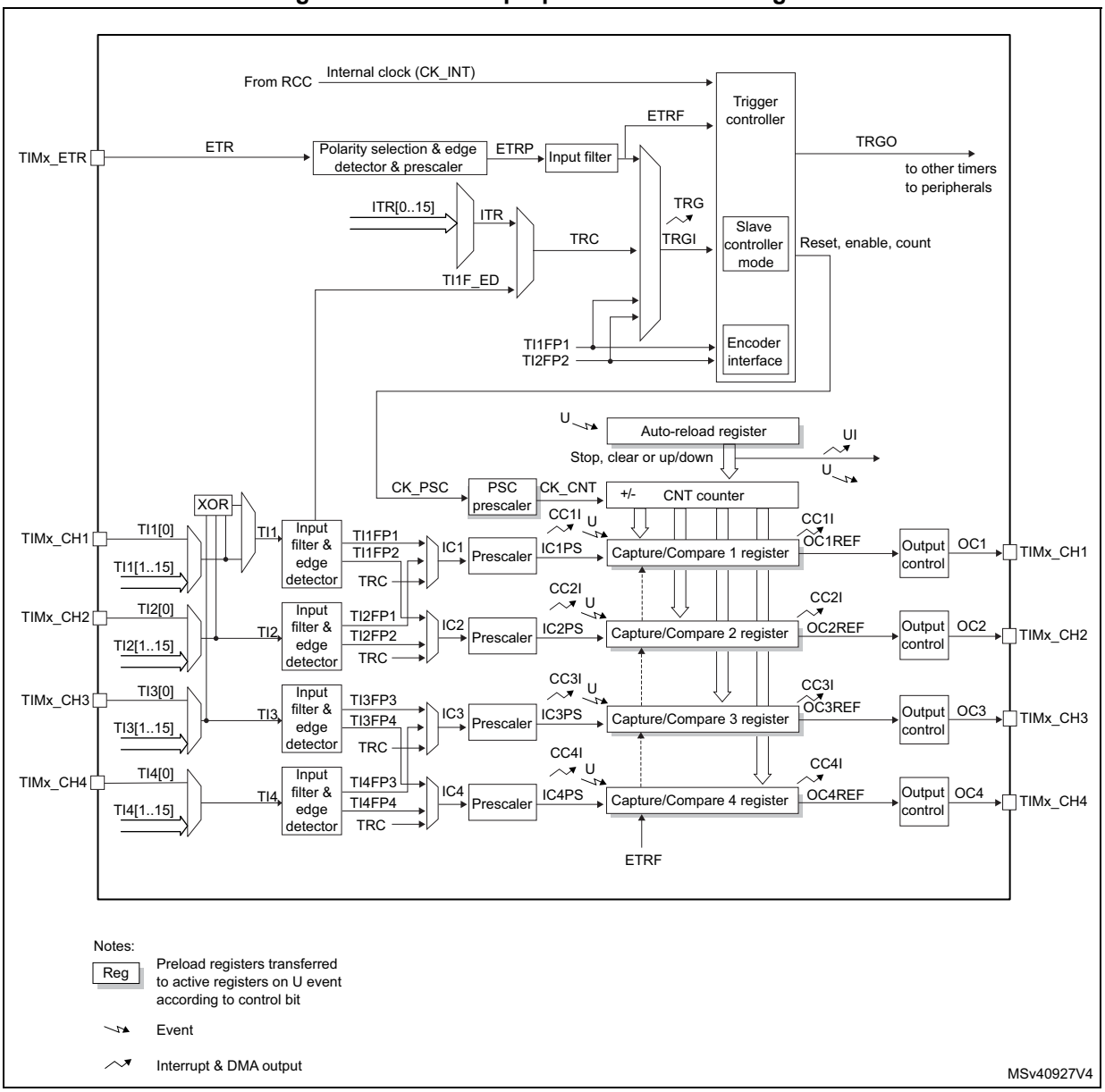
The timer may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

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.

## 20.2 TIM2 main features

- 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- 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 and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

**Figure 175. General-purpose timer block diagram**



## 20.3 TIM2 functional description

### 20.3.1 Time-base unit

The main block of the programmable timer is a 32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also 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)

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 detail 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 actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

#### 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/32-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 176* and *Figure 177* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 176. Counter timing diagram with prescaler division change from 1 to 2**

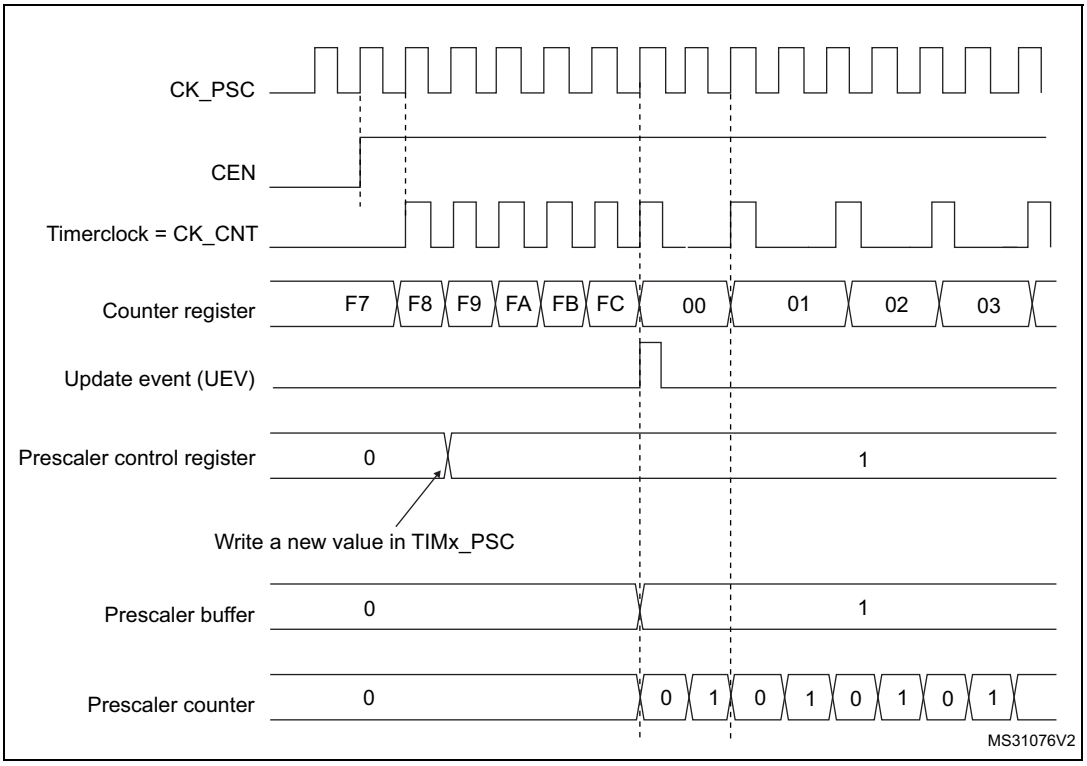
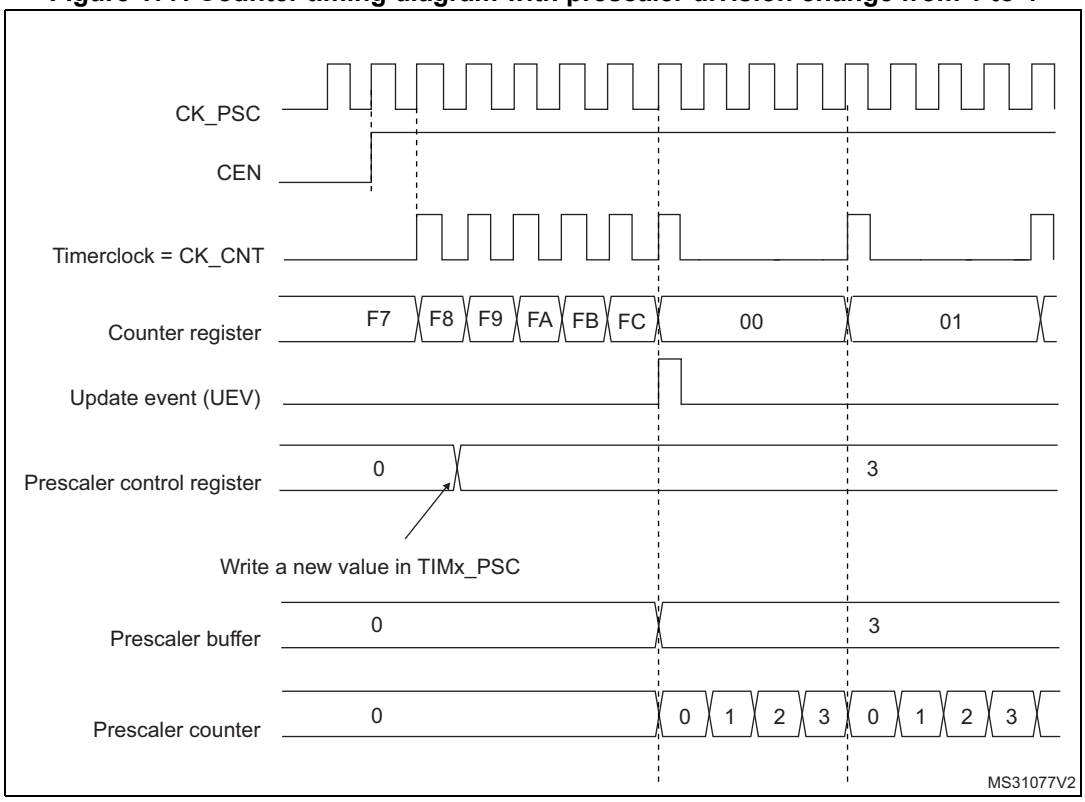



**Figure 177. Counter timing diagram with prescaler division change from 1 to 4**



## 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.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV 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 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 buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

**Figure 178. Counter timing diagram, internal clock divided by 1**

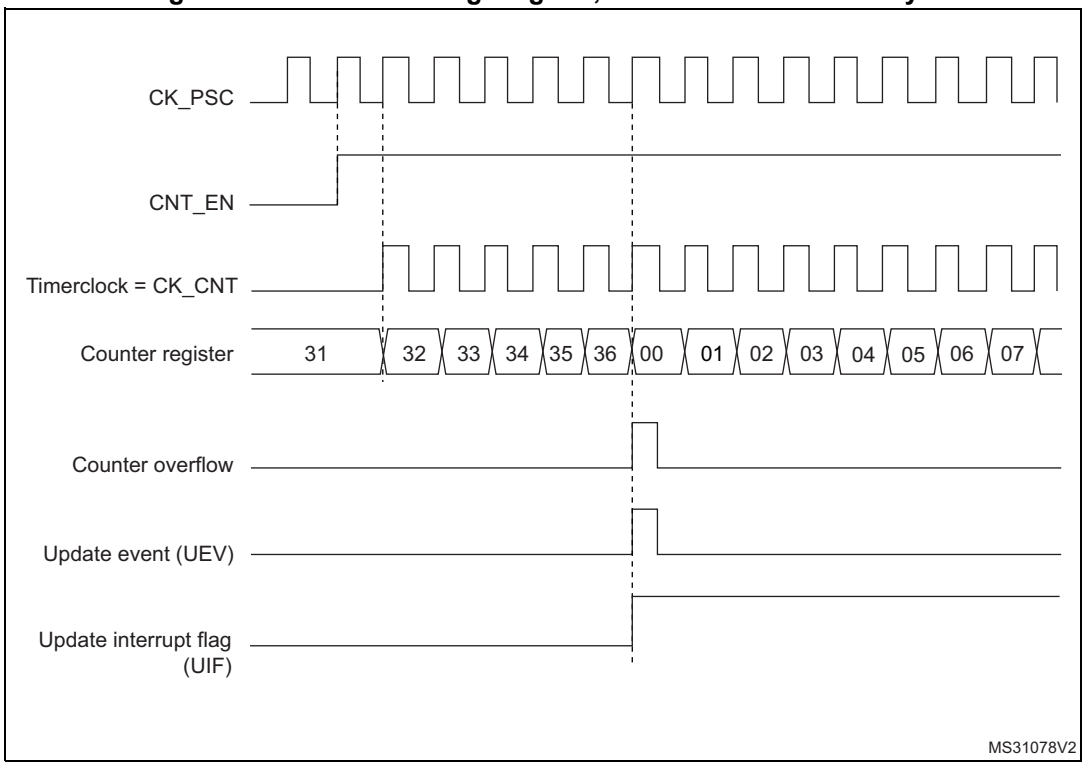


**Figure 179. Counter timing diagram, internal clock divided by 2**

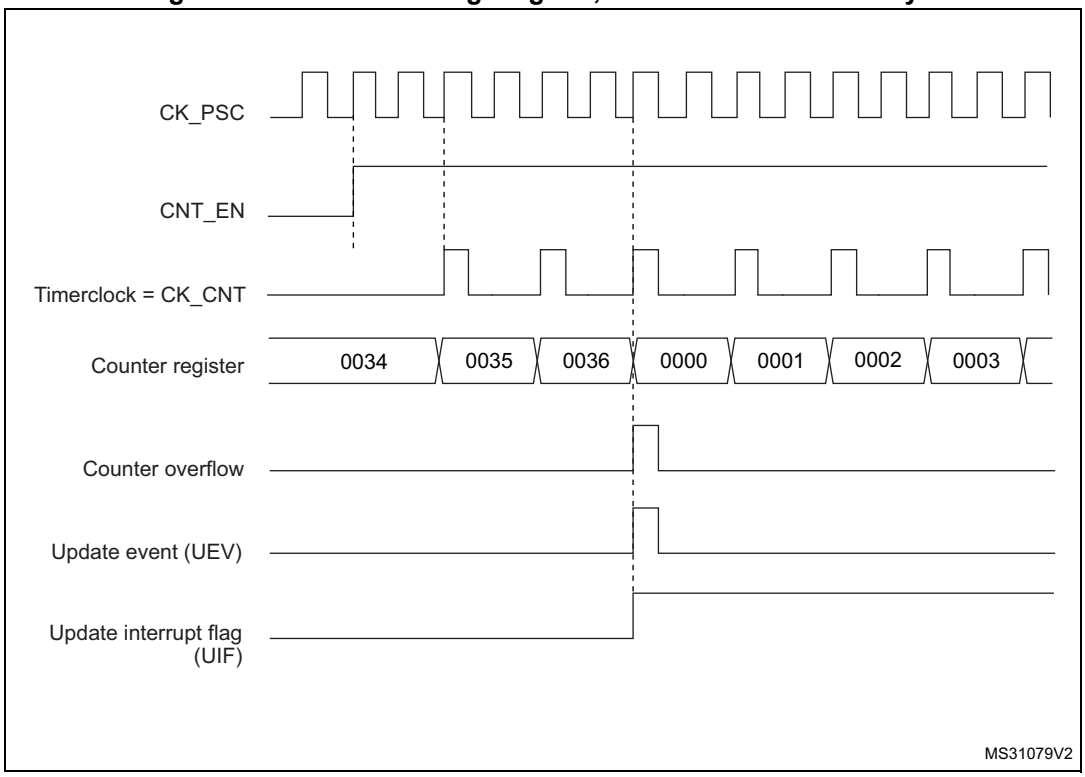



**Figure 180. Counter timing diagram, internal clock divided by 4**

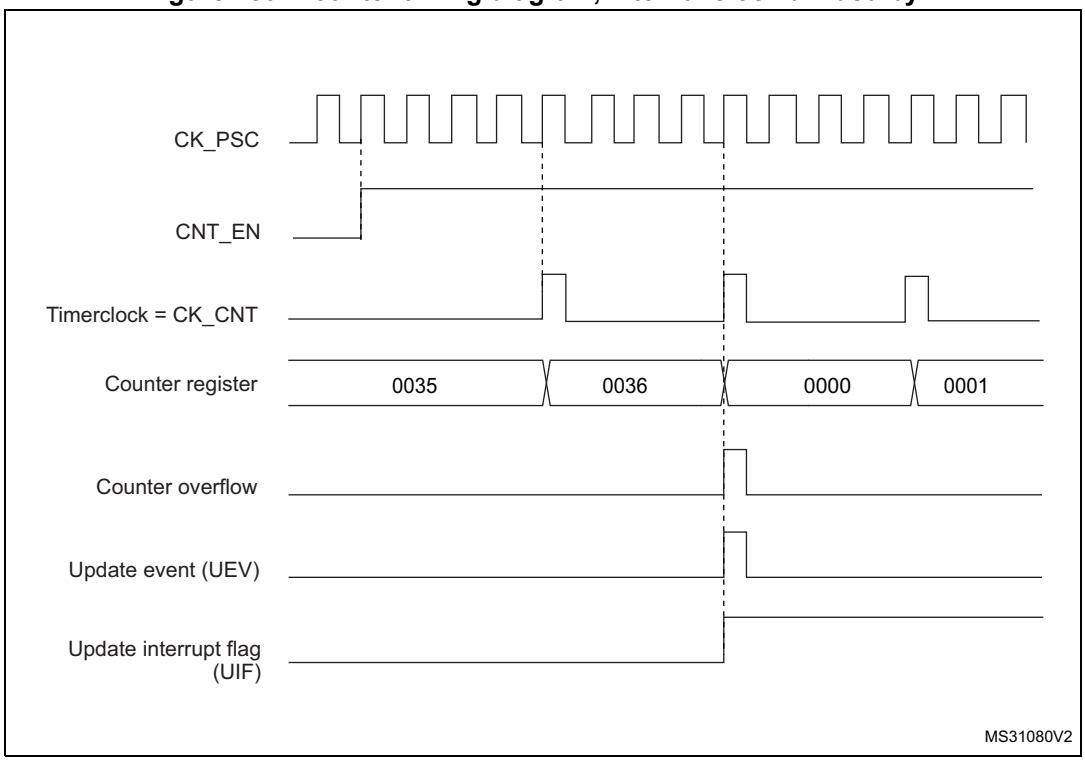


**Figure 181. Counter timing diagram, internal clock divided by N**

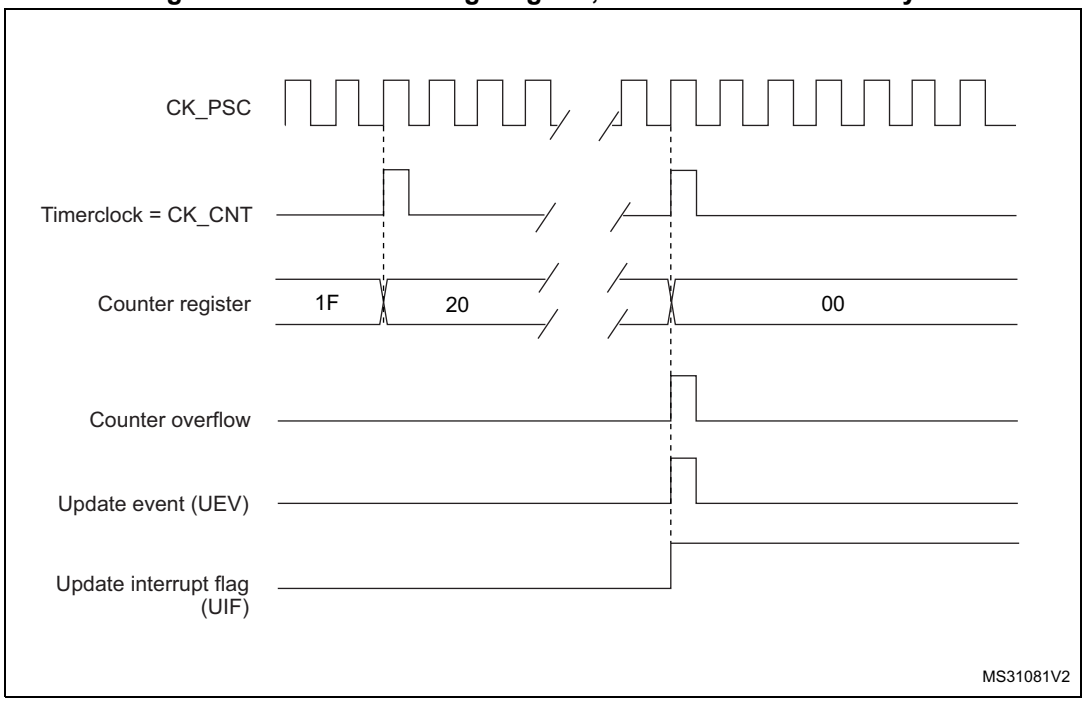



**Figure 182. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)**

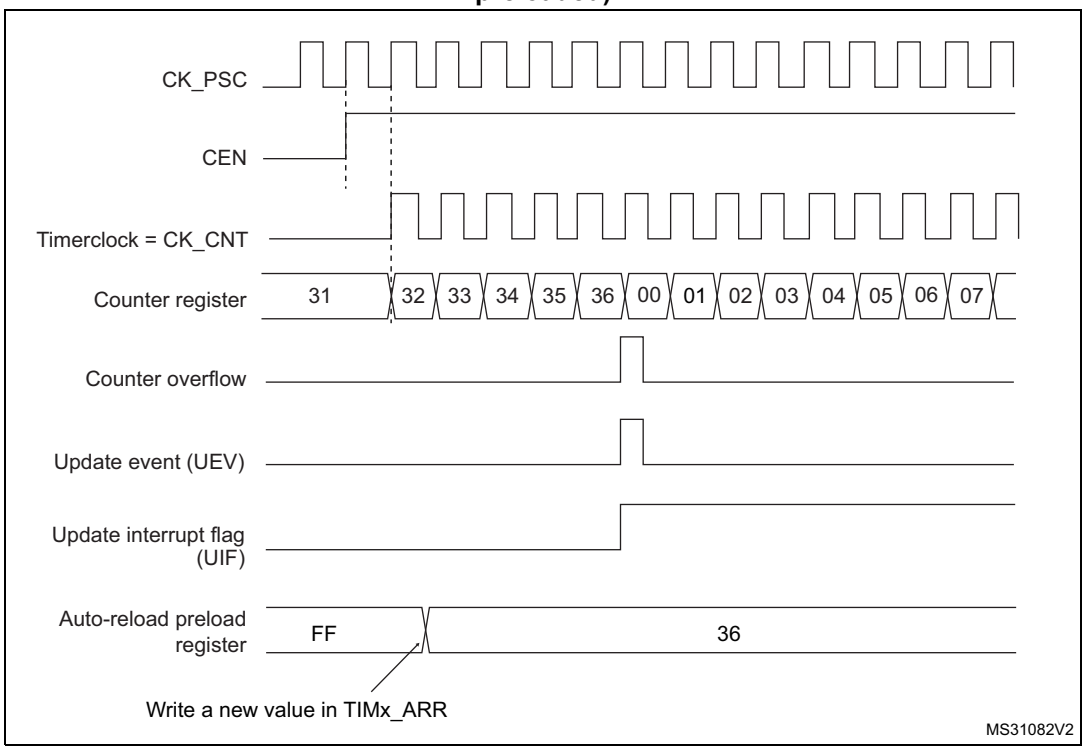
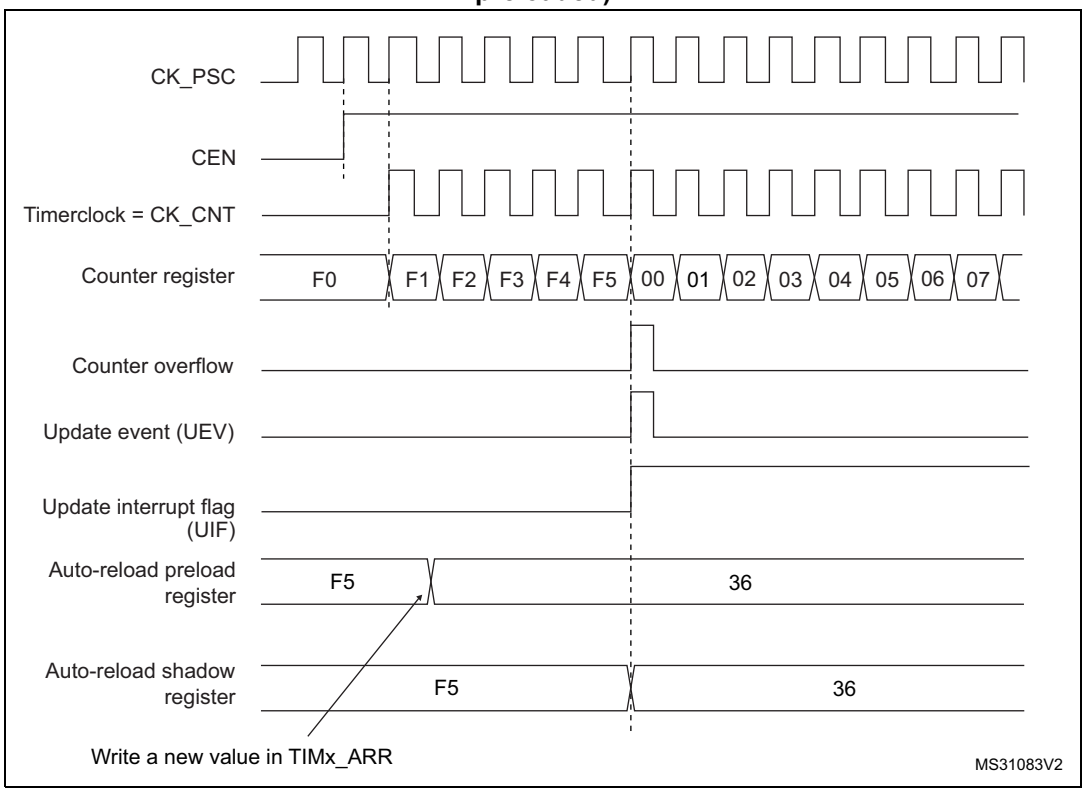


**Figure 183. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)**




## 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.

An Update event can be generate at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

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 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 184. Counter timing diagram, internal clock divided by 1**

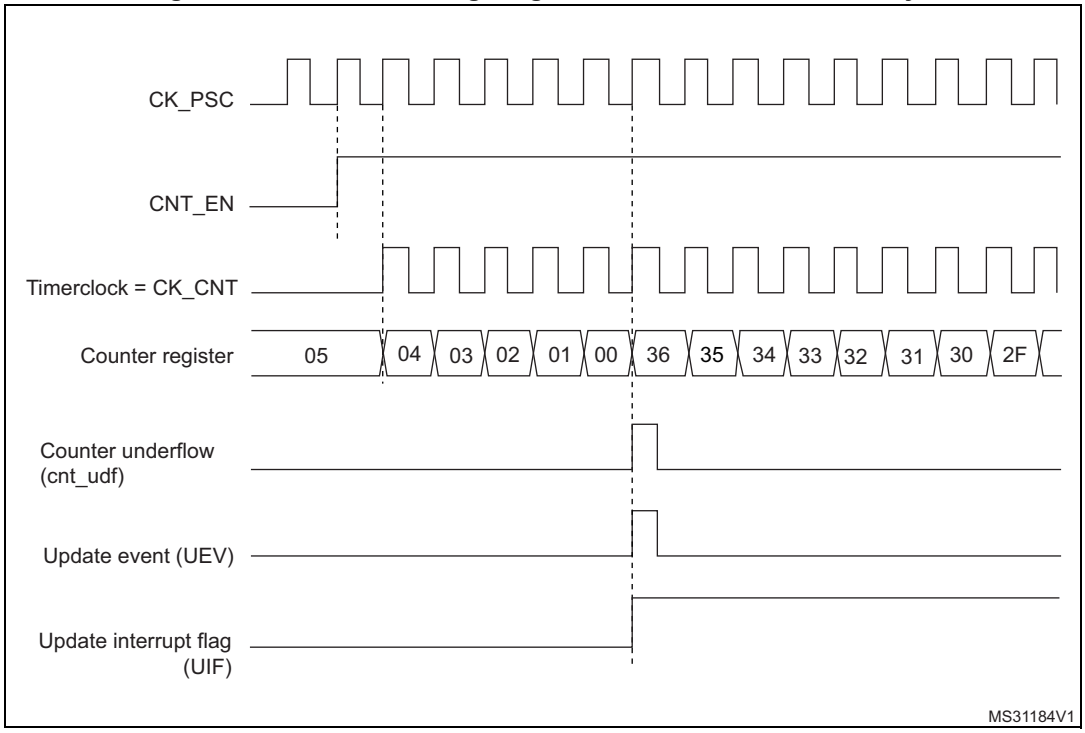



**Figure 185. Counter timing diagram, internal clock divided by 2**

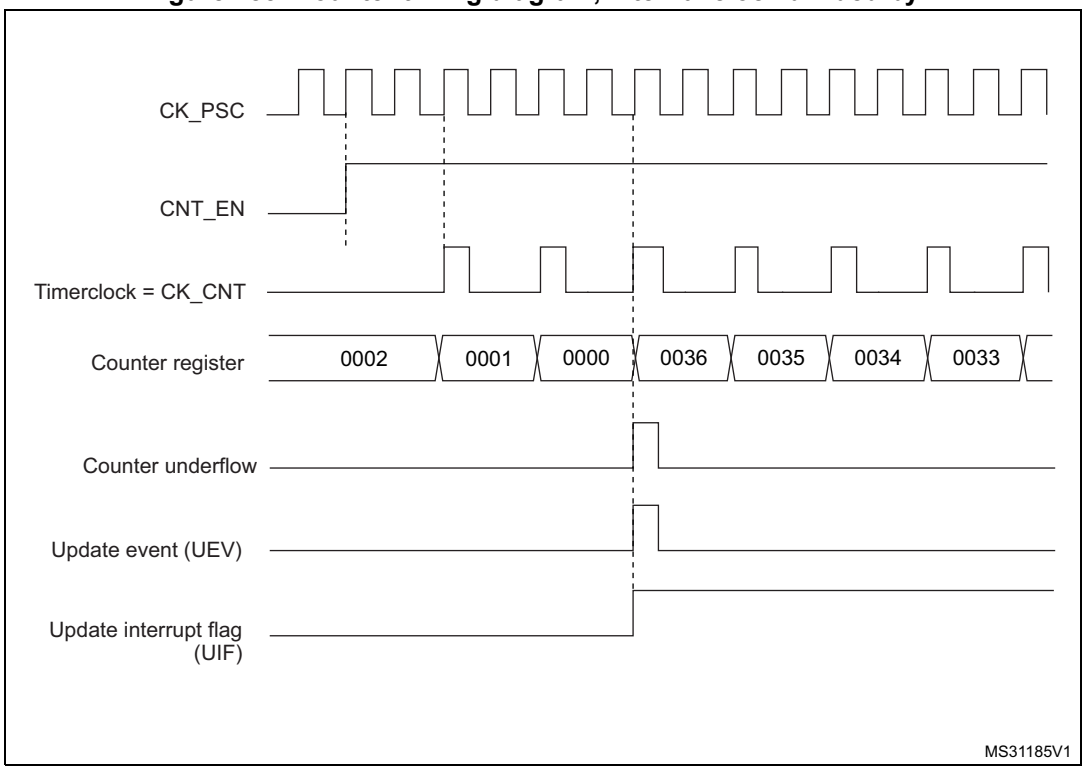


**Figure 186. Counter timing diagram, internal clock divided by 4**

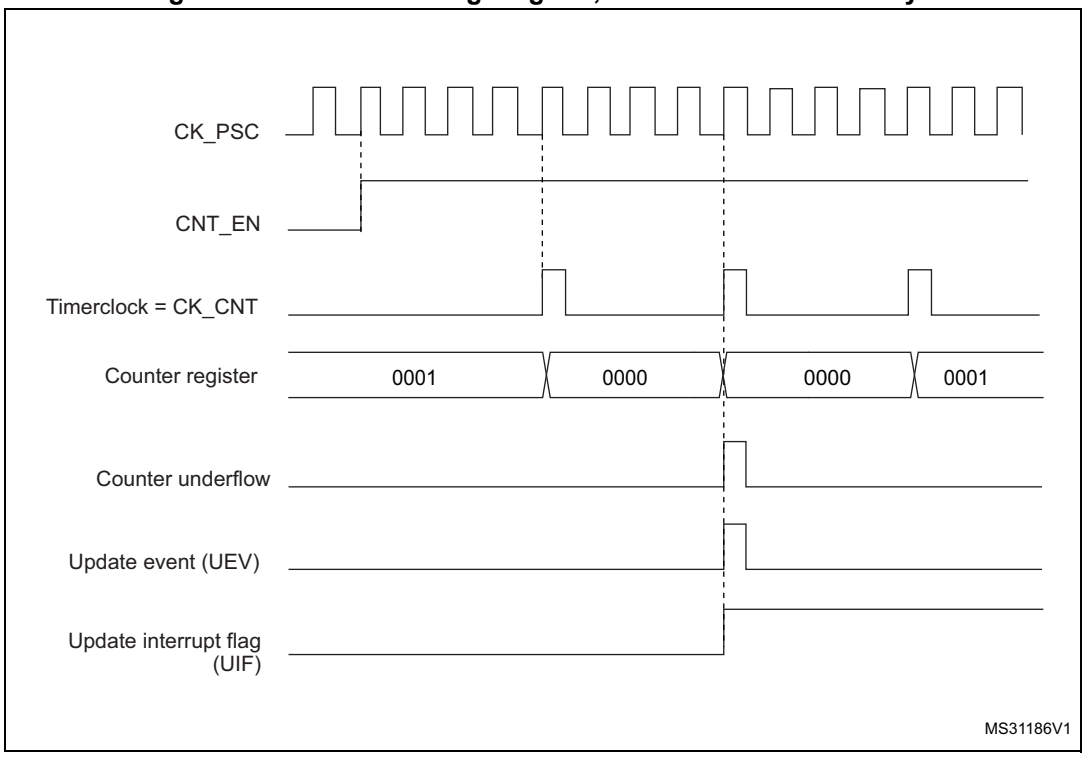



**Figure 187. Counter timing diagram, internal clock divided by N**

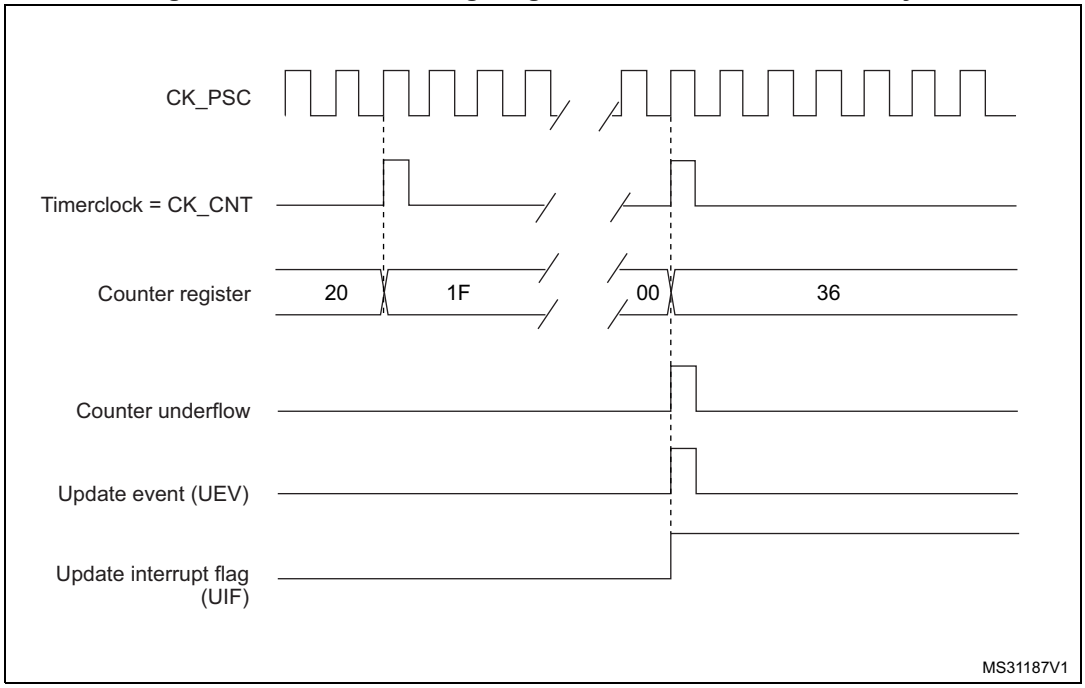
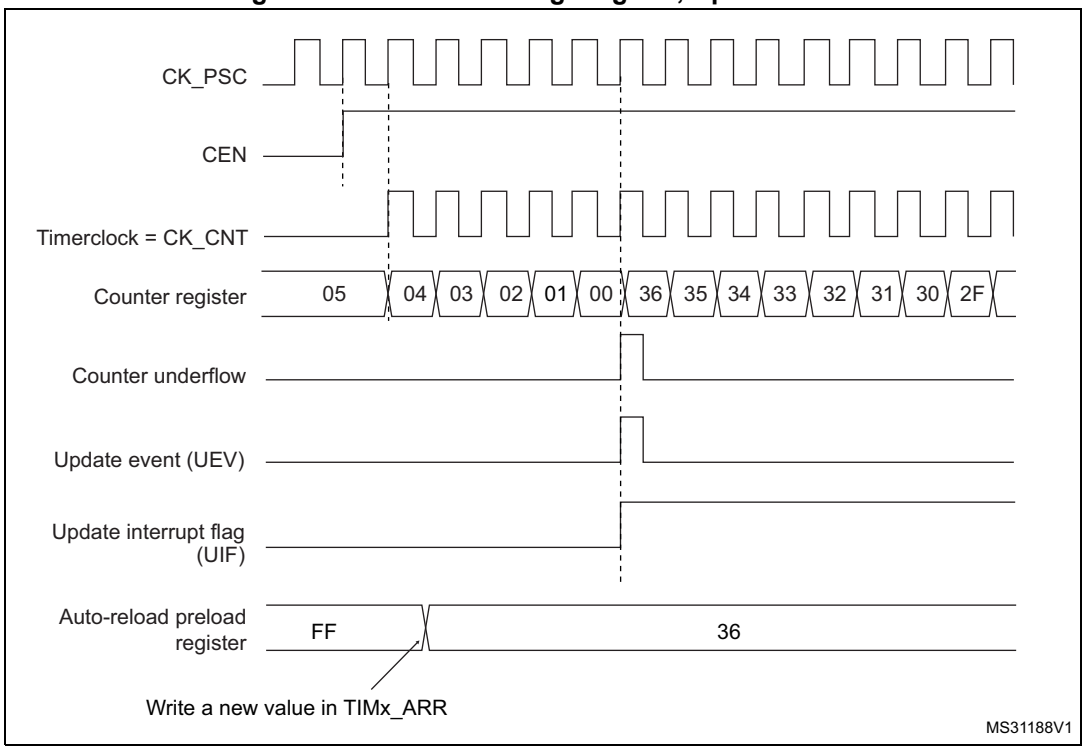


**Figure 188. Counter timing diagram, Update event**




## 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 direction bit (DIR from 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 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 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 update event UEV but without setting the UIF flag (thus no interrupt or

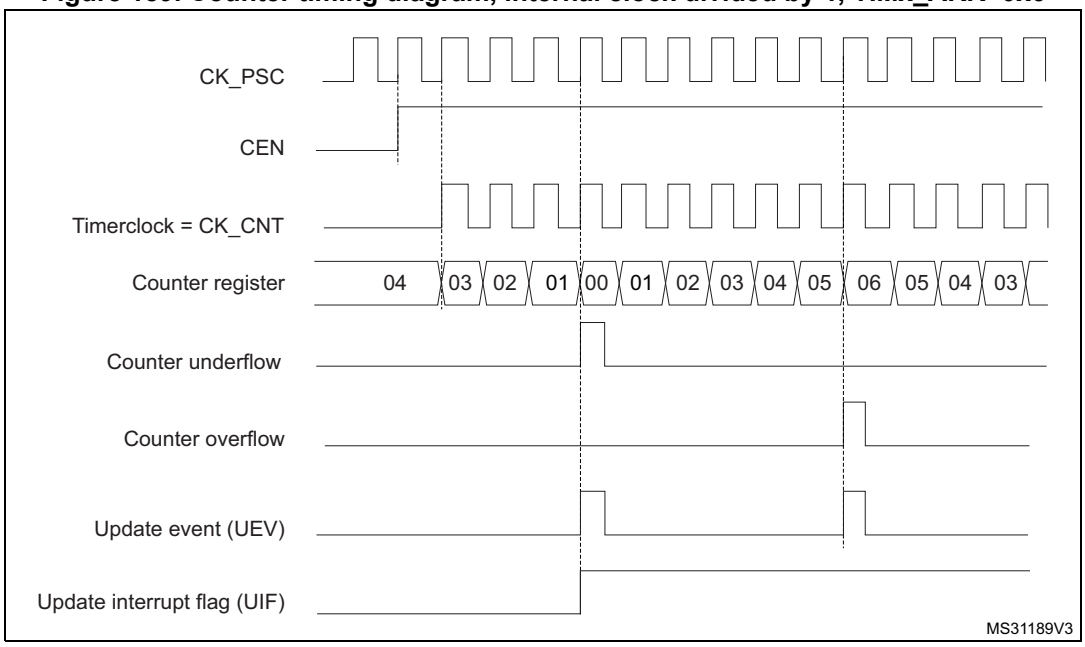
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

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 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 189. 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 20.4.1: TIM2 control register 1 (TIM2_CR1) on page 676*).

**Figure 190. Counter timing diagram, internal clock divided by 2**

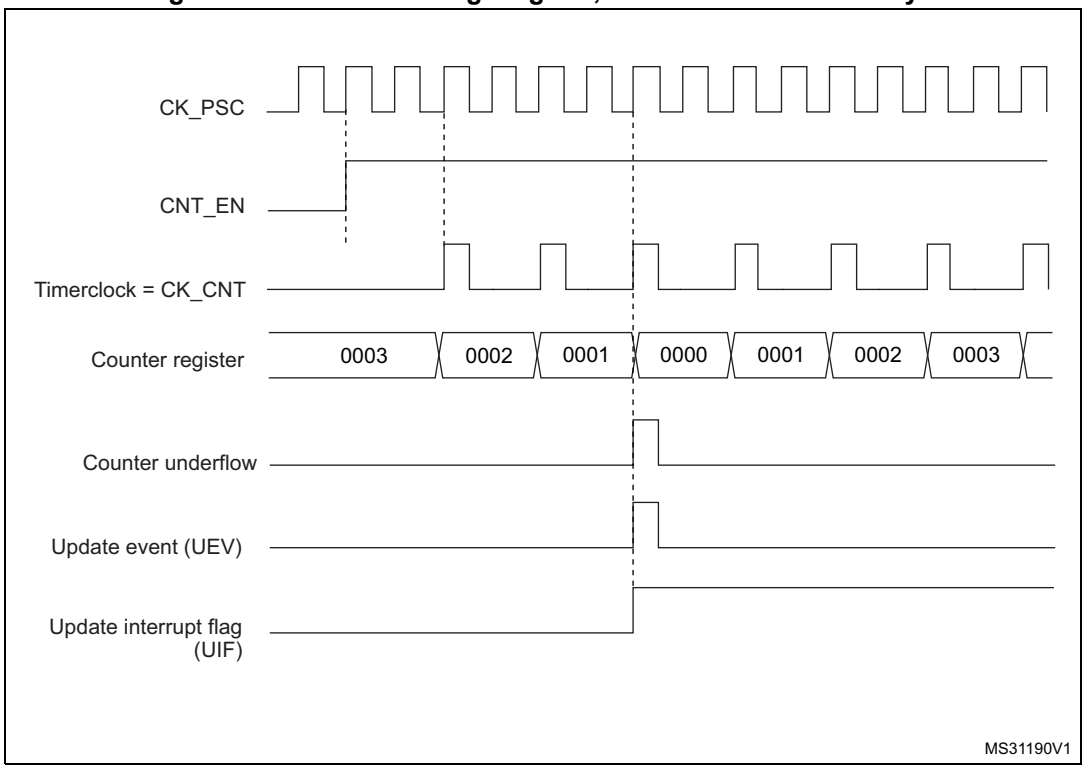
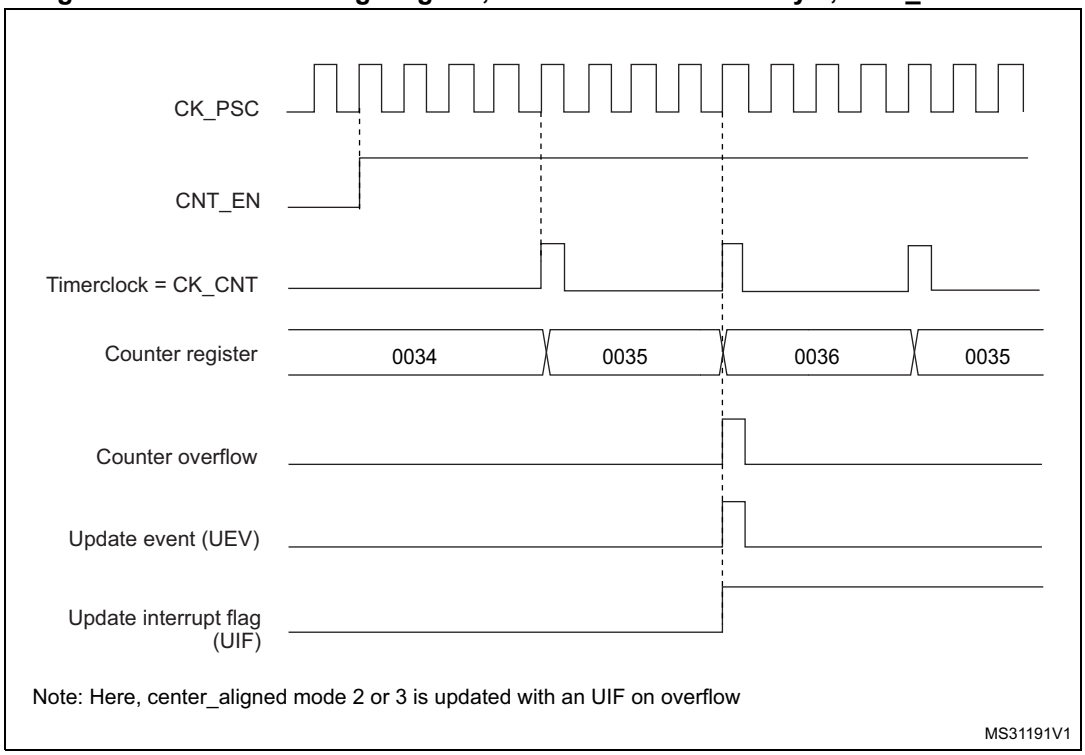



**Figure 191. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36**




1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

**Figure 192. Counter timing diagram, internal clock divided by N**

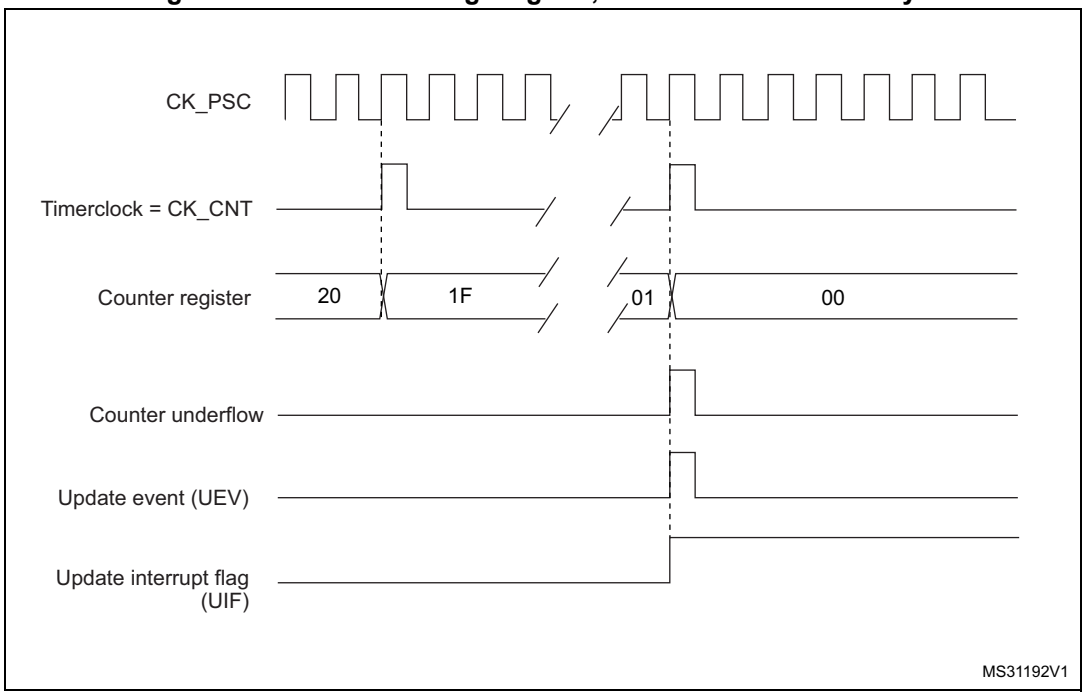



**Figure 193. Counter timing diagram, Update event with ARPE=1 (counter underflow)**

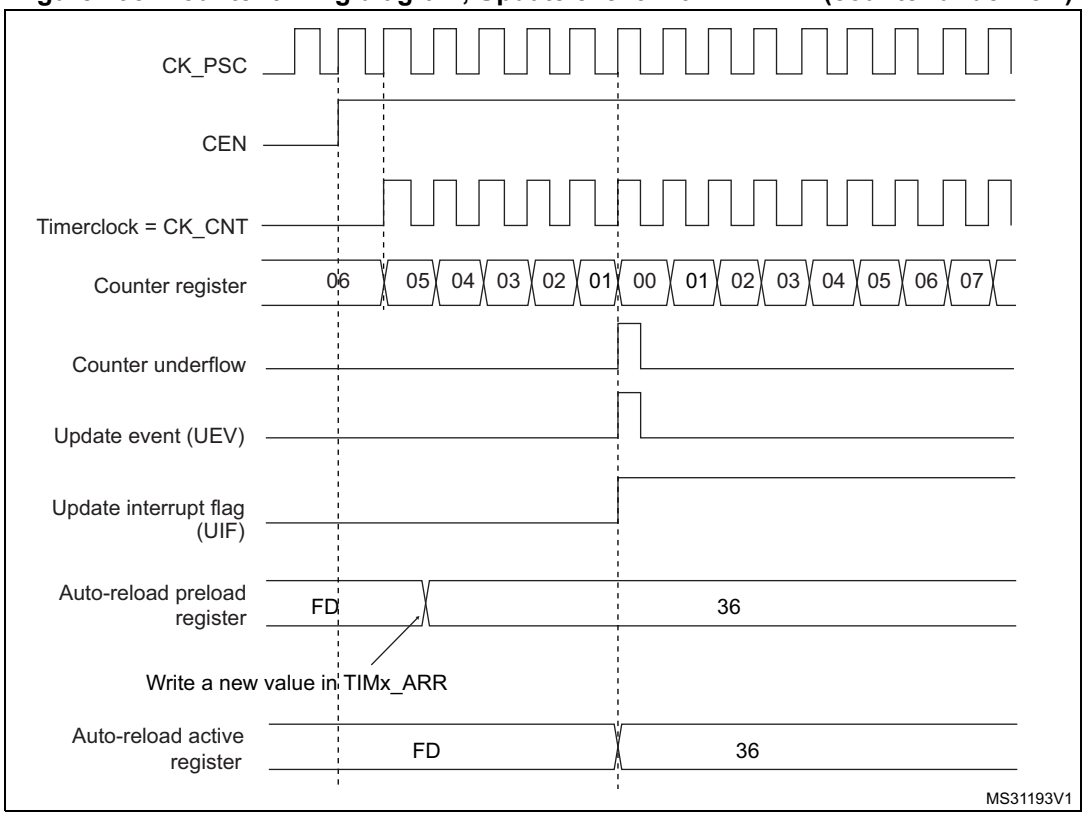
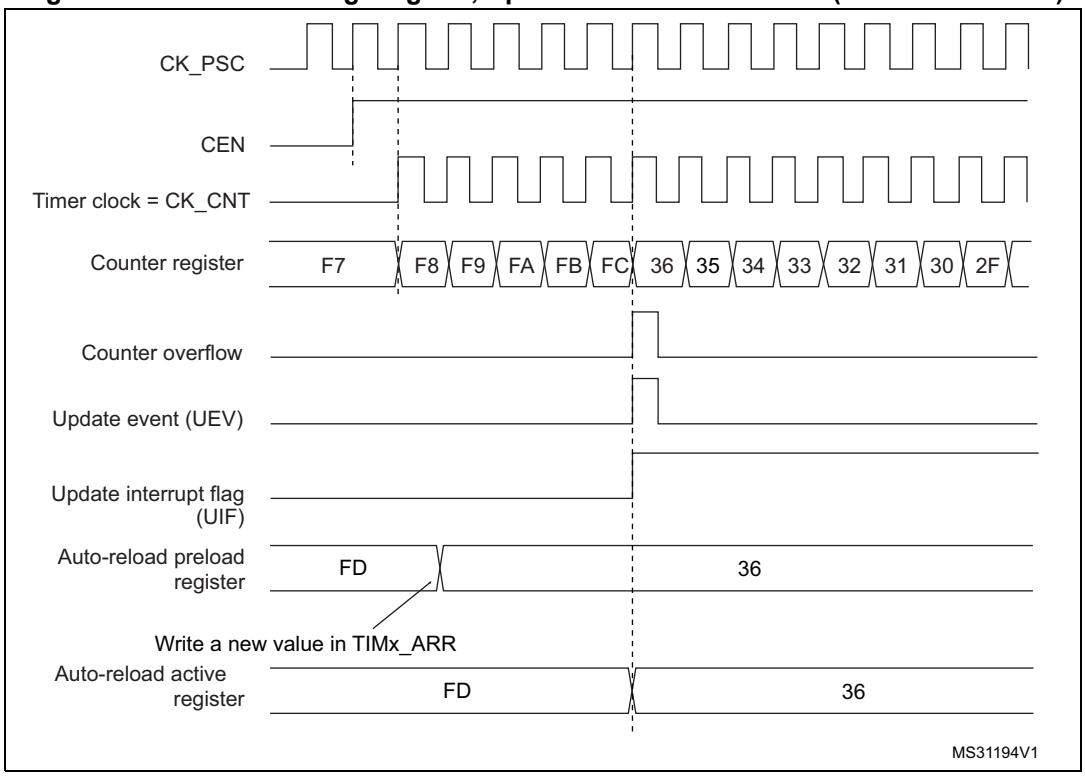


**Figure 194. Counter timing diagram, Update event with ARPE=1 (counter overflow)**




## 20.3.3 Clock selection

The counter clock can be provided by the following clock sources:

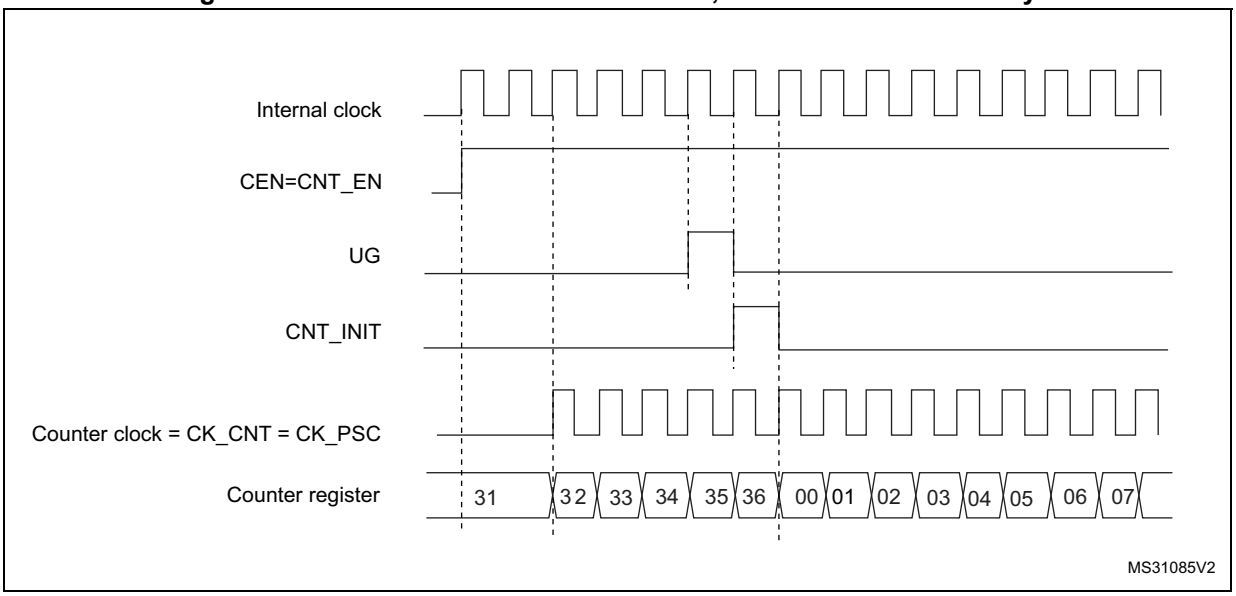
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to *: Using one timer as prescaler for another timer on page 671* for more details.

### Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), 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 195* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

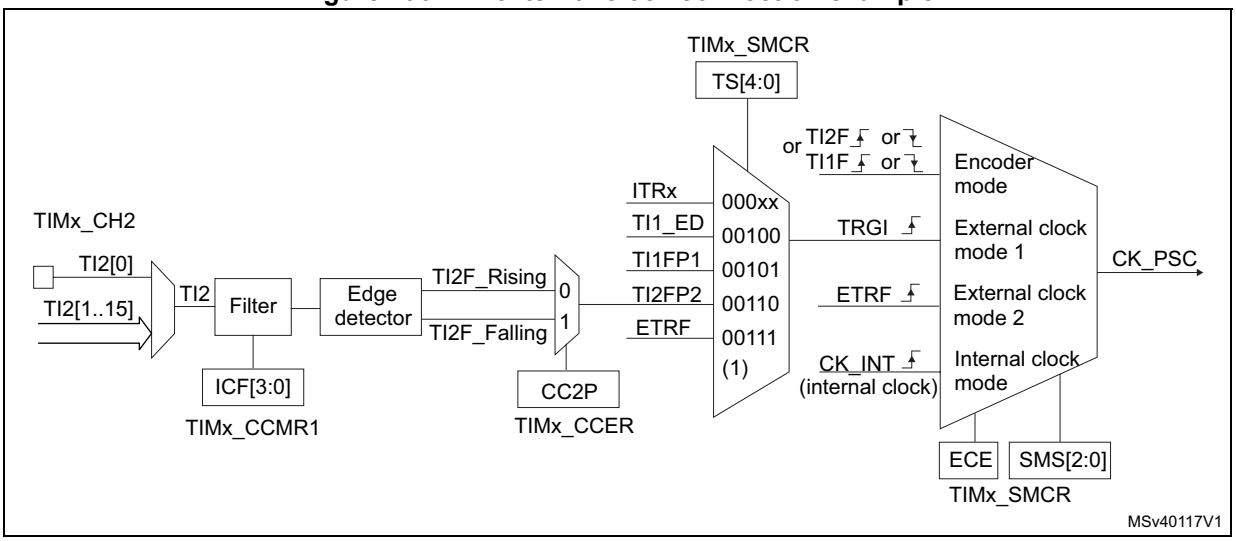
**Figure 195. Control circuit in normal mode, internal clock divided by 1**




## 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 196. TI2 external clock connection example**




1. Codes ranging from 01000 to 11111: ITRy.

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).

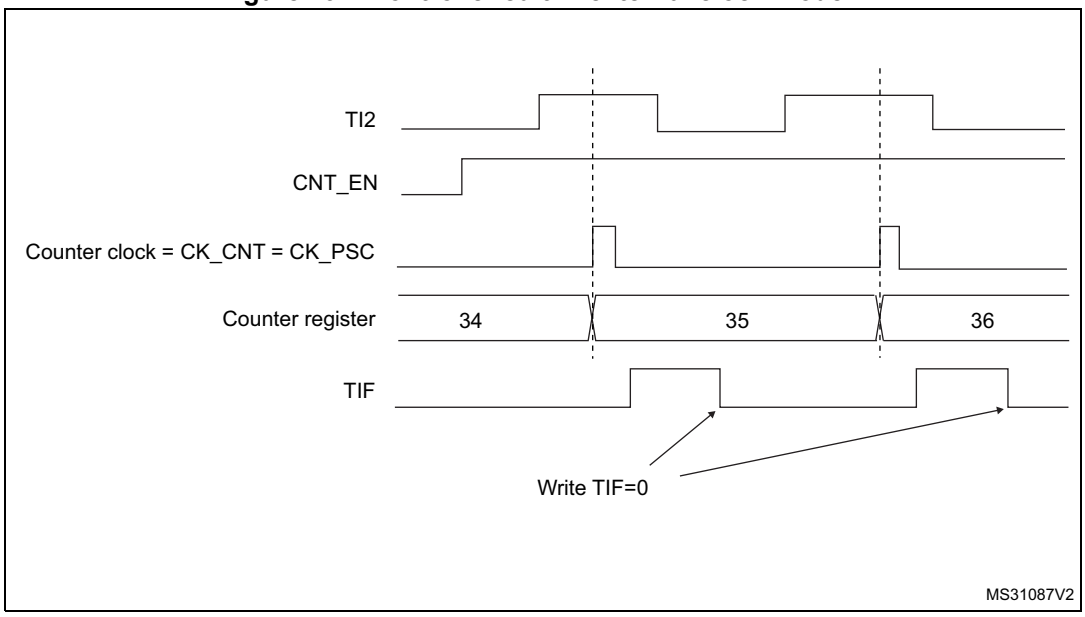
*Note: The capture prescaler is not used for triggering, so it does not need to be configured.*

4. Select rising edge polarity by writing CC2P=0 and CC2NP=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 input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

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 197. Control circuit in external clock mode 1**



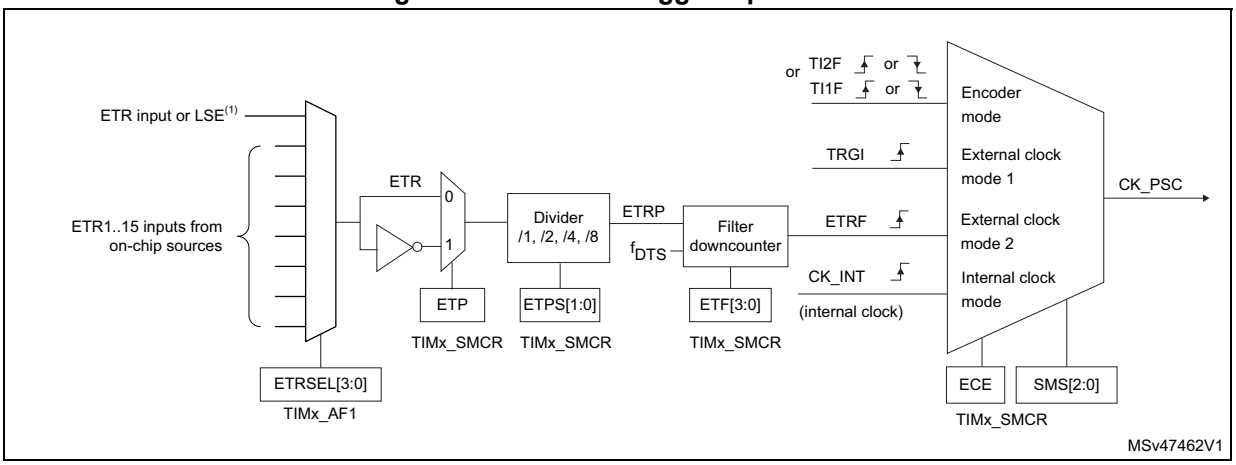

## External clock source mode 2

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.

*Figure 198* gives an overview of the external trigger input block.

**Figure 198. External trigger input block**




1. As per ETR_RMP bit programming.

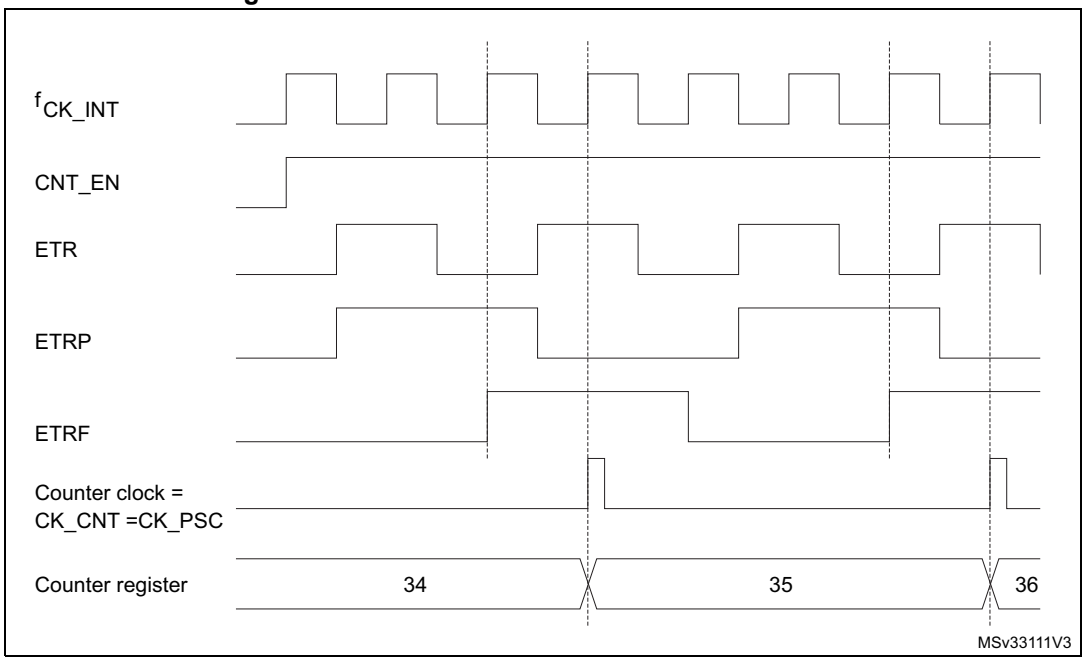
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register and the ETR_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. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.

**Figure 199. Control circuit in external clock mode 2**



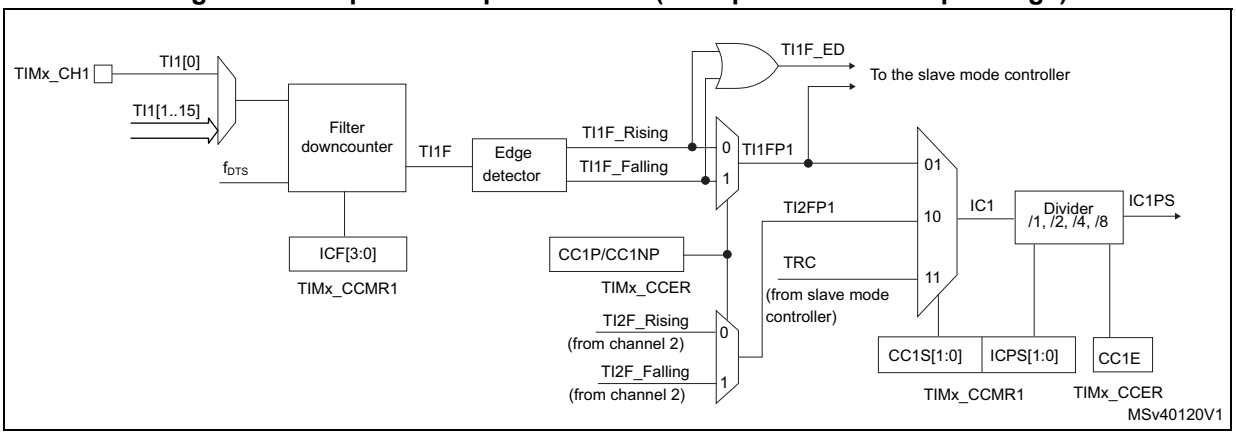

## 20.3.4 Capture/Compare channels

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).

The following figure gives 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 200. 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 201. Capture/Compare channel 1 main circuit**

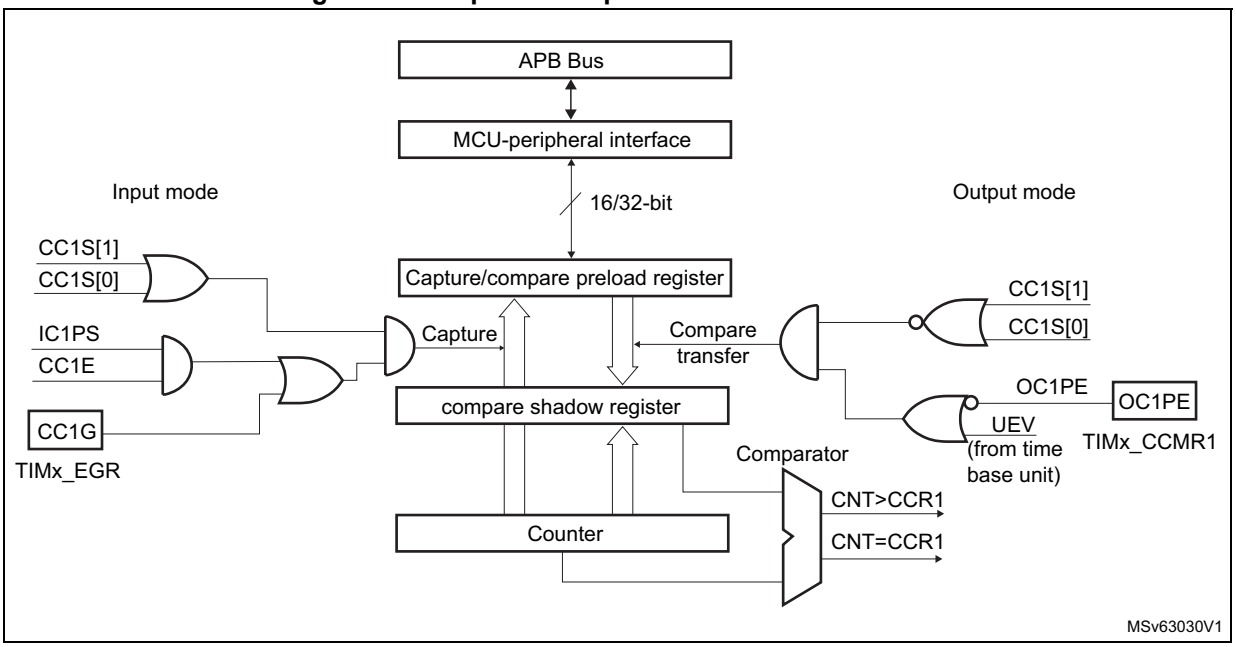
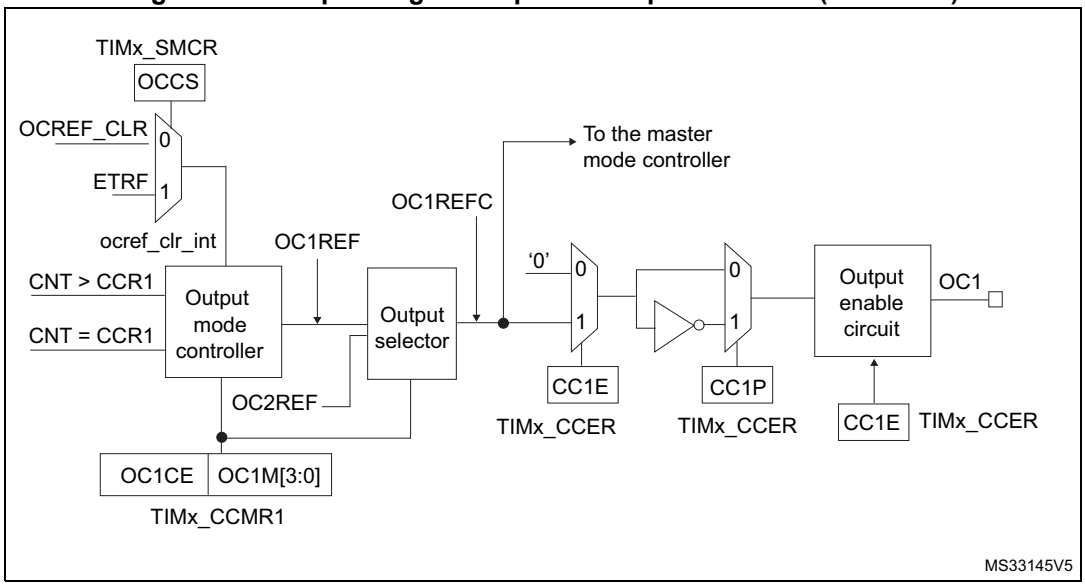



**Figure 202. Output stage of Capture/Compare channel (channel 1)**




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.

### 20.3.5 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 it is written with 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 proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. 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.
3. Program the appropriate input filter duration in relation with the signal connected 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.
4. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. 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).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. 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.*

### 20.3.6 PWM input mode

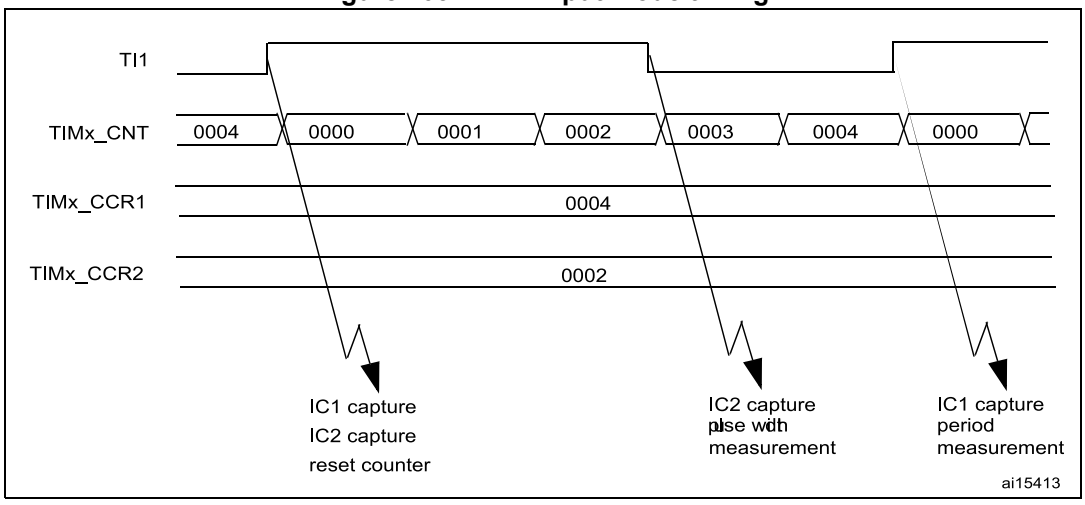
This mode is a particular case of input capture mode. The procedure is the same except:

- 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, one 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):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ‘0’ and the CC1NP bit to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to ‘1’ and the CC2NP bit to ’0’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1 in the TIMx_CCER register.

**Figure 203. PWM input mode timing**




1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

### 20.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) 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, one just needs 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.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

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.

### 20.3.8 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.

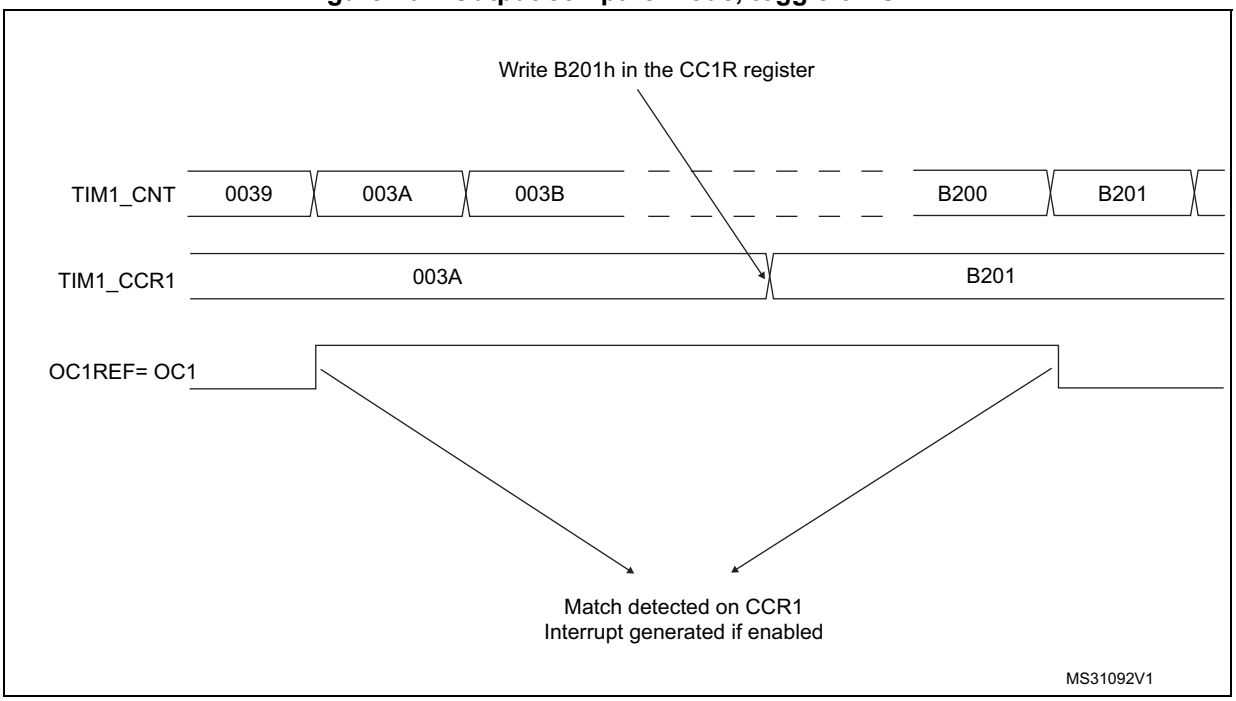
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

#### 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 and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, one must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
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 204*.

**Figure 204. Output compare mode, toggle on OC1**




## 20.3.9 PWM mode

Pulse width modulation mode permits 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. The corresponding preload register must be enabled 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, all registers must be initialized 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 the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx 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). However, to comply with the OCREF_CLR functionality (OCREF can be

cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111).

This forces the PWM by software while the timer is running.

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.

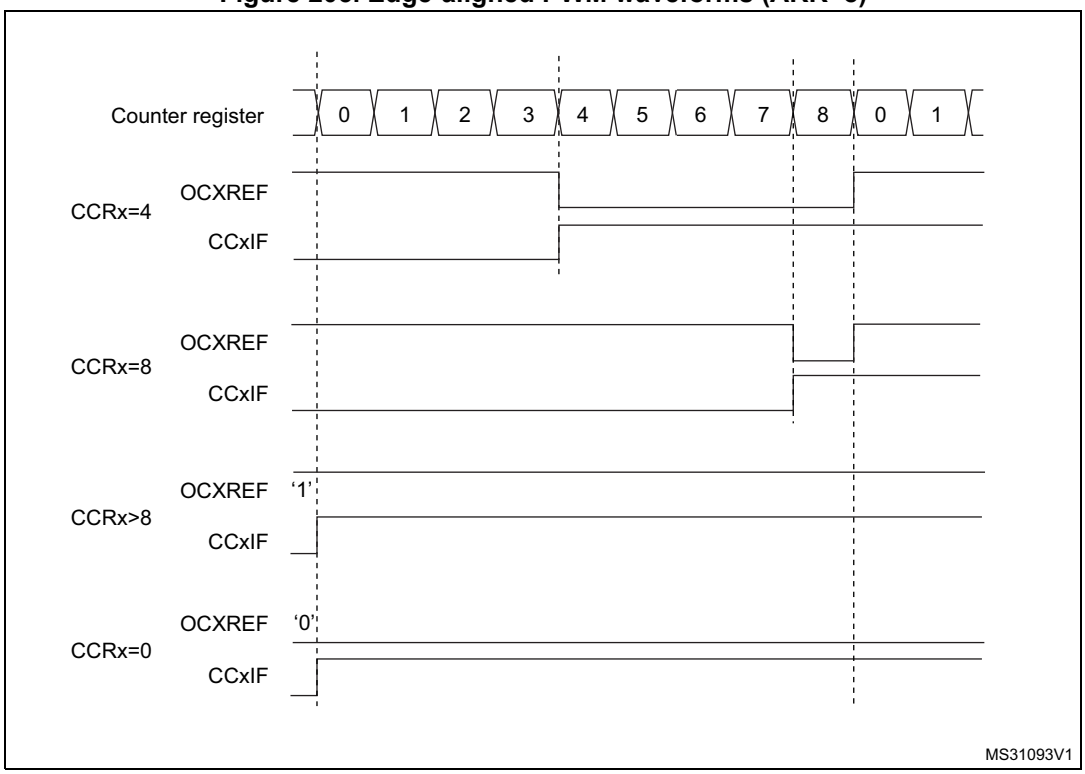
### PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 636*.

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 205* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

**Figure 205. Edge-aligned PWM waveforms (ARR=8)**



### Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode on page 639*.

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 100%. PWM is not possible in this mode.

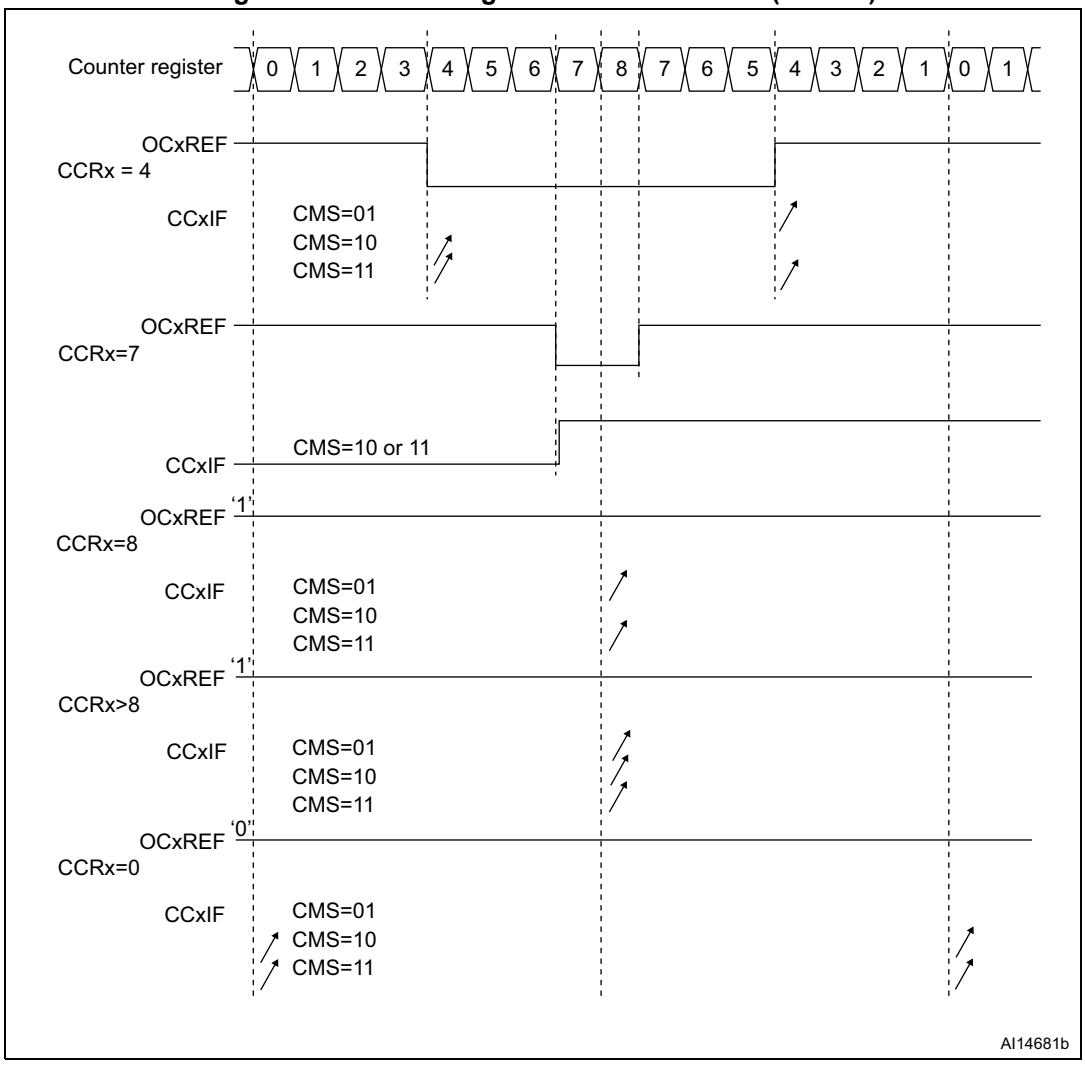
### PWM center-aligned mode

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 *Center-aligned mode (up/down counting) on page 642*.

*Figure 206* 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 206. 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 a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written 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.

### 20.3.10 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 registers. 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

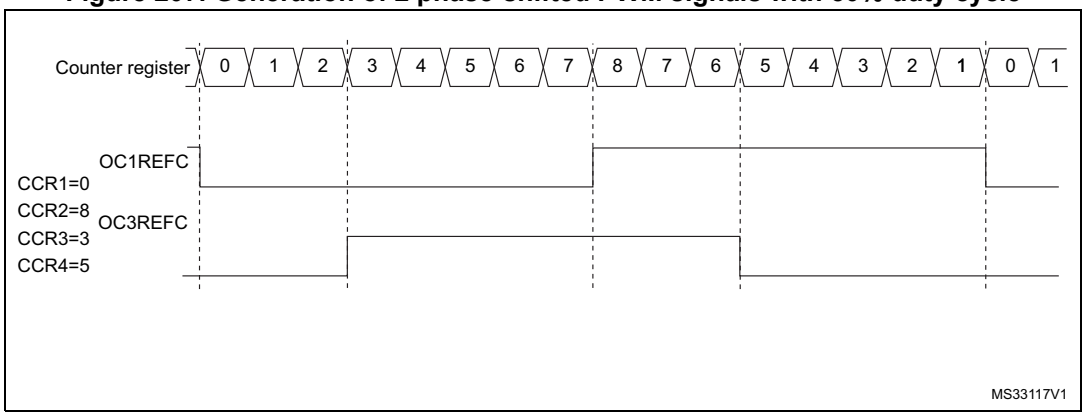
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

*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 secondary 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 2.

*Figure 207* shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

**Figure 207. Generation of 2 phase-shifted PWM signals with 50% duty cycle**




### 20.3.11 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.

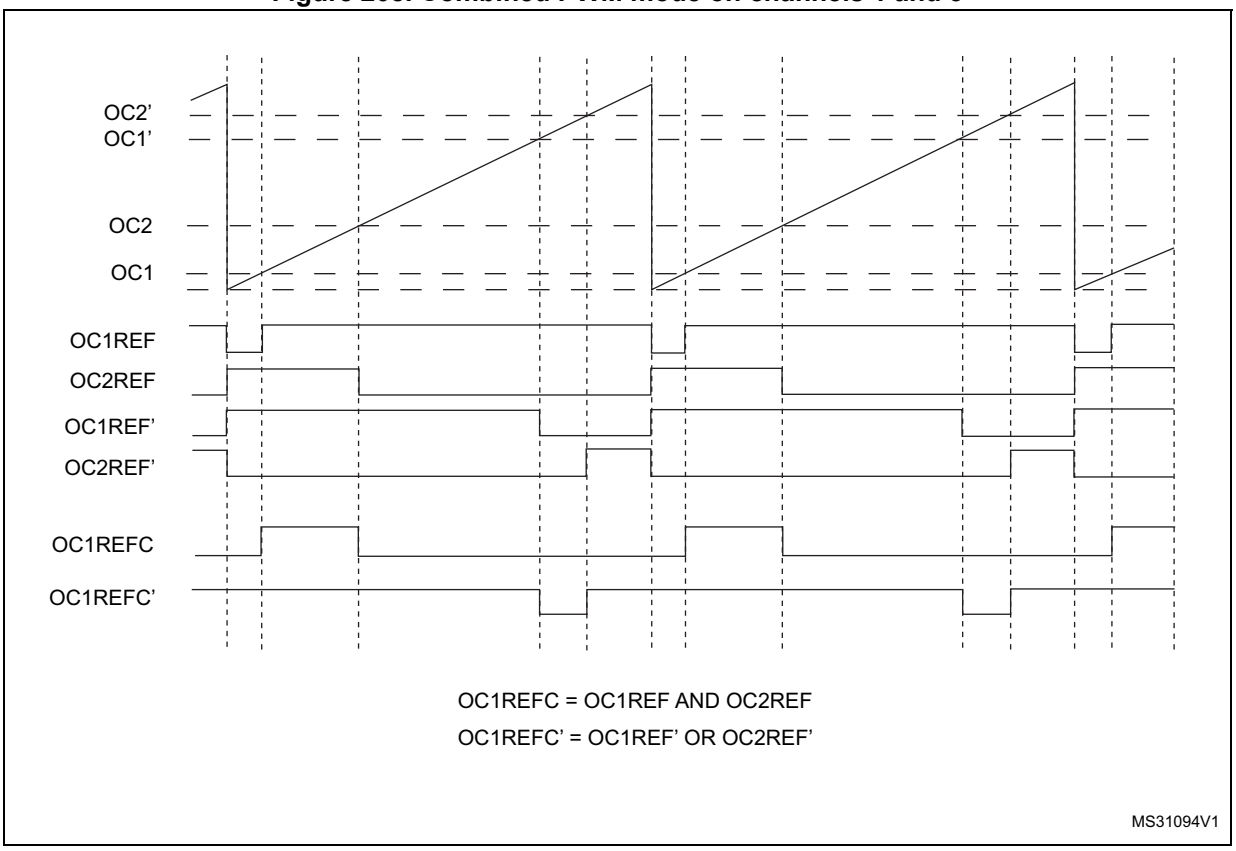
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

*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 208* shows 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 208. Combined PWM mode on channels 1 and 3**




## 20.3.12 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 ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

OCREF_CLR_INPUT can be selected between the OCREF_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next transition to the active state, on the following PWM cycle.

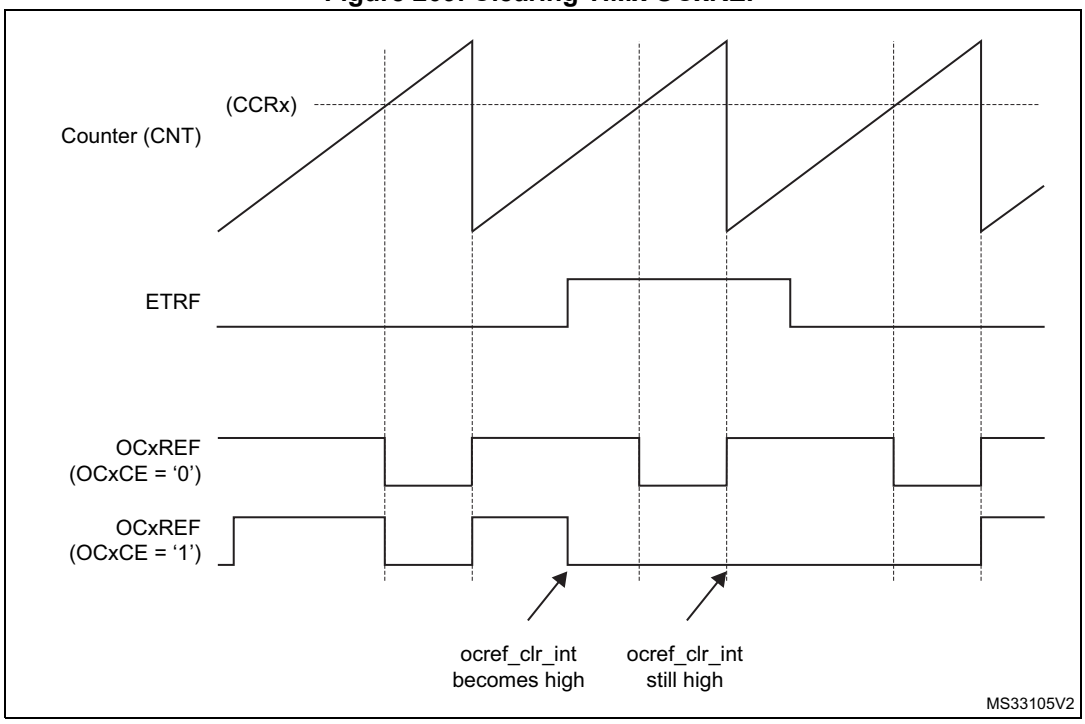
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

*Figure 209* shows the behavior of the OCxREF signal when the ETRF input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

**Figure 209. Clearing TIMx OCxREF**




*Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.*

### 20.3.13 One-pulse mode

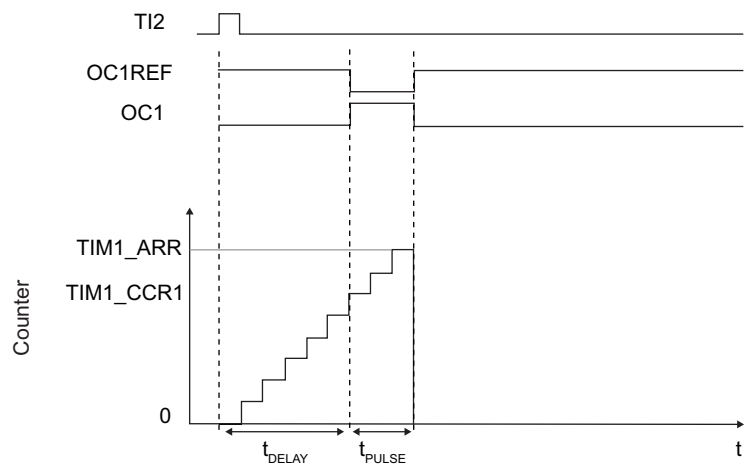
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected 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:

- CNT<CCRx ≤ ARR (in particular, 0<CCRx),

**Figure 210. Example of one-pulse mode.**




For example one 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:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. 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 one 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 PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has 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.

Since only 1 pulse (Single mode) is needed, a 1 must be written 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 one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) is 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.

### 20.3.14 Retriggerable one pulse mode

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 20.3.13*:

- 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 Retriggerable OPM mode 1 or 2.

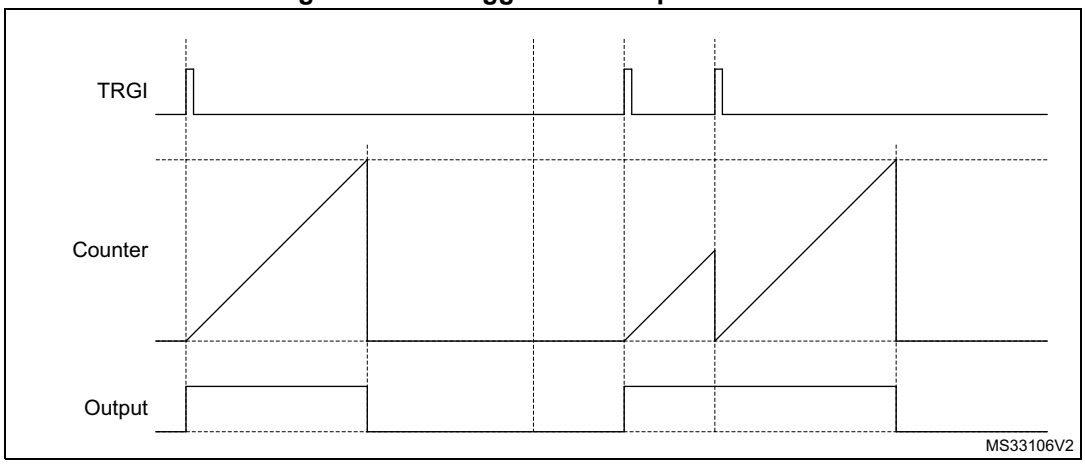
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

*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 211. Retriggerable one-pulse mode.**




## 20.3.15 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. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 131*. 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 the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

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 do not switch at the same time.

**Table 131. 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 |

An external incremental 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.

*Figure 212* 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_CCMR1 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = '0' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = '0' (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

**Figure 212. Example of counter operation in encoder interface mode**

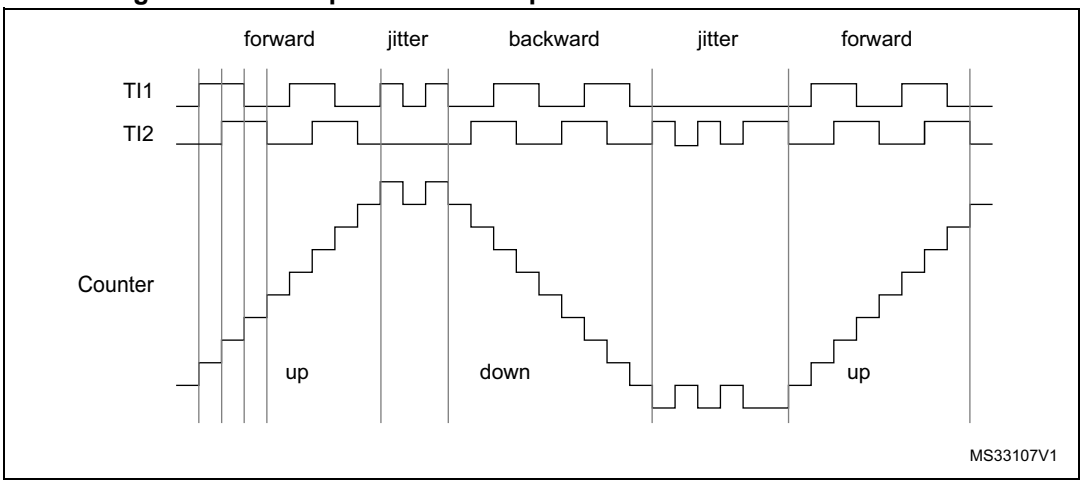
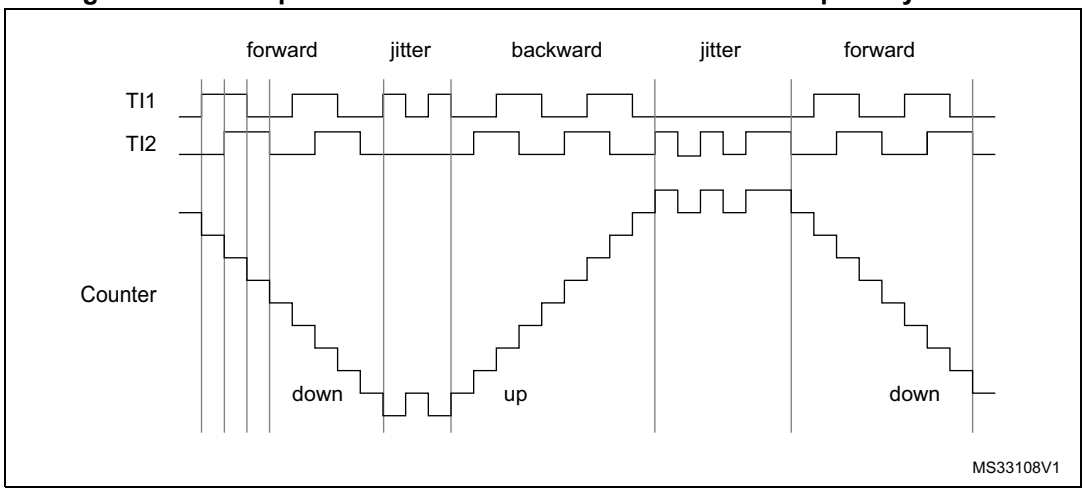



*Figure 213* gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

**Figure 213. 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. Dynamic information can be obtained (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. This can be done 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). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

### 20.3.16 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's bit 31 (TIMxCNT[31]). This permits to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. 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).

### 20.3.17 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 19.3.25: Interfacing with Hall sensors on page 584*.

### 20.3.18 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.

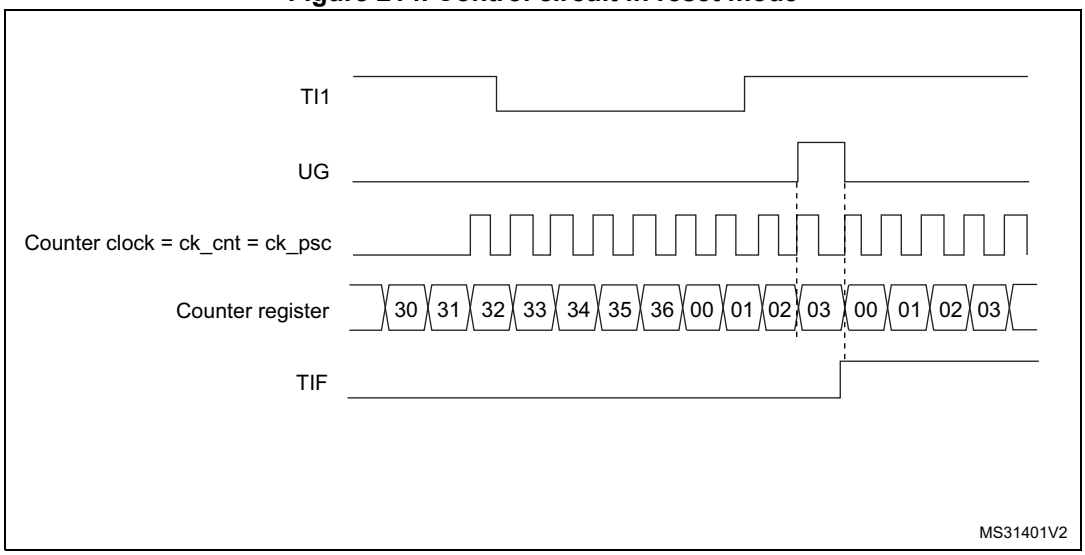
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

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 214. Control circuit in reset mode**




#### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

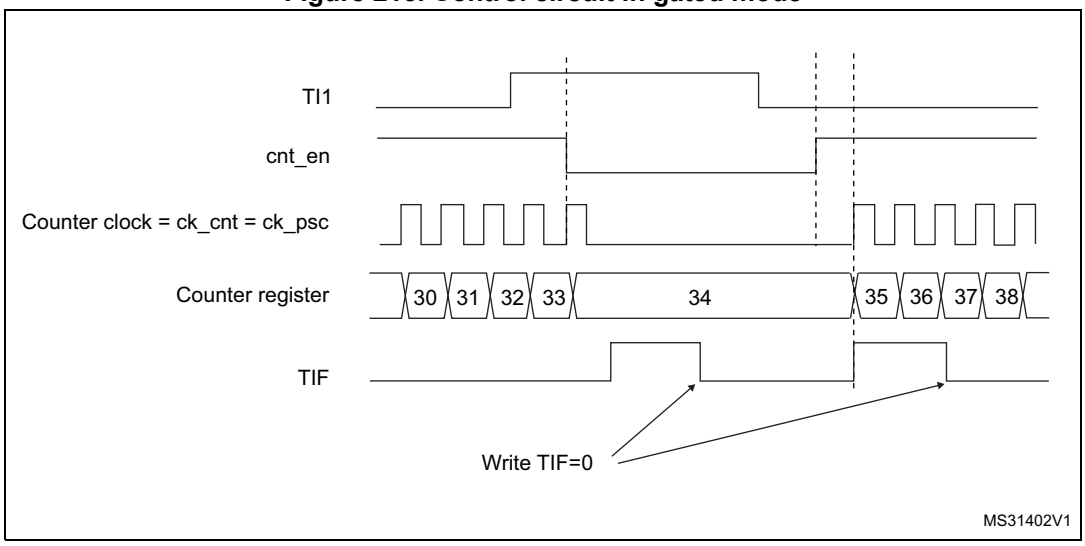
In the following example, the upcounter counts only when TI1 input is low:

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 215. 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

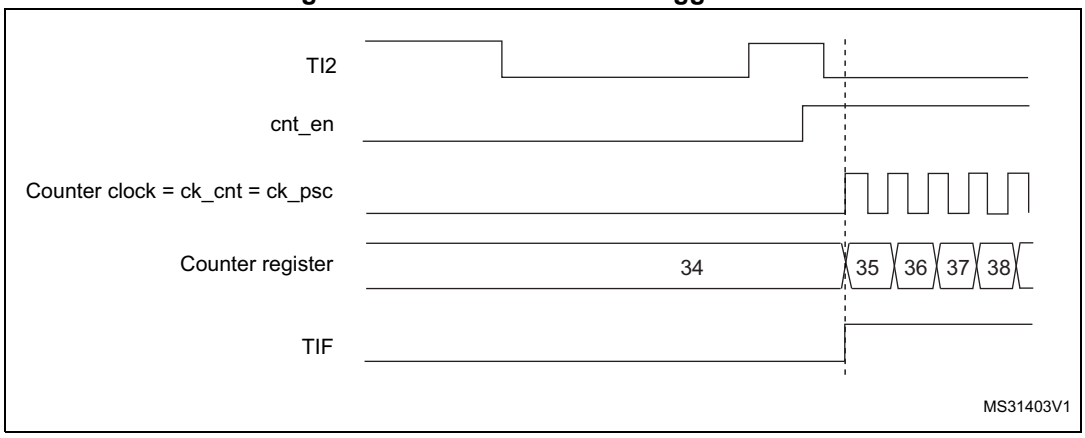
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

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 216. 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.

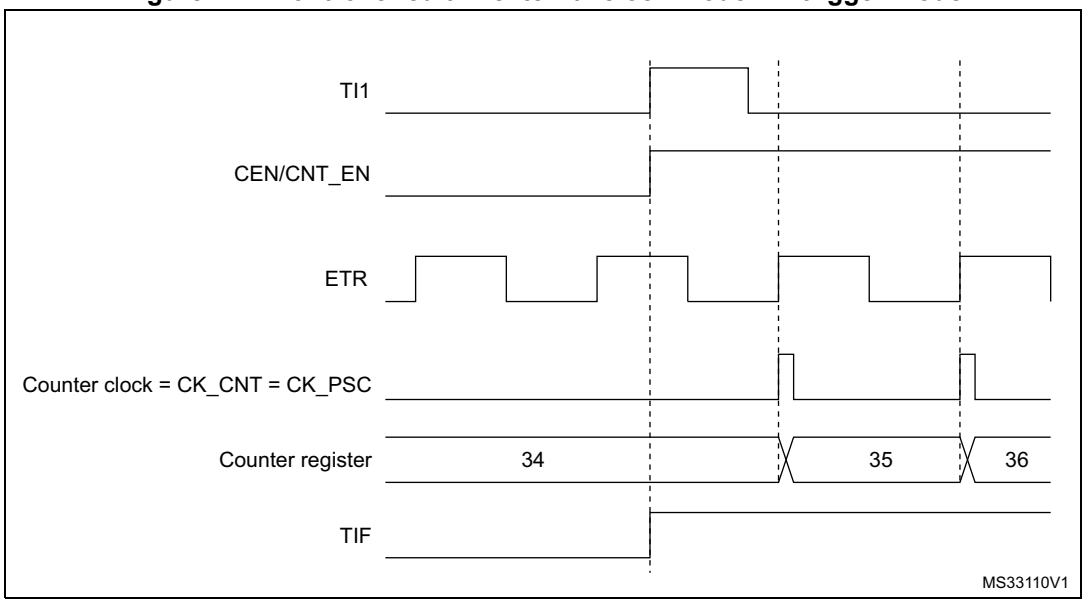
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

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=00101 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 217. Control circuit in external clock mode 2 + trigger mode**




## 20.3.19 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 218: Master/Slave timer example* and *Figure 219: Master/slave connection example with 1 channel only timers* present an overview of the trigger selection and the master mode selection blocks.

**Figure 218. Master/Slave timer example**

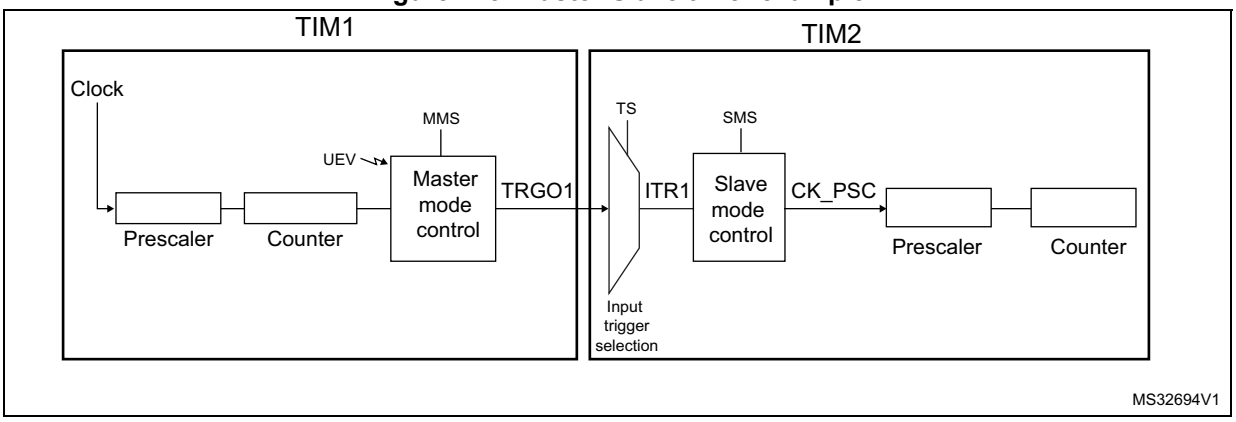
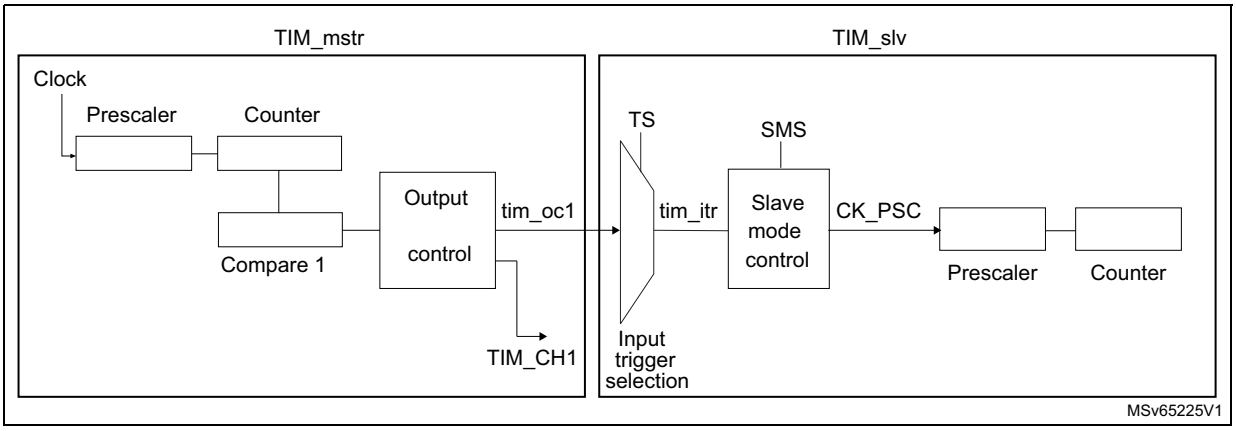


**Figure 219. Master/slave connection example with 1 channel only timers**




*Note: The timers with one channel only (see Figure 219) do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the "TIMx internal trigger connection" table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.*
*The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.*
*For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.*

### Using one timer as prescaler for another timer

For example, TIM1 can be configured to act as a prescaler for TIM2. Refer to *Figure 218*. To do this:

1. Configure TIM1 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM1_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM1 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 TIM1 trigger signal (which correspond to the TIM1 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

*Note: If OCx is selected on TIM1 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.*

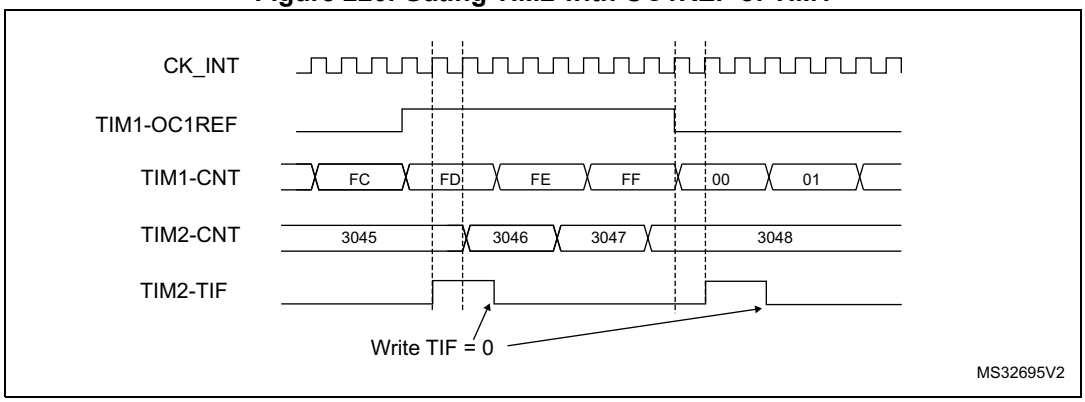
### Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 1. Refer to *Figure 218* for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM1 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 TIM1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
2. Configure the TIM1 OC1REF waveform (TIM1_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM1 (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 (TIM_CR1 register).
6. Start TIM by writing ‘1 in the CEN bit (TIM1_CR1 register).

*Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.*

**Figure 220. Gating TIM2 with OC1REF of TIM1**

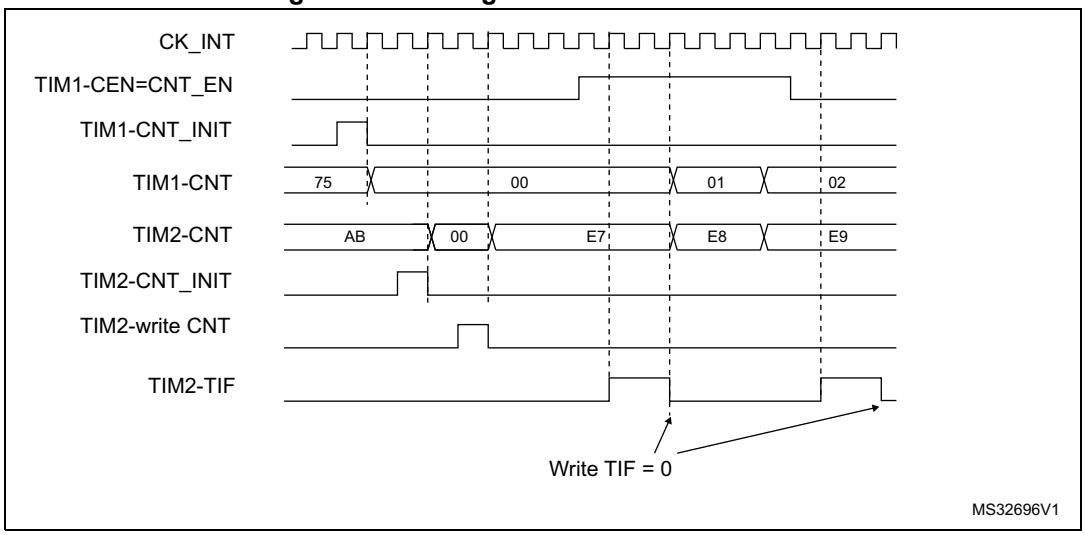



In the example in *Figure 220*, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM1. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 221*), we synchronize TIM1 and TIM2. TIM1 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 TIM1 is disabled by writing ‘0 to the CEN bit in the TIM1_CR1 register:

1. Configure TIM1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
2. Configure the TIM1 OC1REF waveform (TIM1_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM1 by writing ‘1 in UG bit (TIM1_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 TIM1 by writing ‘1 in the CEN bit (TIM1_CR1 register).
10. Stop TIM1 by writing ‘0 in the CEN bit (TIM1_CR1 register).

**Figure 221. Gating TIM2 with Enable of TIM1**

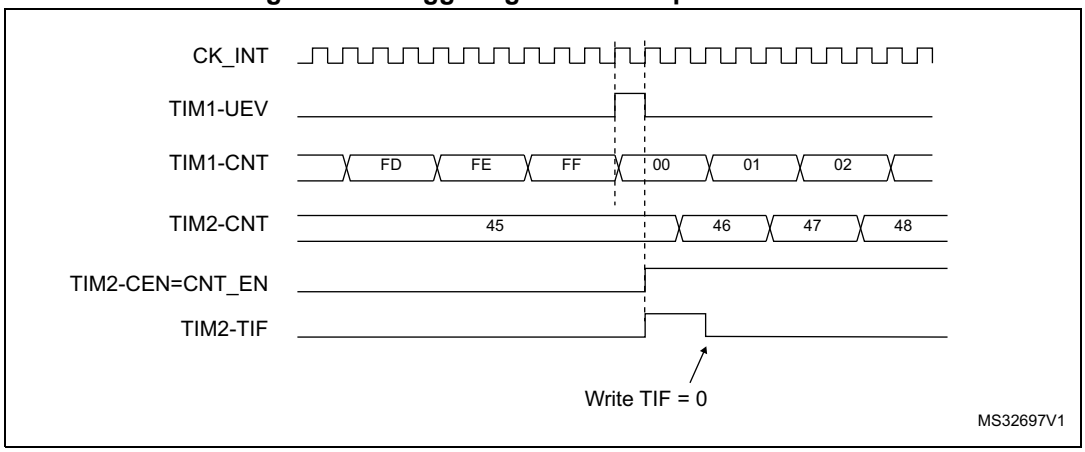



### Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to *Figure 218* 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 1. 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 ($f_{CK_CNT} = f_{CK_INT}/3$).

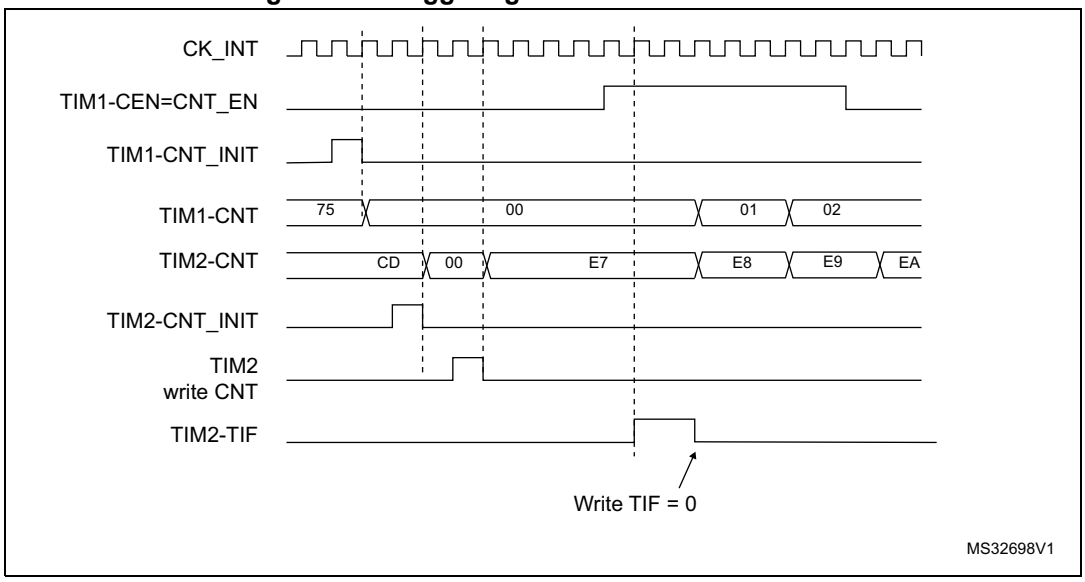
1. Configure TIM1 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM1_CR2 register).
2. Configure the TIM1 period (TIM1_ARR registers).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM1 by writing ‘1 in the CEN bit (TIM1_CR1 register).

**Figure 222. Triggering TIM2 with update of TIM1**



As in the previous example, both counters can be initialized before starting counting. *Figure 223* shows the behavior with the same configuration as in *Figure 222* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

**Figure 223. Triggering TIM2 with Enable of TIM1**




*Note: The clock of the slave peripherals (timer, ADC, ...) 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.*

## 20.3.20 DMA burst mode

The TIMx timers have 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.*

### 20.3.21 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 32.8.3: DBGMCU CPU1 APB1 peripheral freeze register 1 (DBGMCU_APB1FZR1)*.

## 20.4 TIM2 registers

In this section, “TIMx” should be understood as “TIM2” since there is only one instance of this type of timer for the products to which this reference manual applies.

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 20.4.1 TIM2 control register 1 (TIM2_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. | UIFREMAP | 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, 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 sampling clock used by the digital filters (ETR, TIx),

00: $t_{DTS} = t_{CK_INT}$
01: $t_{DTS} = 2 \times t_{CK_INT}$
10: $t_{DTS} = 4 \times t_{CK_INT}$
11: Reserved

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.*

CEN is cleared automatically in one-pulse mode, when an update event occurs.

### 20.4.2 TIM2 control register 2 (TIM2_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. | Res. | Res. | TI1S | MMS[2:0] | | | CCDS | Res. | Res. | Res. |
| | | | | | | | | rw | rw | rw | rw | rw | | | |

Bits 15:8 Reserved, must be kept at reset value.

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)
See also *Section 19.3.25: Interfacing with Hall sensors on page 584*

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits permit 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 AND 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** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

*Note: The clock of the slave timer or ADC 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

Bits 2:0 Reserved, must be kept at reset value.

### 20.4.3 TIM2 slave mode control register (TIM2_SMCR)

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. | 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, must be kept at reset value.

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

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: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).*

*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 00111).*

*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 CK_INT 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 consecutive 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, 6, 5, 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)
01000: Internal Trigger 4 (ITR4)
01001: Internal Trigger 5 (ITR5)
01010: Internal Trigger 6 (ITR6)
01011: Internal Trigger 7 (ITR7)
01100: Internal Trigger 8 (ITR8)
Others: Reserved
See *Table 132: TIM2 internal trigger connection on page 682* 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
1: OCREF_CLR_INT is connected to ETRF

Bits 16, 2, 1, 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.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 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.

*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 (timer, ADC, ...) 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 132. TIM2 internal trigger connection**

| Slave TIM | ITR0 | ITR1 | ITR2 - ITR8 |
|---|---|---|---|
| TIM2 | TIM1 | - | - |

## 20.4.4 TIM2 DMA/Interrupt enable register (TIM2_DIER)

Address offset: 0x0C

Reset value: 0x0000

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

Bit 13 Reserved, must be kept at reset value.

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 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 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

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 (TIM2_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. | 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 15:13 Reserved, must be kept at reset value.

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

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG 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, must be kept at reset value.

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
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
**If channel CC1 is configured as output**: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
**If channel CC1 is configured as input**: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

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 and if 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 the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

### 20.4.6 TIM2 event generation register (TIM2_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, must be kept at reset value.

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 Reserved, must be kept at reset value.

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: Re-initialize 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 (TIM2_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). 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.

**Input capture 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IC2F[3:0] | | | | IC2PSC[1:0] | | CC2S[1:0] | | IC1F[3:0] | | | | IC1PSC[1:0] | | CC1S[1:0] | |
| 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 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

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 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 consecutive 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[1:0]**: 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[1:0]**: 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 1 [alternate] (TIM2_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). 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.

**Output compare mode:**

| 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] |
| | | | | | | | rw | | | | | | | | rw |

| 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] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

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
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 16, 6:4 **OC1M[3:0]**: 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 can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

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: Retriggerable 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 inactive 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: Retriggerable 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: 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.*

*Note: The OC1M[3] bit is not contiguous, located in bit 16.*

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.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
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[1:0]**: 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).*

## 20.4.9 TIM2 capture/compare mode register 2 (TIM2_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). 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.

**Input capture 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IC4F[3:0] | | | | IC4PSC[1:0] | | CC4S[1:0] | | IC3F[3:0] | | | | IC3PSC[1:0] | | CC3S[1:0] | |
| 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 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: 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[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: 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).*

### 20.4.10 TIM2 capture/compare mode register 2 [alternate] (TIM2_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). 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.

**Output compare mode:**

| 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] |
| | | | | | | | rw | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OC4CE | OC4M[2:0] | | | OC4PE | OC4FE | CC4S[1:0] | | OC3CE | OC3M[2:0] | | | OC3PE | OC3FE | CC3S[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode

Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: 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 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: 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).*

### 20.4.11 TIM2 capture/compare enable register (TIM2_CCER)

Address offset: 0x20

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CC4NP | Res. | CC4P | CC4E | CC3NP | Res. | CC3P | CC3E | CC2NP | Res. | CC2P | CC2E | CC1NP | Res. | CC1P | CC1E |
| rw | | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | | rw | rw |

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

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 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

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 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

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 **CC1NP**: *Capture/Compare 1 output Polarity.*
**CC1 channel configured as output**: CC1NP must be kept cleared in this case.
**CC1 channel configured as input**: This bit is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
**When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: 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).
CC1NP=0, CC1P=1: 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).
CC1NP=1, CC1P=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), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*
0: Capture mode disabled / OC1 is not active
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**Table 133. Output control bit for standard OCx channels**

| CCxE bit | OCx output state |
|---|---|
| 0 | Output disabled (not driven by the timer: Hi-Z) |
| 1 | Output enabled (tim_ocx = tim_ocxref + Polarity) |

*Note: The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO control and alternate function registers.*

### 20.4.12 TIM2 counter (TIM2_CNT)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- This section is for UIFREMAP = 0
- Next section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[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 |
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **CNT[31:0]**: counter value

## 20.4.13 TIM2 counter [alternate] (TIM2_CNT)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- Previous section is for UIFREMAP = 0
- This section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UIFCPY | CNT[30:16] | | | | | | | | | | | | | | |
| 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 |
| 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

Bits 30:0 **CNT[30:0]**: counter value

## 20.4.14 TIM2 prescaler (TIM2_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”).

### 20.4.15 TIM2 auto-reload register (TIM2_ARR)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARR[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 |
| ARR[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **ARR[31:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 20.3.1: Time-base unit on page 634* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 20.4.16 TIM2 capture/compare register 1 (TIM2_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR1[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 |
| CCR1[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **CCR1[31:0]**: 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 CC1is configured as input**:

CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

### 20.4.17 TIM2 capture/compare register 2 (TIM2_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR2[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 |
| CCR2[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **CCR2[31:0]**: 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_CCMR1 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). The TIMx_CCR2 register is read-only and cannot be programmed.

## 20.4.18 TIM2 capture/compare register 3 (TIM2_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR3[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 |
| CCR3[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **CCR3[31:0]**: 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_CCMR2 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 CC3is configured as input**:
CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

## 20.4.19 TIM2 capture/compare register 4 (TIM2_CCR4)

Address offset: 0x40

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR4[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 |
| CCR4[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **CCR4[31:0]**: Capture/Compare value

1. if CC4 channel is configured as output (CC4S bits):
   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_CCMR2 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 signalled on OC4 output.
2. if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):
   CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

## 20.4.20 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 vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
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 vector 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: TIMx_SMCR
...
**Example**: Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

### 20.4.21 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).

### 20.4.22 TIM2 option register 1 (TIM2_OR1)

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. | TI4_RMP | | ETR_ RMP | ITR_ RMP |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bits 3:2 **TI4_RMP**: Timer input 4 remap

Set and cleared by software.

00: TIM2 TI4 is connected to GPIO: Refer to Alternate Function mapping

Others: Reserved

Bit 1 **ETR_RMP**: External trigger 1 remap

Set and cleared by software.

0: TIM2 ETR is connected to GPIO: Refer to Alternate Function mapping

1: LSE internal clock is connected to TIM2_ETR input

Bit 0 **ITR_RMP**: Internal trigger remap

0: TIM2 Internal trigger ITR2 is not connected

1: Reserved

### 20.4.23 TIM2 alternate function option register 1 (TIM2_AF1)

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. | ETRSEL[3:2] | |
| | | | | | | | | | | | | | | rw | rw |

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

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

Bits 17:14 **ETRSEL[3:0]**: ETR source selection
These bits select the ETR input source.
0000: GPIO or LSE internal clock, as per ETR_RMP bit in TIM2_OR1
Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

### 20.4.24 TIM2 timer input selection register (TIM2_TISEL)

Address offset: 0x68

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. | TI2SEL[3:0] | | | | Res. | Res. | Res. | Res. | TI1SEL[3:0] | | | |
| | | | | rw | rw | rw | rw | | | | | rw | rw | rw | rw |

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM2_CH2 input
Others: Reserved

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

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM2_CH1 input
Others: Reserved

## 20.4.25 TIMx register map

TIMx registers are mapped as described in the table below:

**Table 134. TIM2 register map and reset values**

| Offset | Register name | 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. | 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] | | | Res. | SMS[2:0] | | |
| | Reset value | | | | | | | | | | | 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. | Res. | Res. | CC4DE | CC3DE | CC2DE | CC1DE | UDE | Res. | TIE | Res. | CC4IE | CC3IE | CC2IE | CC1IE | UIE |
| | Reset value | | | | | | | | | | | | | | | | | | | | 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] | O24CE | 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. | CC4NP | Res. | CC4P | CC4E | CC3NP | Res. | CC3P | CC3E | CC2NP | Res. | CC2P | CC2E | CC1NP | Res. | CC1P | CC1E |
| | Reset value | | | | | | | | | | | | | | | | | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 |

**Table 134. TIM2 register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x24 | **TIMx_CNT** | CNT[31] or UIFCPY | CNT[30:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | **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** | ARR[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 |
| 0x30 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x34 | **TIMx_CCR1** | CCR1[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 |
| 0x38 | **TIMx_CCR2** | CCR2[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 |
| 0x3C | **TIMx_CCR3** | CCR3[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 |
| 0x40 | **TIMx_CCR4** | CCR4[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 |
| 0x44 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 | **TIM2_OR1** | Res. | Res. | Res. | Res. | 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 | | ETR_RMP | ITR_RMP |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x60 | **TIM2_AF1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ETRSEL [3:0] | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | |

**Table 134. TIM2 register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x68 | TIM2_TISEL | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TI2SEL[3:0] | | | | Res. | Res. | Res. | Res. | TI1SEL[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 0 |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 21 General-purpose timers (TIM16/TIM17)

## 21.1 TIM16/TIM17 introduction

The TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They 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).

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.

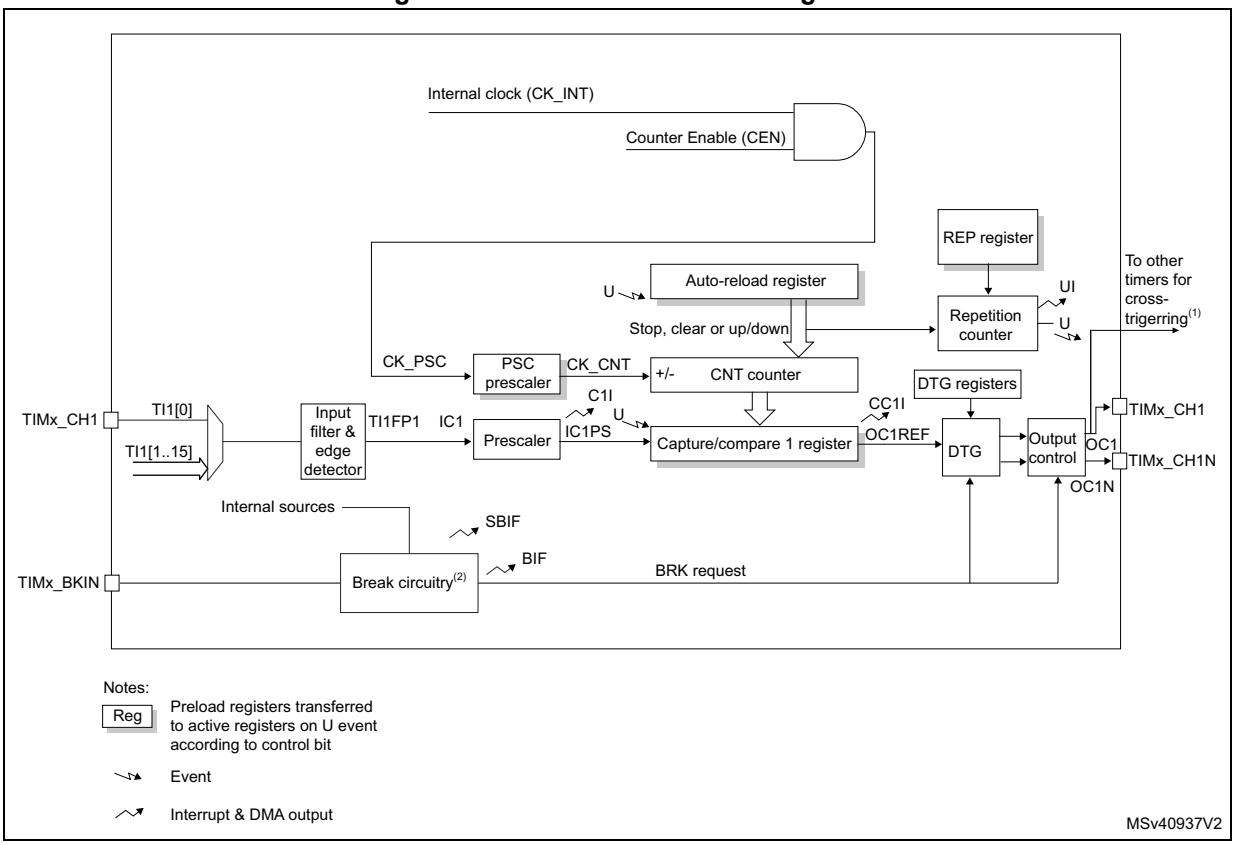
The TIM16/TIM17 timers are completely independent, and do not share any resources.

## 21.2 TIM16/TIM17 main features

The TIM16/TIM17 timers include 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 outputs with programmable dead-time
- 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
  - Input capture
  - Output compare
  - Break input

**Figure 224. TIM16/TIM17 block diagram**




1. This signal can be used as trigger for some slave timer, see *Section 21.3.18: Using timer output as trigger for other timers (TIM16/TIM17)*.

## 21.3 TIM16/TIM17 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 225* and *Figure 226* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 225. Counter timing diagram with prescaler division change from 1 to 2**

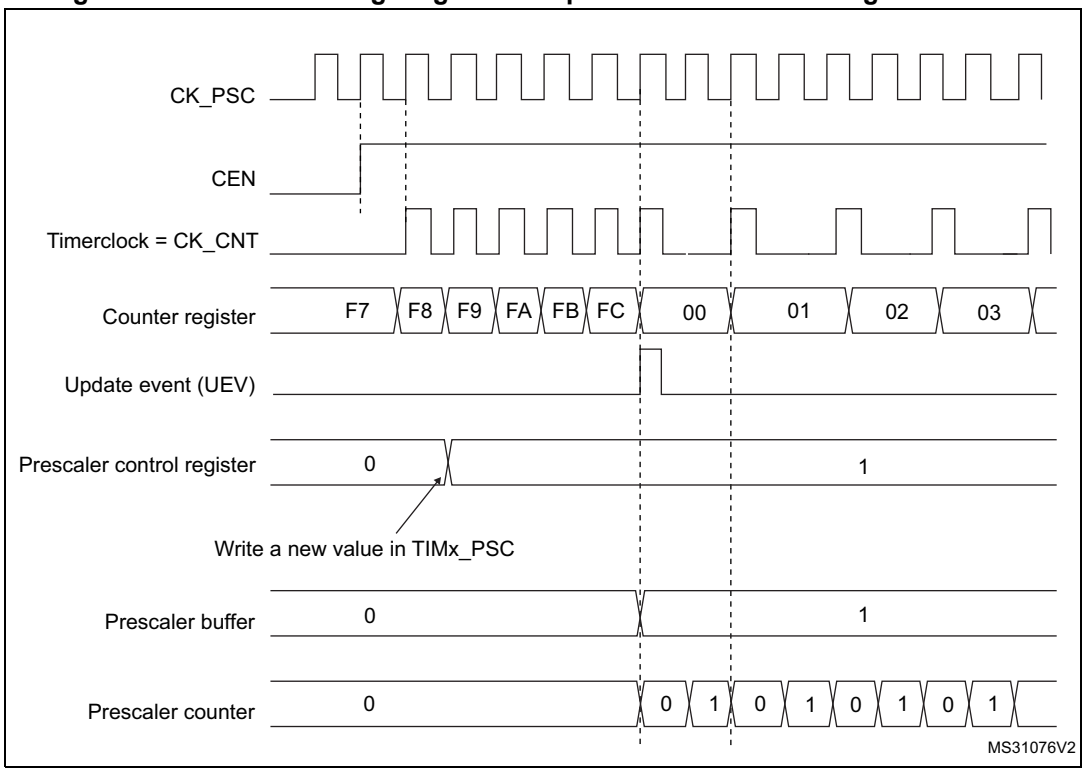
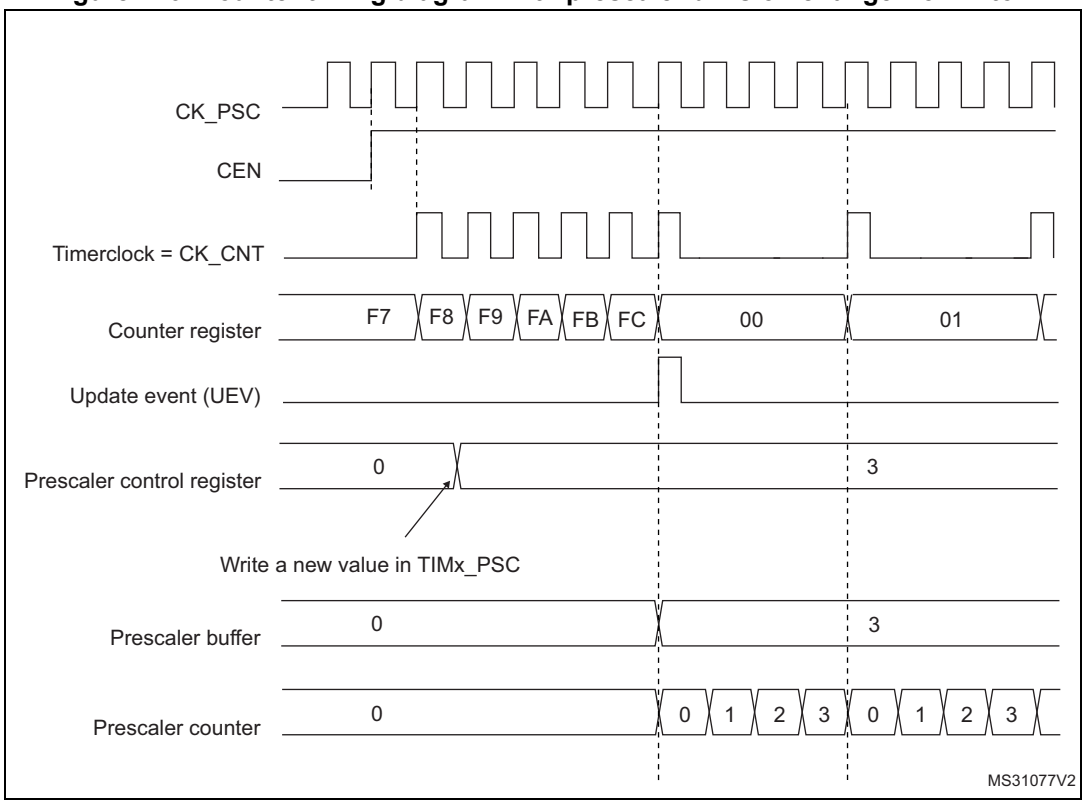



**Figure 226. Counter timing diagram with prescaler division change from 1 to 4**



## 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 227. Counter timing diagram, internal clock divided by 1**

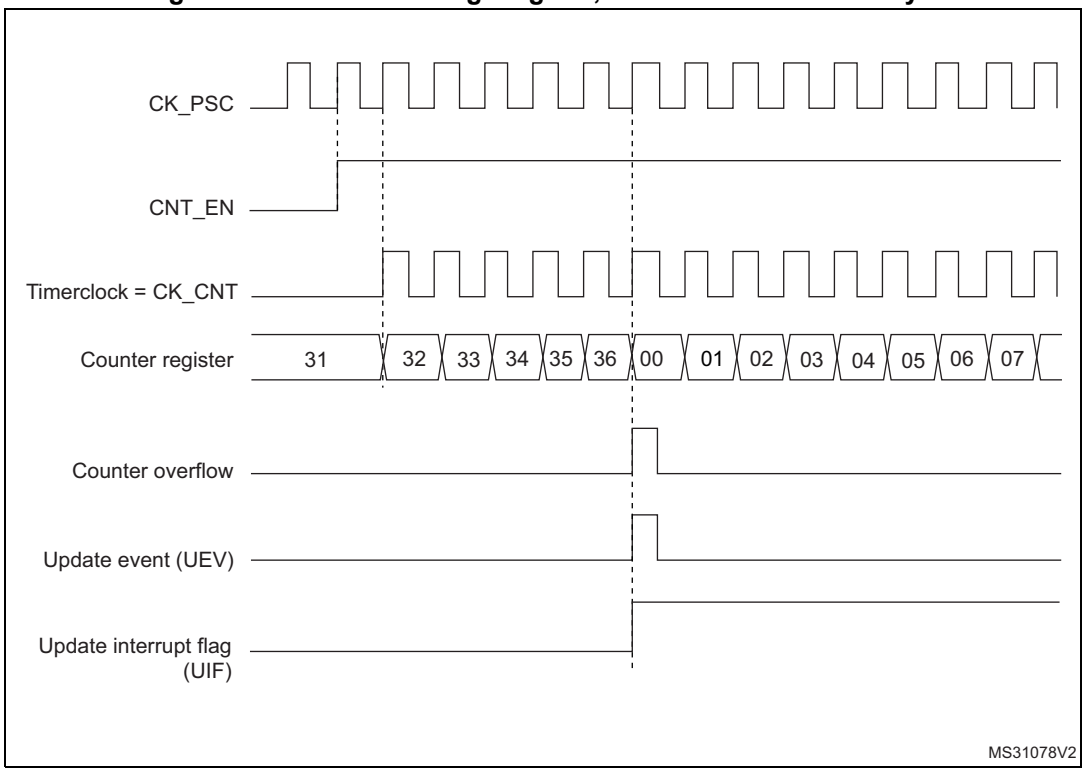



**Figure 228. Counter timing diagram, internal clock divided by 2**

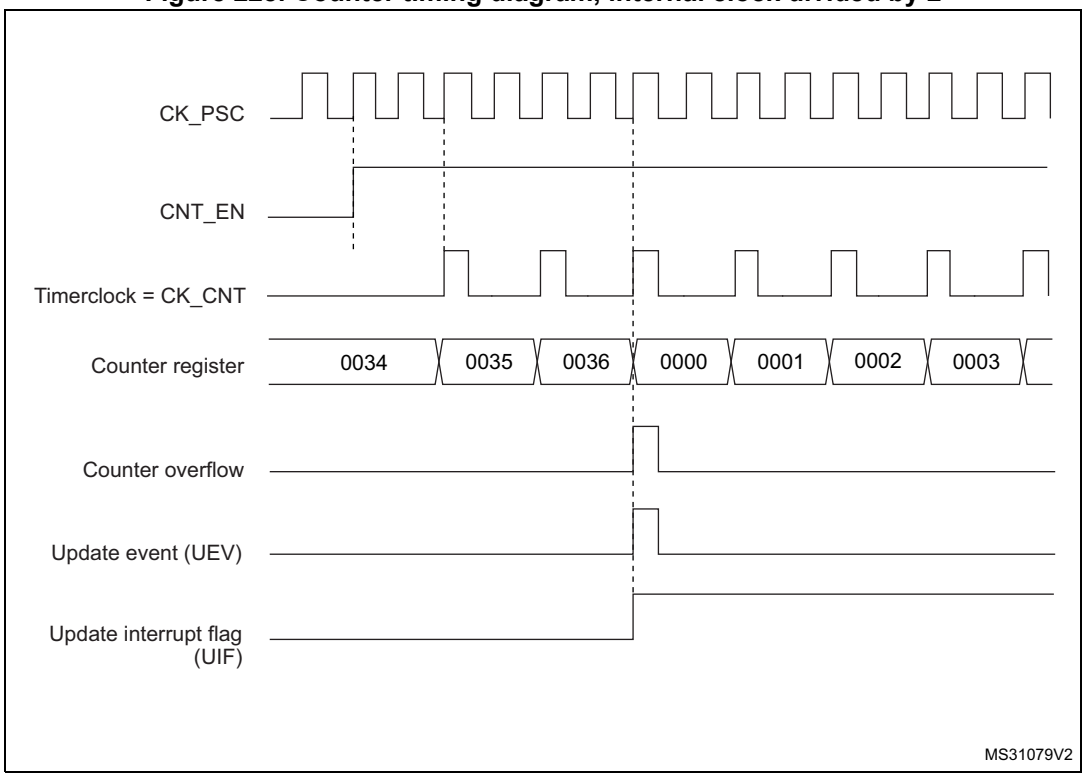


**Figure 229. Counter timing diagram, internal clock divided by 4**

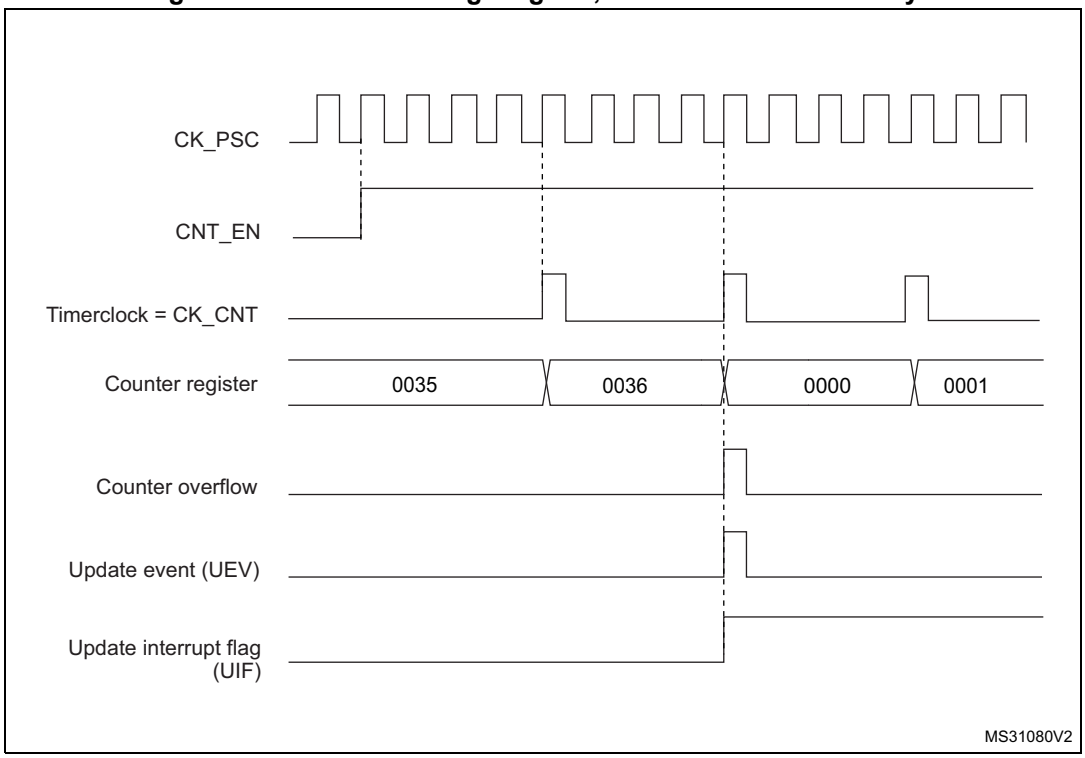



**Figure 230. Counter timing diagram, internal clock divided by N**

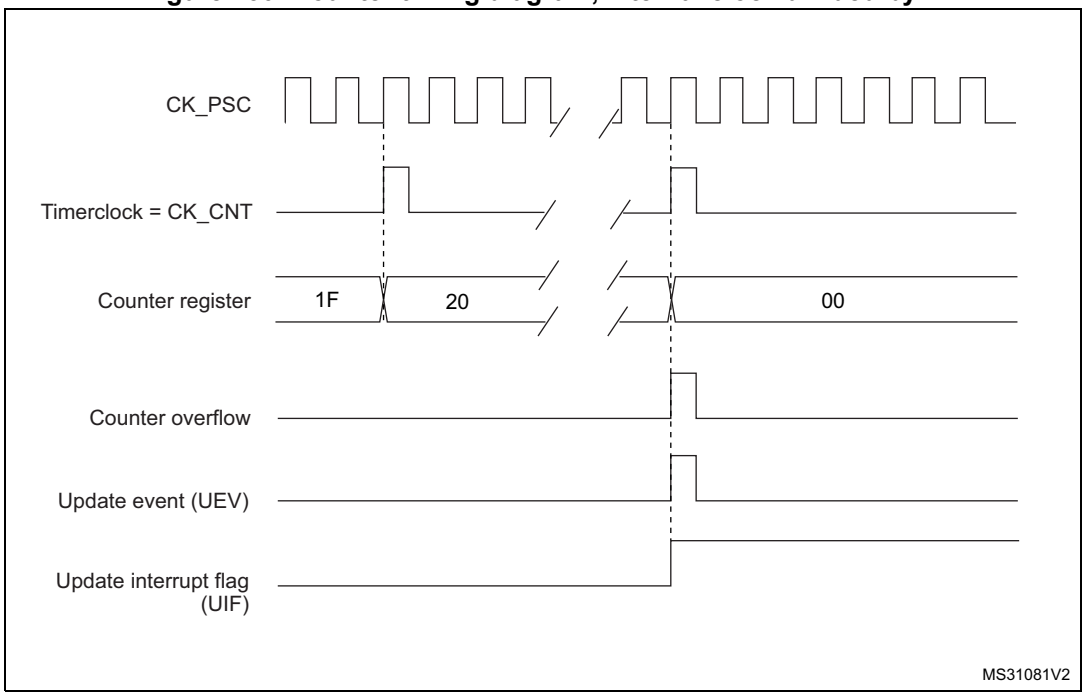


**Figure 231. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)**

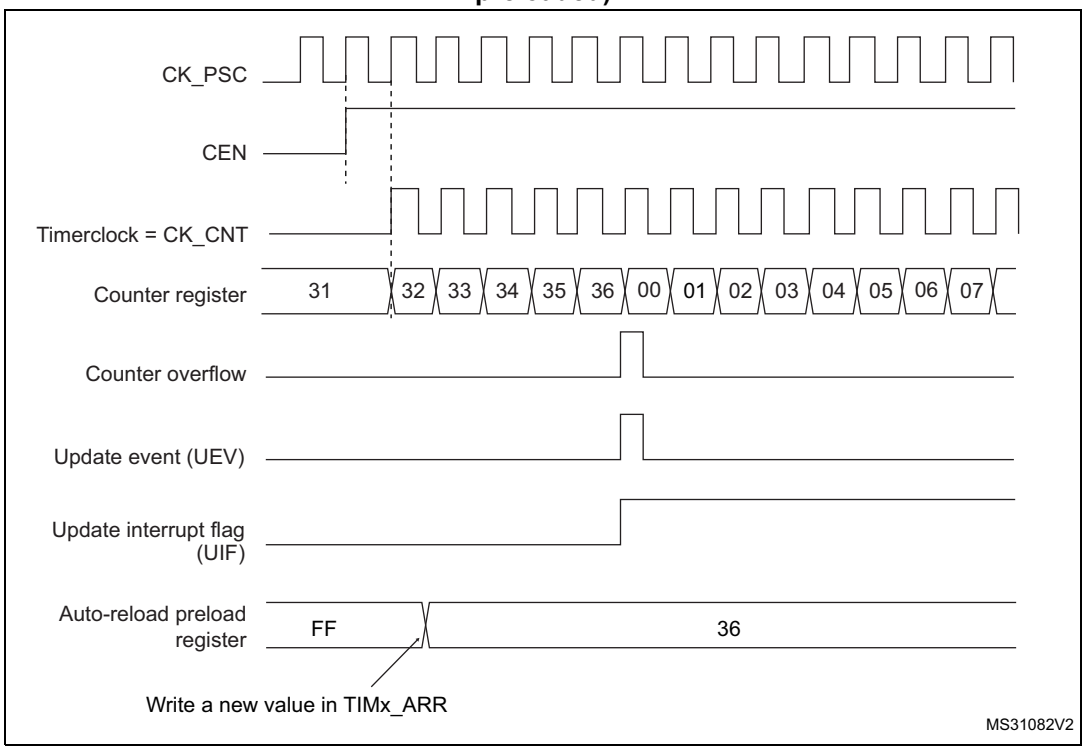
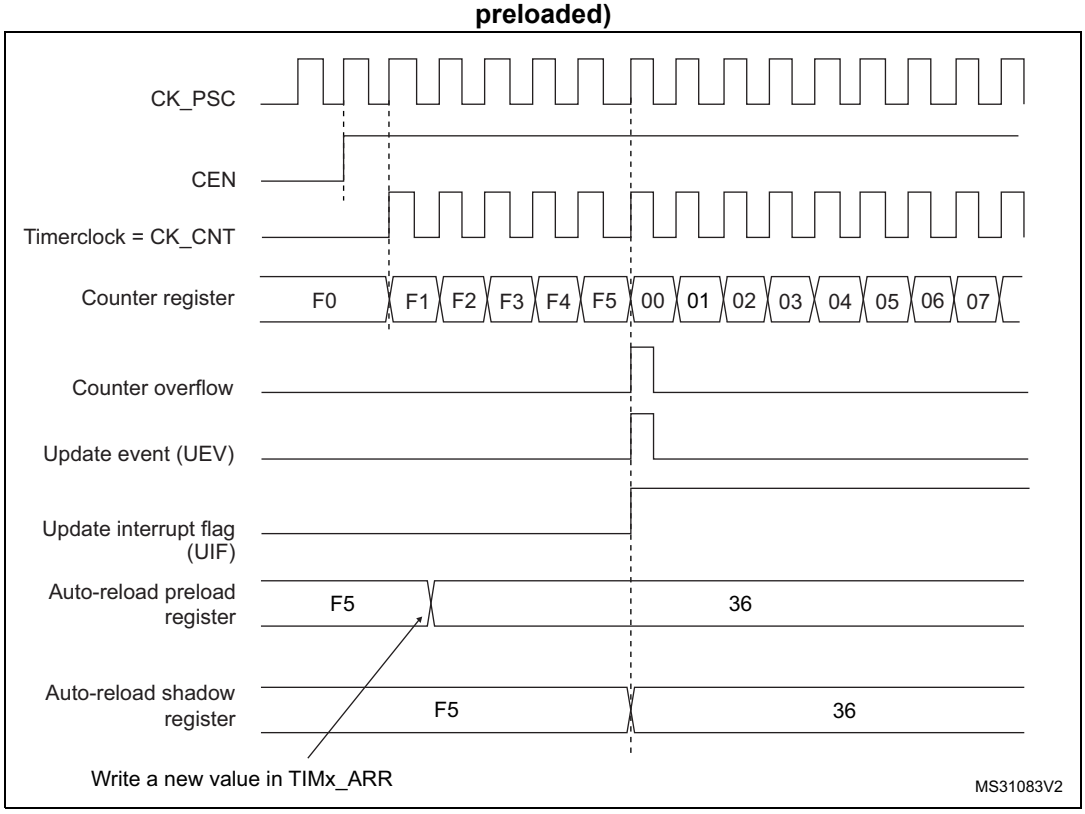



**Figure 232. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)**



### 21.3.3 Repetition counter

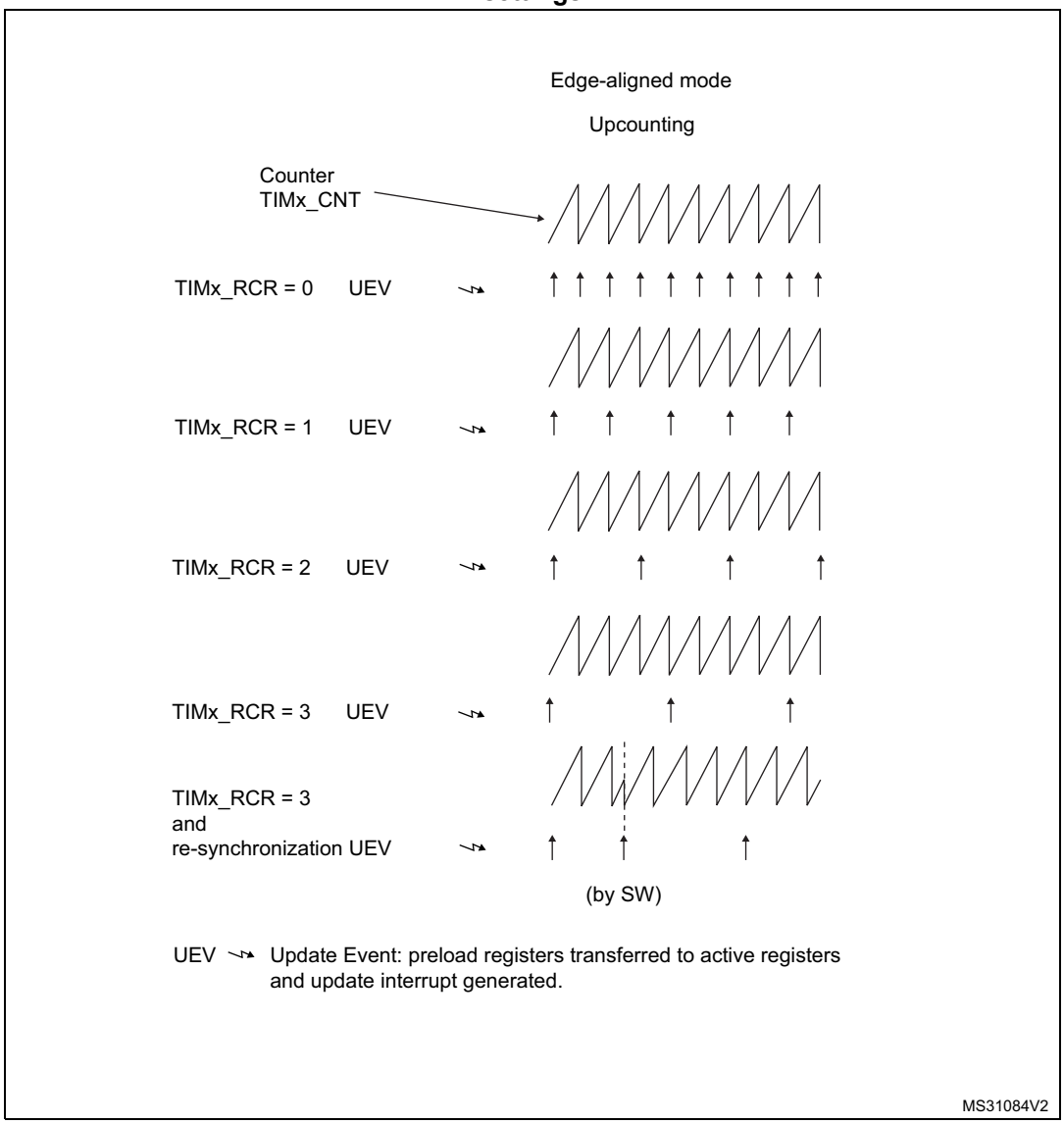
*Section 21.3.1: Time-base unit* describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

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 233*). 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 233. 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:

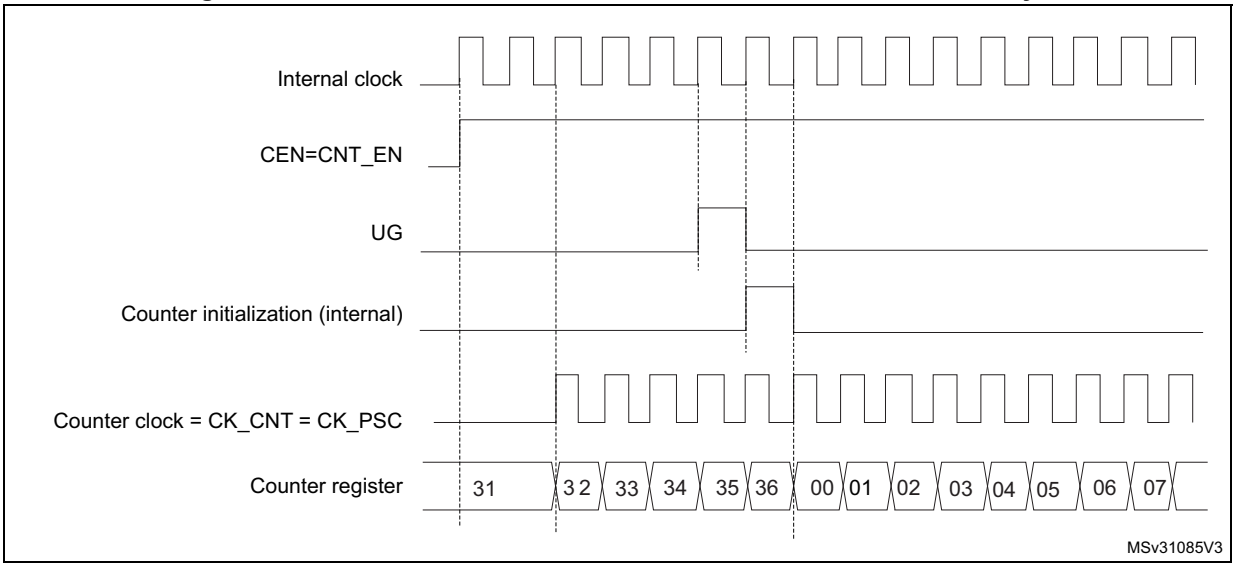
- Internal clock (CK_INT)
- External clock mode1: external input pin

### Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (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 234* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

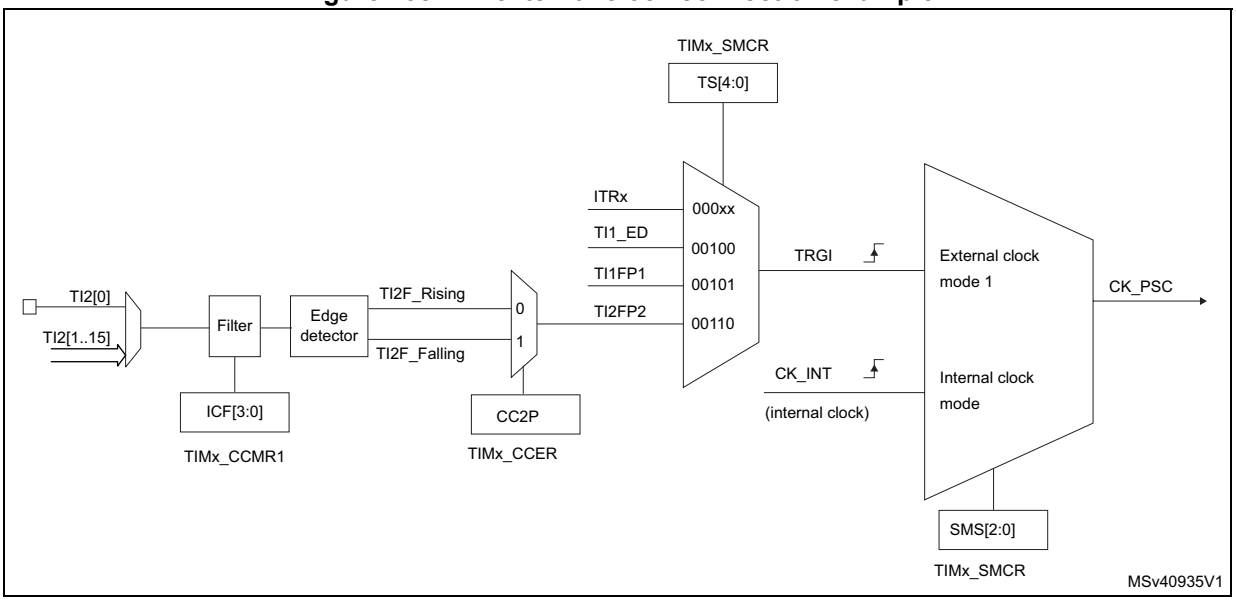
**Figure 234. Control circuit in normal mode, internal clock divided by 1**




## 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 235. 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:

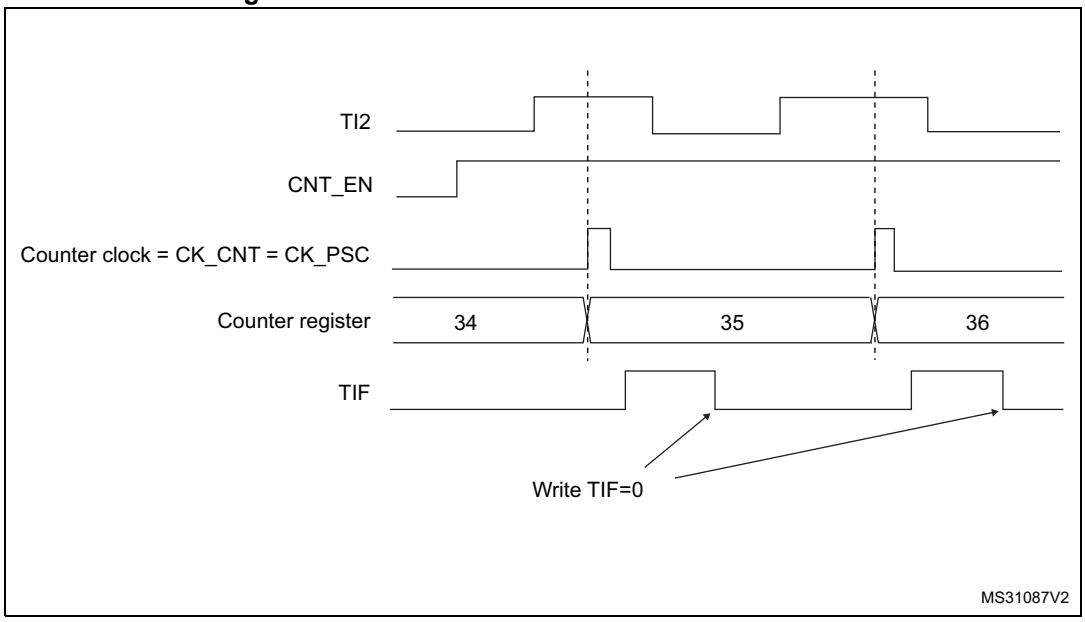
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 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=00110 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 it does not need to be configured.*

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 236. Control circuit in external clock mode 1**



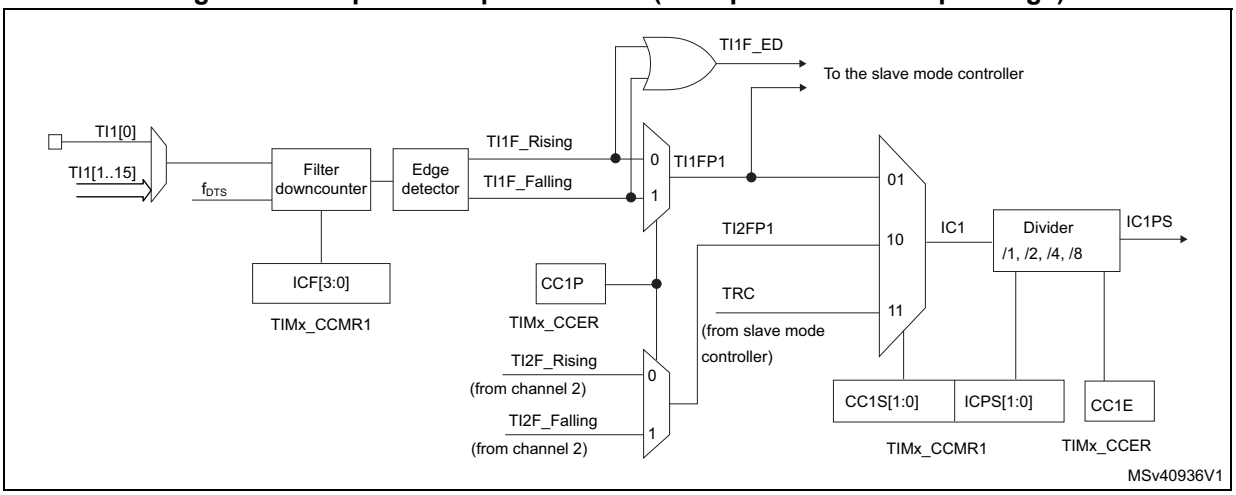

## 21.3.5 Capture/compare channels

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 237* to *Figure 239* 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 237. 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 238. Capture/compare channel 1 main circuit**

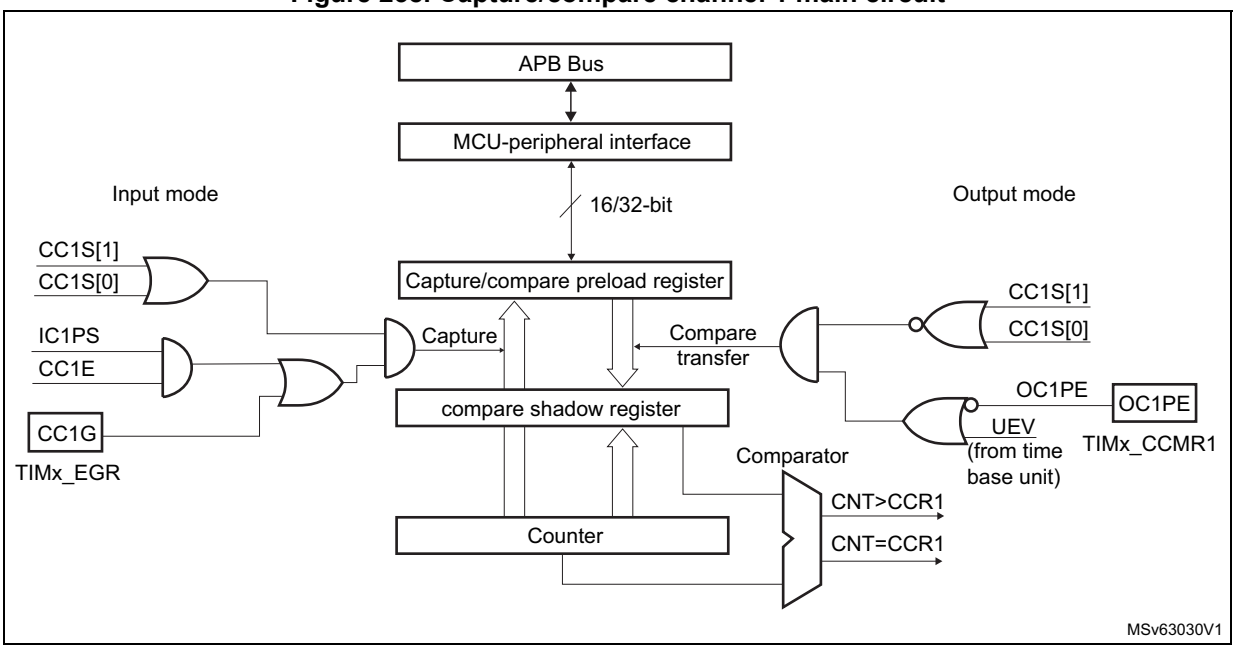
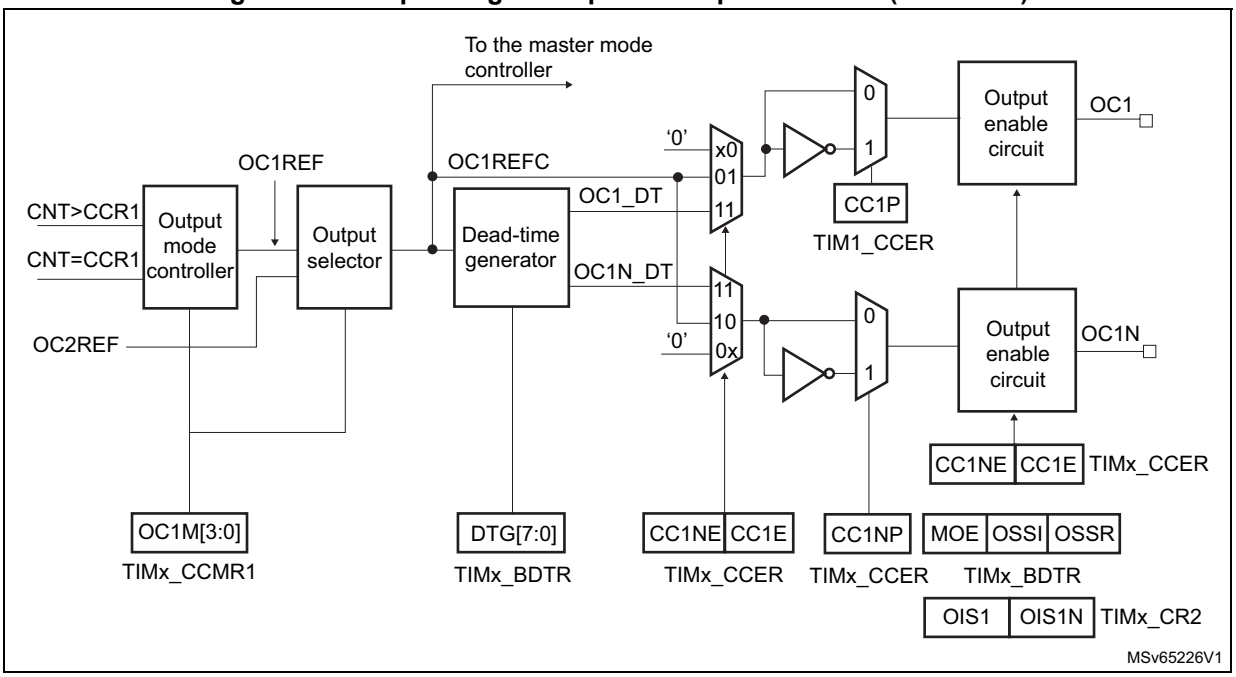


**Figure 239. Output stage of capture/compare channel (channel 1)**




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 it is written with 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 proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. 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.
3. Program the appropriate input filter duration in relation with the signal connected 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 least 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.

4. 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).
5. 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).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. 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, one just needs 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.

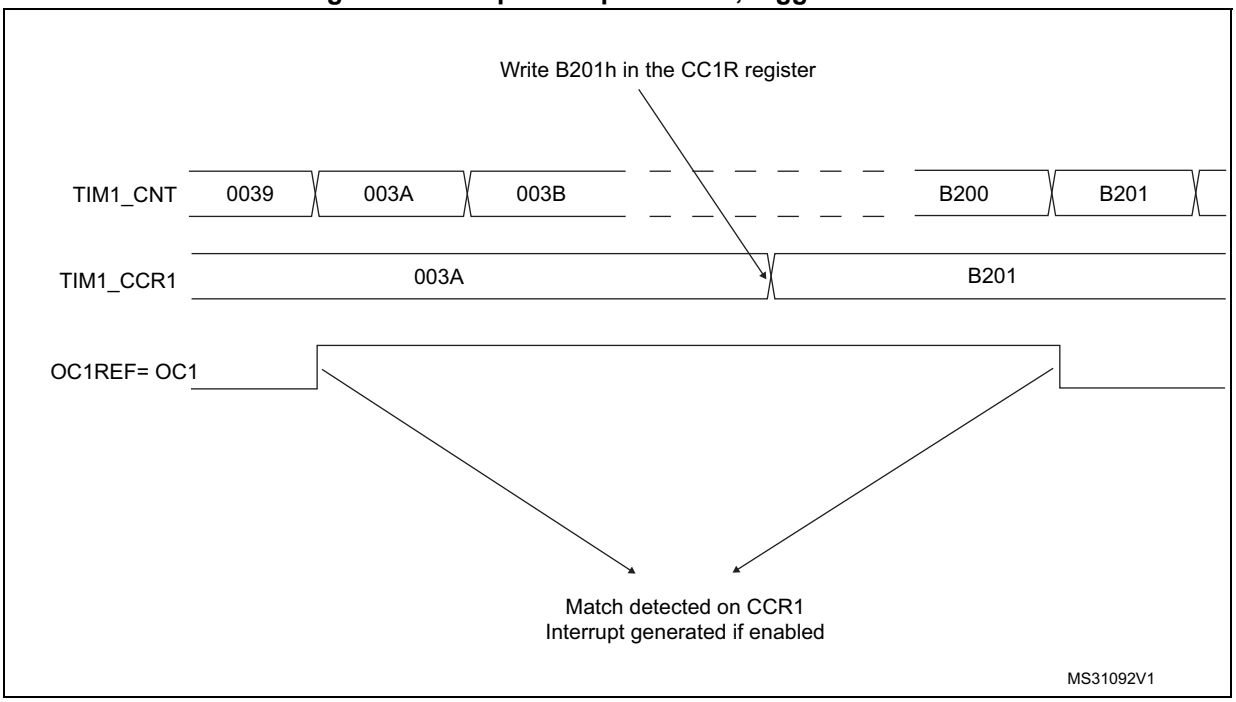
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

### 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 240*.

**Figure 240. Output compare mode, toggle on OC1**




## 21.3.9 PWM mode

Pulse Width Modulation mode allows a signal to be generated 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. The corresponding preload register must be enabled 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, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

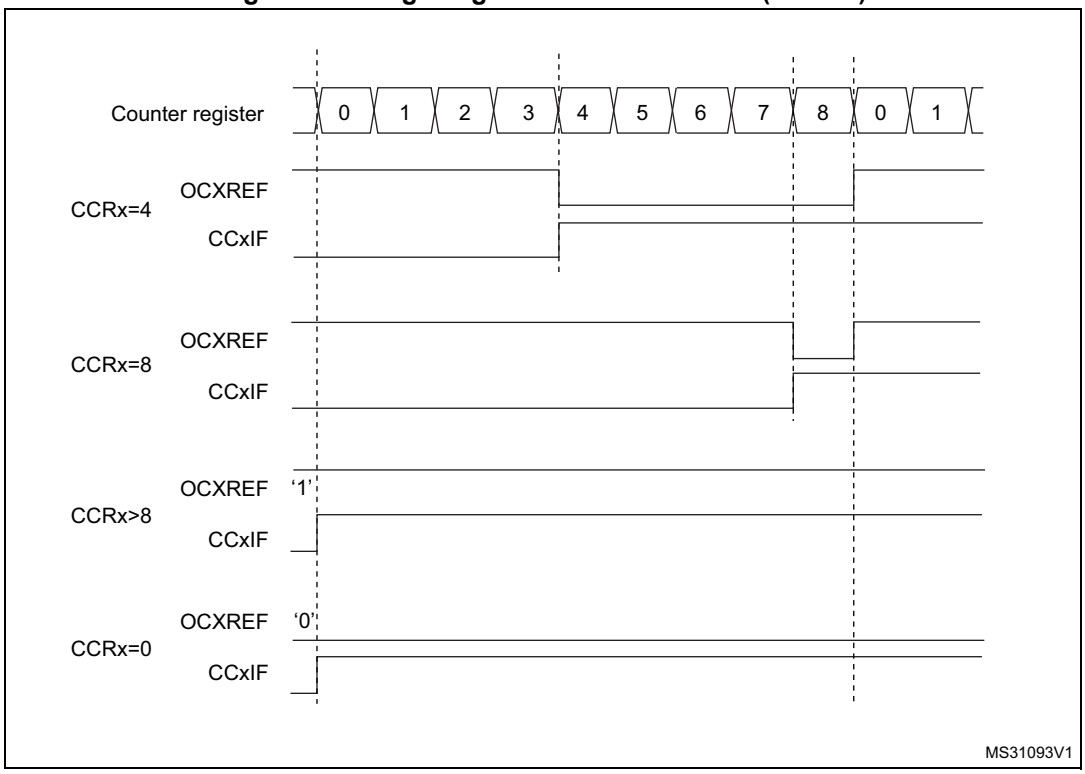
OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for 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/TIM17 are capable of upcounting only. Refer to *Upcounting mode on page 707*.

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 241* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

**Figure 241. Edge-aligned PWM waveforms (ARR=8)**




## 21.3.10 Complementary outputs and dead-time insertion

The TIM16/TIM17 general-purpose timers 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 it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected 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 136: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) on page 742* 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 242. Complementary output with dead-time insertion.**

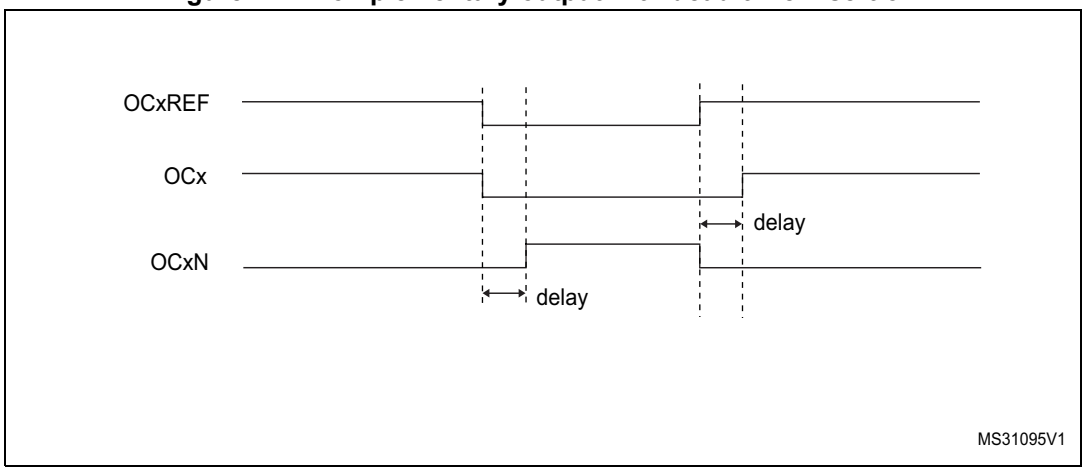



**Figure 243. Dead-time waveforms with delay greater than the negative pulse.**

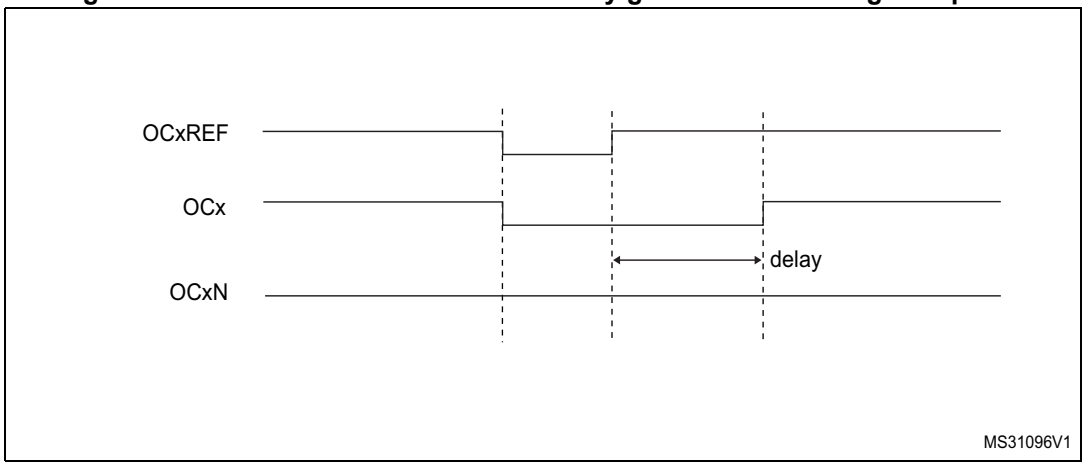
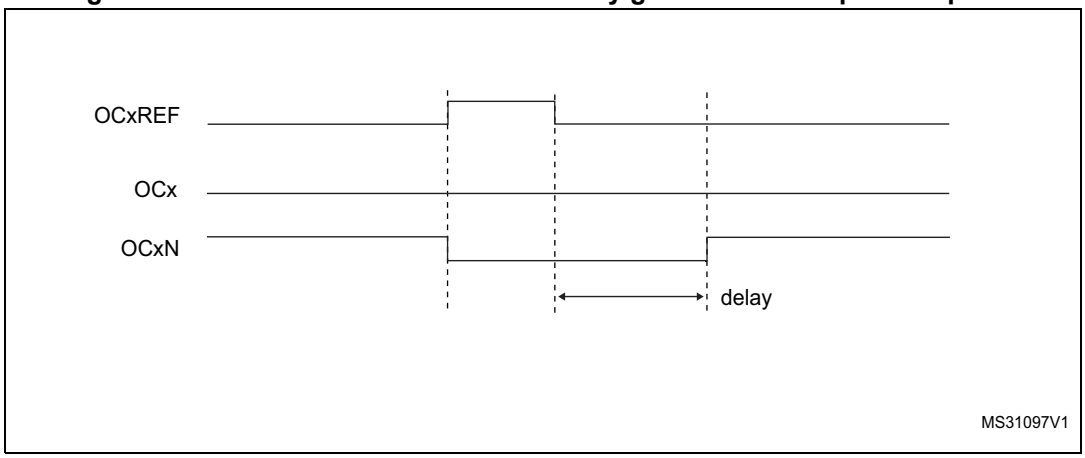


**Figure 244. Dead-time waveforms with delay greater than the positive pulse.**




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: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 745* 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 a specific waveform to be sent (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/TIM17 timers. The break input is 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 exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled 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 MOE is set to 1 whereas it was low, a delay

must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

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 GPIO (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 GPIO) 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 GPIO 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.
- 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 it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

*Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value.*
*If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.*

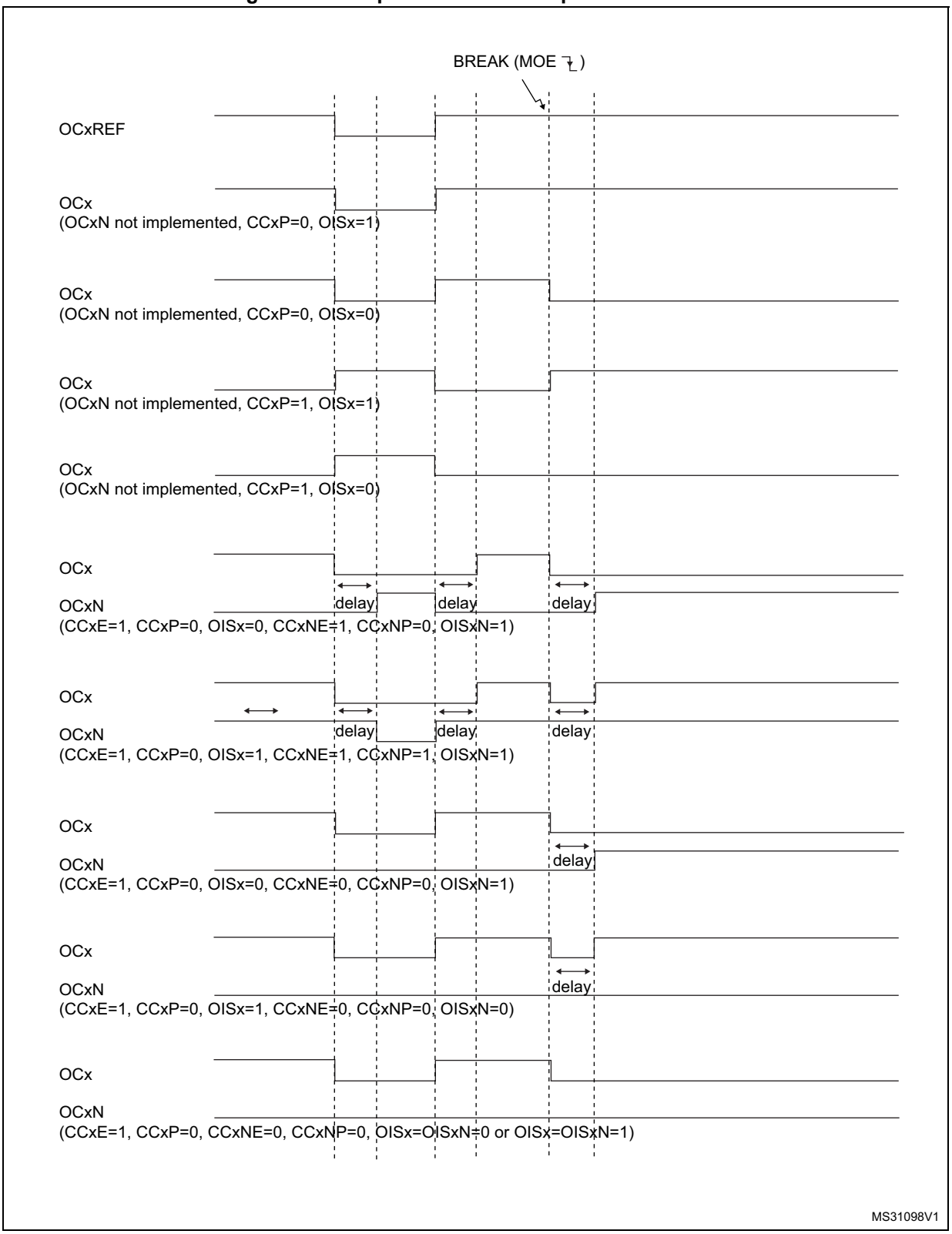
*Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.*

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 the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 21.4.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 745*. The LOCK bits can be written only once after an MCU reset.

The *Figure 245* shows an example of behavior of the outputs in response to a break.

**Figure 245. Output behavior in response to a break**



### 21.3.12 Bidirectional break inputs

The TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 246*.

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 either from system (e.g. CSS), from on-chip peripherals or 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 135*)

**Table 135. Break protection disarming conditions**

| MOE | 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

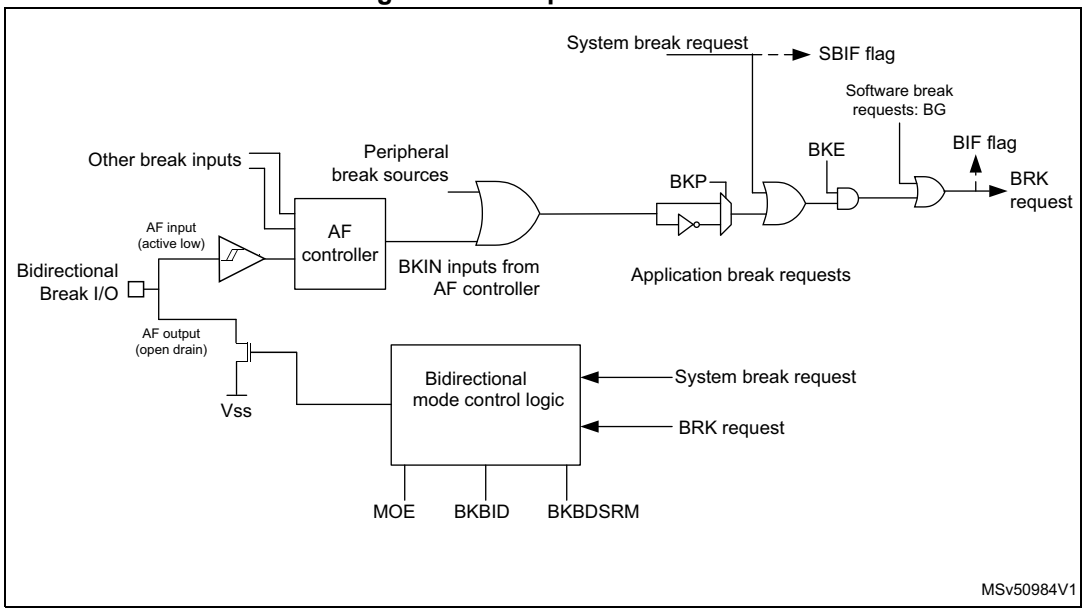
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

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 wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- 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 246. Output redirection**



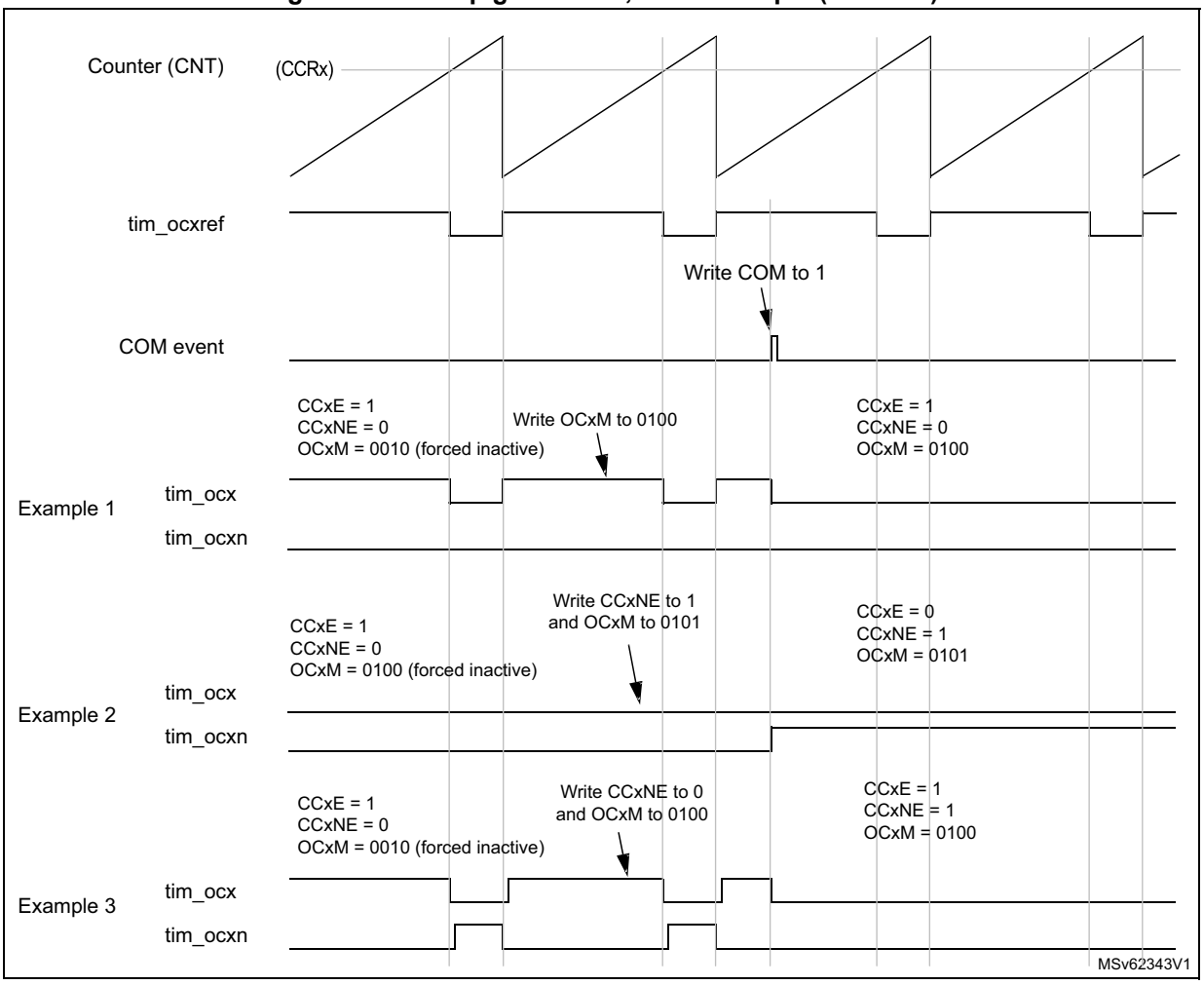

## 21.3.13 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 247* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

**Figure 247. 6-step generation, COM example (OSSR=1)**



### 21.3.14 One-pulse mode

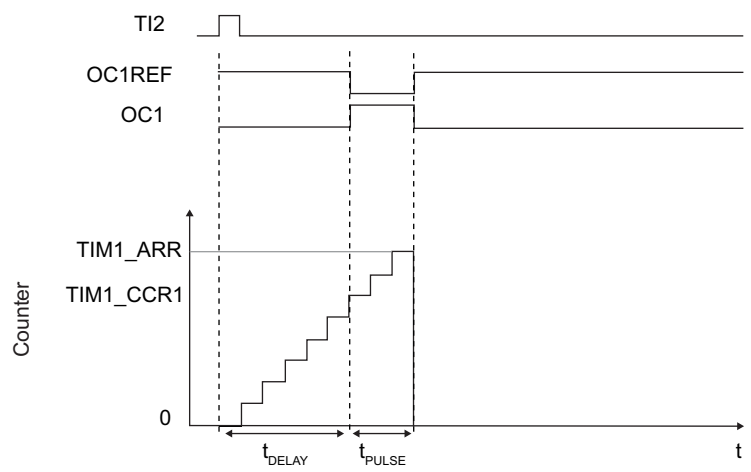
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected 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:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

**Figure 248. Example of one pulse mode**




For example one 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:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx_SMCR register.
5. 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 one 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 PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has 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.

Since only 1 pulse is needed, a 1 must be written 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 one wants to output a waveform with the minimum delay, the OCxFE bit can be set 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.

### 21.3.15 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 both the counter value and a potential roll-over condition signaled by the UIFCPY flag, to be atomically read. 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 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 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,

00010: TIMx_SMCR,

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.

*Note: A null value can be written to the reserved registers.*

### 21.3.18 Using timer output as trigger for other timers (TIM16/TIM17)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

### 21.3.19 Debug mode

When the microcontroller enters debug mode (CPU1 Cortex®-M4 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module. For more details, refer to *Section 32.8.7: DBGMCU CPU1 APB2 peripheral freeze register (DBGMCU_APB2FZR)*.

For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

## 21.4 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 21.4.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

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. | UIFREMAP | 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 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

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 | Res. | Res. | Res. | Res. | CCDS | CCUS | Res. | CCPC |
| | | | | | | 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_BDTR 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_BDTR register).*

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

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 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

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. | CC1DE | UDE | BIE | Res. | COMIE | Res. | Res. | Res. | CC1IE | UIE |
| | | | | | | rw | rw | rw | | rw | | | | rw | rw |

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

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 Reserved, must be kept at reset value.

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

### 21.4.4 TIMx status register (TIMx_SR)(x = 16 to 17)

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 | Res. | COMIF | Res. | Res. | Res. | CC1IF | UIF |
| | | | | | | 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 Reserved, must be kept at reset value.

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

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
**If channel CC1 is configured as output**: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
**If channel CC1 is configured as input**: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

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.

### 21.4.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

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 | Res. | COMG | Res. | Res. | Res. | CC1G | UG |
| | | | | | | | | 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 Reserved, must be kept at reset value.

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).

### 21.4.6 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). 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.

**Input capture 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. | IC1F[3:0] | | | | IC1PSC[1:0] | | CC1S[1:0] | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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 consecutive 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[1:0]**: 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[1:0]**: 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
Others: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).*

### 21.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). 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.

**Output compare 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. | OC1M [3] |
| | | | | | | | | | | | | | | | 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] | |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

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

Bits 15:7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: 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 can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

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 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

All other values: Reserved

*Note: 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).*

*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.*

*The OC1M[3] bit is not contiguous, located in bit 16.*

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: 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).*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

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[1:0]**: 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

Others: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).*

### 21.4.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

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 and TI2FP1. Refer to the description of CC1P.

*Note: 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).*

*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

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

**When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: 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).

CC1NP=0, CC1P=1: 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).

CC1NP=1, CC1P=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), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

*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

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**When CC1 channel is configured as output**, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 136* for details.

**Table 136. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)**

<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)<br>OCx=0<br>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)<br>OCx=0</td><td>OCxREF + Polarity<br>OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>OCxREF + Polarity<br>OCx=OCxREF XOR CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)<br>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)<br>OCx=CCxP</td><td>OCxREF + Polarity<br>OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>OCxREF + Polarity<br>OCx=OCxREF XOR CCxP,<br>OCx_EN=1</td><td>Off-State (output enabled with inactive state)<br>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).</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)<br>Asynchronously: OCx=CCxP, OCxN=CCxNP<br>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 GPIO control and alternate function registers.*

### 21.4.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

| 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 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

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 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)

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]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 21.3.1: Time-base unit on page 705* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 21.4.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

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 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

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 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

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. | BKBID | Res. | BK DSRM | 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 and DTG[7:0] bits 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 (open-drain 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 25:20 Reserved, must be kept at reset value.

Bits 19:16 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: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 740*).

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: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

*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 and CCS clock failure event) disabled

1; Break inputs (BRK and CCS clock failure event) enabled

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

*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: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 740*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO, 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: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 740*).
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 and BKE/BKP/AOE bits in TIMx_BDTR 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]) x $t_{dtg}$ with $t_{dtg} = 2 \times t_{DTS}$
DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x $t_{dtg}$ with $t_{dtg} = 8 \times t_{DTS}$
DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x $t_{dtg}$ with $t_{dtg} = 16 \times t_{DTS}$
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 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

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: TIMx_SMCR,
...

**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 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

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_OR1)

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. | TI1_RMP[1:0] | |
| | | | | | | | | | | | | | | rw | rw |

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

Bits 1:0 **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 LSI
10: TIM16 TI1 is connected to LSE
11: TIM16 TI1 is connected to RTC wake-up interrupt

### 21.4.18 TIM16 alternate function register 1 (TIM16_AF1)

Address offset: 0x60

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. | BKINP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BKINE |
| | | | | | | rw | | | | | | | | | rw |

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

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 8:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable
This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other 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 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. | 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: TIM16_CH1 input
Others: Reserved

### 21.4.20 TIM17 option register 1 (TIM17_OR1)

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. | TI1_RMP[1:0] | |
| | | | | | | | | | | | | | | rw | rw |

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

Bits 1:0 **TI1_RMP[1:0]**: Timer 17 input 1 connection
This bit is set and cleared by software.
00: TIM17 TI1 is connected to GPIO
01: TIM17 TI1 is connected to MSI
10: TIM17 TI1 is connected to HSE/32
11: TIM17 TI1 is connected to MCO

### 21.4.21 TIM17 alternate function register 1 (TIM17_AF1)

Address offset: 0x60

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. | BKINP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BKINE |
| | | | | | | rw | | | | | | | | | rw |

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

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 8:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other 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).*

## 21.4.22 TIM17 input selection register (TIM17_TISEL)

Address offset: 0x68

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. | 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: TIM17_CH1 input

Others: Reserved

### 21.4.23 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 137. TIM16/TIM17 register map and reset values**

| Offset | Register name | 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. | Res. | Res. | Res. | CCDS | CCUS | Res. | CCPC |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 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. | Res. | Res. | Res. | Res. | Res. | CC1DE | UDE | BIE | Res. | COMIE | Res. | Res. | Res. | CC1IE | UIE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 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 | Res. | COMIF | Res. | Res. | Res. | CC1IF | UIF |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 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 | Res. | COMG | Res. | Res. | Res. | CC1G | UG |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 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. | | 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] | | | | IC1PSC [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 137. TIM16/TIM17 register map and reset values (continued)**

| Offset | Register name | 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] | | DTG[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 |
| 0x50 | **TIM16_OR1** | Res. | Res. | Res. | Res. | Res. | Res. | 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] | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 |
| 0x50 | **TIM17_OR1** | Res. | Res. | Res. | Res. | Res. | Res. | 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] | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 |
| 0x60 | **TIM16_AF1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BKINP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BKINE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | | | | | 1 |
| 0x60 | **TIM17_AF1** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BKINP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BKINE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | | | | | 1 |
| 0x68 | **TIM16_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 |
| 0x68 | **TIM17_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 on page 56* for the register boundary addresses.

# 22 Low-power timer (LPTIM)

## 22.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

## 22.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no embedded oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode

## 22.3 LPTIM implementation

*Table 138* describes LPTIM implementation: the full set of features is implemented in LPTIM1. LPTIM2 supports a smaller set of features, but is otherwise identical to LPTIM1.

**Table 138. LPTIM features**

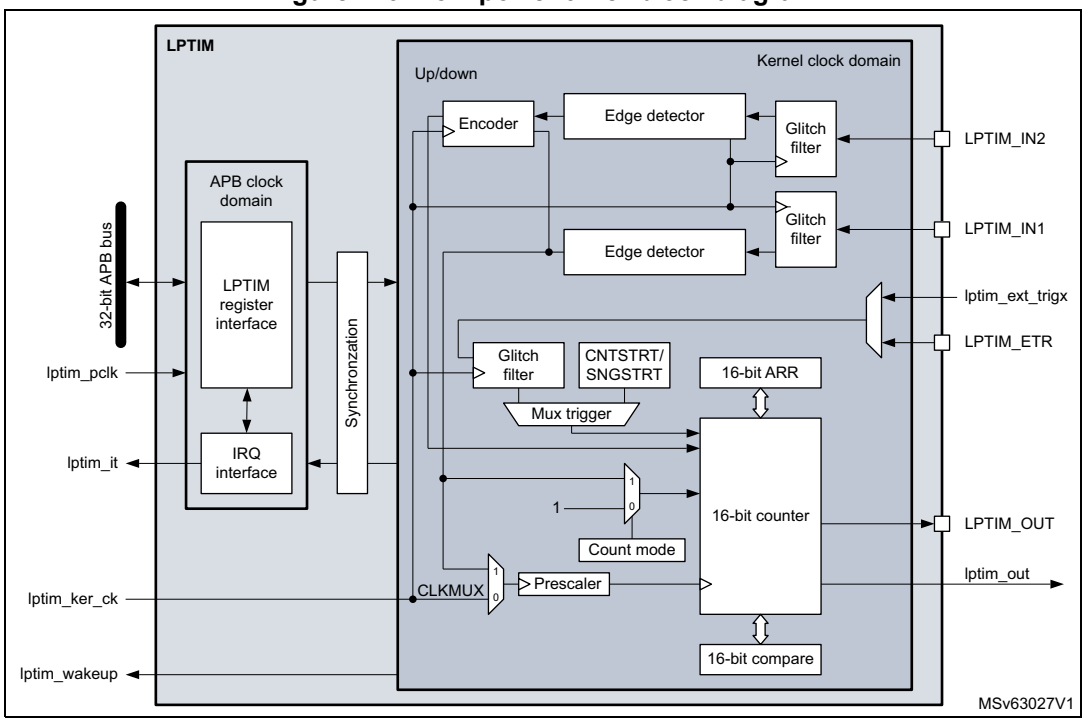
| LPTIM modes/features(1) | LPTIM1 | LPTIM2 |
|---|---|---|
| Encoder mode | X | - |
| External input clock | X | X |
| Wake-up from Stop | (2) | (3) |

1. X = supported.
2. Wake-up supported from Stop 0, Stop 1 and Stop 2 modes.
3. Wake-up supported from Stop 0 and Stop 1 modes.

## 22.4 LPTIM functional description

### 22.4.1 LPTIM block diagram

**Figure 249. Low-power timer block diagram**



### 22.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

**Table 139. LPTIM input/output pins**

| Names | Signal type | Description |
|---|---|---|
| LPTIM_IN1 | Digital input | LPTIM Input 1 from GPIO pin |
| LPTIM_IN2 | Digital input | LPTIM Input 2 from GPIO pin |
| LPTIM_ETR | Digital input | LPTIM external trigger GPIO pin |
| LPTIM_OUT | Digital output | LPTIM Output GPIO pin |

**Table 140. LPTIM internal signals**

| Names | Signal type | Description |
|---|---|---|
| lptim_pclk | Digital input | LPTIM APB clock domain |
| lptim_ker_ck | Digital input | LPTIM kernel clock |
| lptim_ext_trigx | Digital input | LPTIM external trigger input x |
| lptim_out | Digital output | LPTIM counter output |
| lptim_it | Digital output | LPTIM global interrupt |
| lptim_wakeup | Digital output | LPTIM wake-up event |

### 22.4.3 LPTIM trigger mapping

The LPTIM external trigger connections are detailed hereafter:

**Table 141. LPTIM1 external trigger connection**

| TRIGSEL | External trigger |
|---|---|
| lptim_ext_trig0 | GPIO |
| lptim_ext_trig1 | RTC alarm A |
| lptim_ext_trig2 | RTC alarm B |
| lptim_ext_trig3 | Not connected |
| lptim_ext_trig4 | RTC_TAMP2 input detection |
| lptim_ext_trig5 | Not connected |
| lptim_ext_trig6 | Not connected |
| lptim_ext_trig7 | Not connected |

**Table 142. LPTIM2 external trigger connection**

| TRIGSEL | External trigger |
|---|---|
| lptim_ext_trig0 | GPIO |
| lptim_ext_trig1 | RTC alarm A |

**Table 142. LPTIM2 external trigger connection (continued)**

| TRIGSEL | External trigger |
|---|---|
| lptim_ext_trig2 | RTC alarm B |
| lptim_ext_trig3 | Not connected |
| lptim_ext_trig4 | RTC_TAMP2 input detection |
| lptim_ext_trig5 | Not connected |
| lptim_ext_trig6 | Not connected |
| lptim_ext_trig7 | Not connected |

### 22.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM will use an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal should also be provided (first configuration). In this case, the internal clock signal frequency should be at least four times higher than the external clock signal frequency.

### 22.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source should first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

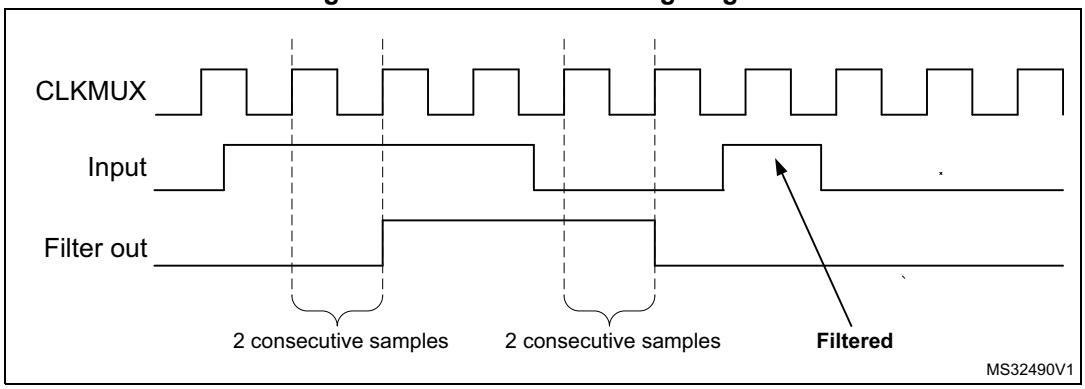
The digital filters are divided into two groups:

- The first group of digital filters protects the LPTIM external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.

*Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.*

The filter sensitivity acts on the number of consecutive equal samples that should be detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 250* shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

**Figure 250. Glitch filter timing diagram**




*Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT and TRGFLT bits to '0'. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.*

## 22.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

**Table 143. Prescaler division ratios**

| programming | dividing factor |
|---|---|
| 000 | /1 |
| 001 | /2 |
| 010 | /4 |
| 011 | /8 |
| 100 | /16 |
| 101 | /32 |
| 110 | /64 |
| 111 | /128 |

## 22.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it will be ignored (unless timeout function is enabled).

*Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled will be discarded by hardware.*

*Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.*

## 22.4.8 Operating mode

The LPTIM features two operating modes:

- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when reaching the ARR value.

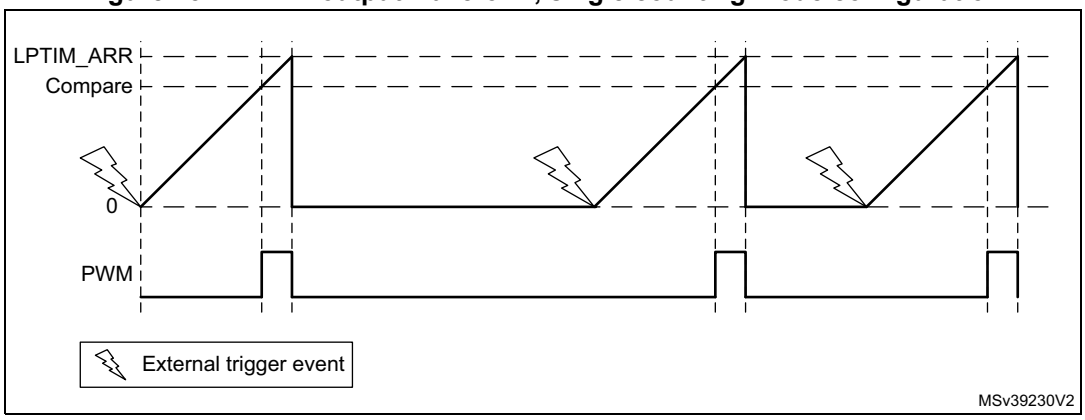
### One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event will re-start the timer. Any trigger event occurring after the counter starts and before the counter reaches ARR will be discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the counter register has stopped (contains zero value), will start the counter for a new one-shot counting cycle as shown in *Figure 251*.

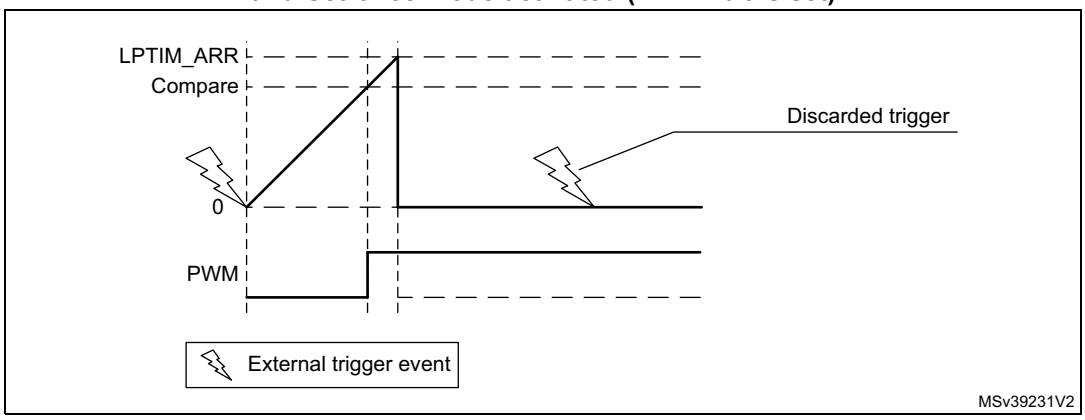
**Figure 251. LPTIM output waveform, single counting mode configuration**



Set-once mode activated:

It should be noted that when the WAVE bit-field in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 252*.

**Figure 252. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)**




In case of software start (TRIGEN[1:0] = ‘00’), the SNGSTRT setting will start the counter for one-shot counting.

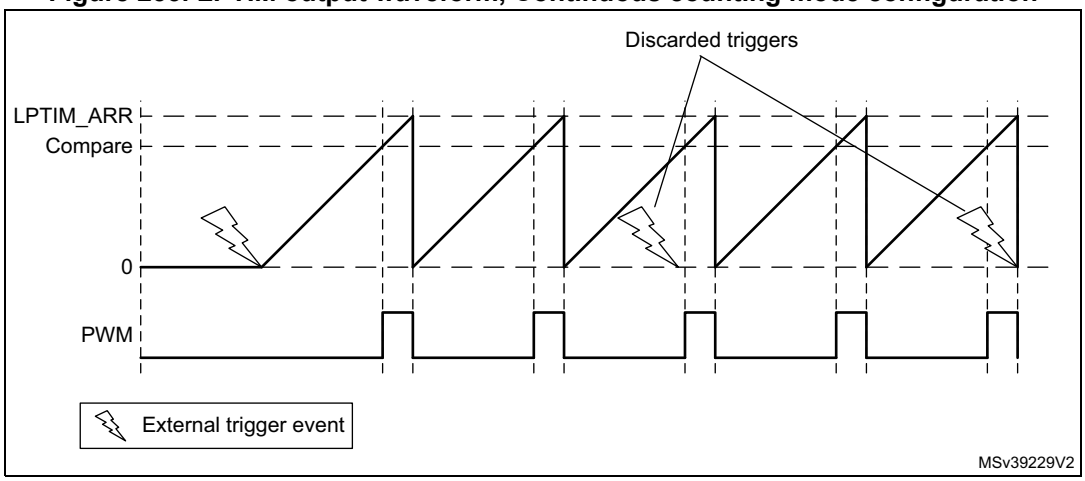
## Continous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set will start the counter for continuous counting. Any subsequent external trigger event will be discarded as shown in *Figure 253*.

In case of software start (TRIGEN[1:0] = ‘00’), setting CNTSTRT will start the counter for continuous counting.

**Figure 253. LPTIM output waveform, Continuous counting mode configuration**




SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to ‘1’). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT will switch the LPTIM to the One-shot mode. The counter (if active) will stop as soon as it reaches ARR.

If the One-shot mode was previously selected, setting CNTSTRT will switch the LPTIM to the Continuous mode. The counter (if active) will restart as soon as it reaches ARR.

### 22.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event will start the timer, any successive trigger event will reset the counter and the timer will restart.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

### 22.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CMP (compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CMP. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT registers.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM_ARR register value be strictly greater than the LPTIM_CMP register value.

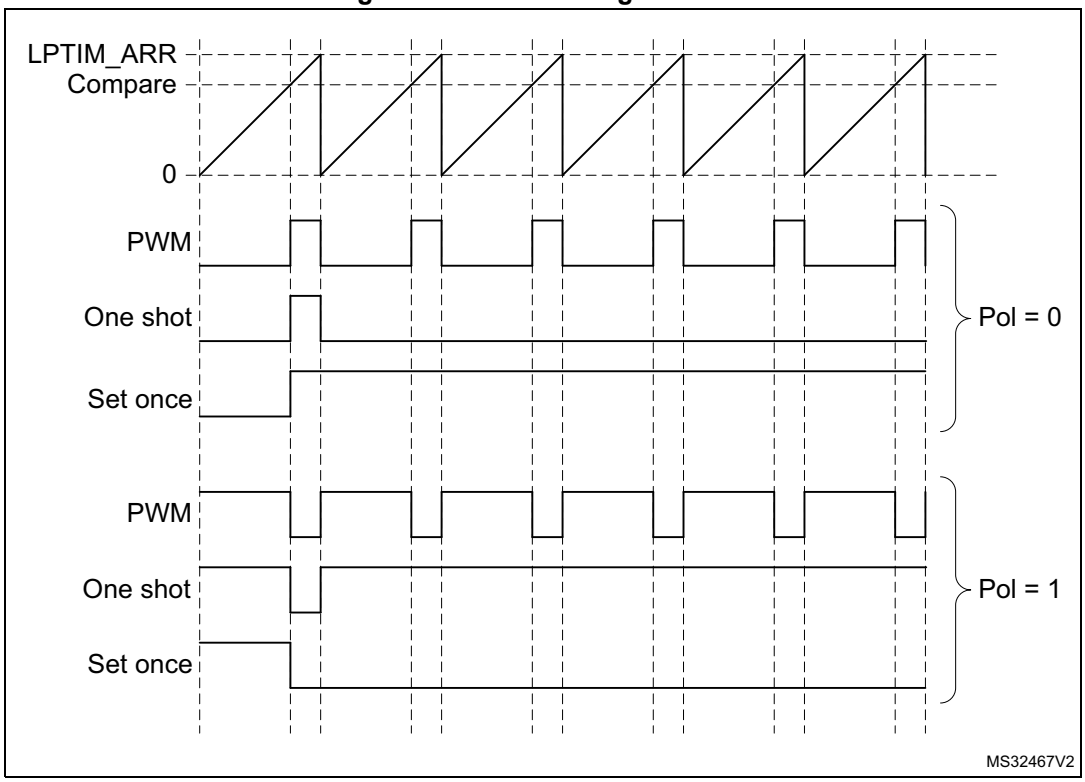
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to ‘0’ forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to ‘1’ forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value will change immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. *Figure 254* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL bit.

Figure 254. Waveform generation




## 22.4.11 Register update

The LPTIM_ARR register and LPTIM_CMP register are updated immediately after the APB bus write operation, or at the end of the current period if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR and the LPTIM_CMP registers are updated:

- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR and the LPTIM_CMP registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR and the LPTIM_CMP registers are updated at the end of the current period, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag and the CMPOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register and the LPTIM_CMP register.

After a write to the LPTIM_ARR register or the LPTIM_CMP register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag or the CMPOK flag be set, will lead to unpredictable results.

### 22.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source will be used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

### 22.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM_CFGR and LPTIM_IER registers must be modified only when the LPTIM is disabled.

### 22.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bit-field to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic will elapse before the reset is taken into account. This will make the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.
- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register will reset its content to zero. Asynchronous reset should be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset should be applied only when there is enough insurance that no toggle will occur on the LPTIM Input1.
  It should be noted that to read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

---

**Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So developer should make sure that these two mechanisms are used exclusively.**

---

### 22.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. 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 programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two Down and Up flags in the LPTIM_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to ‘1’. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder’s position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

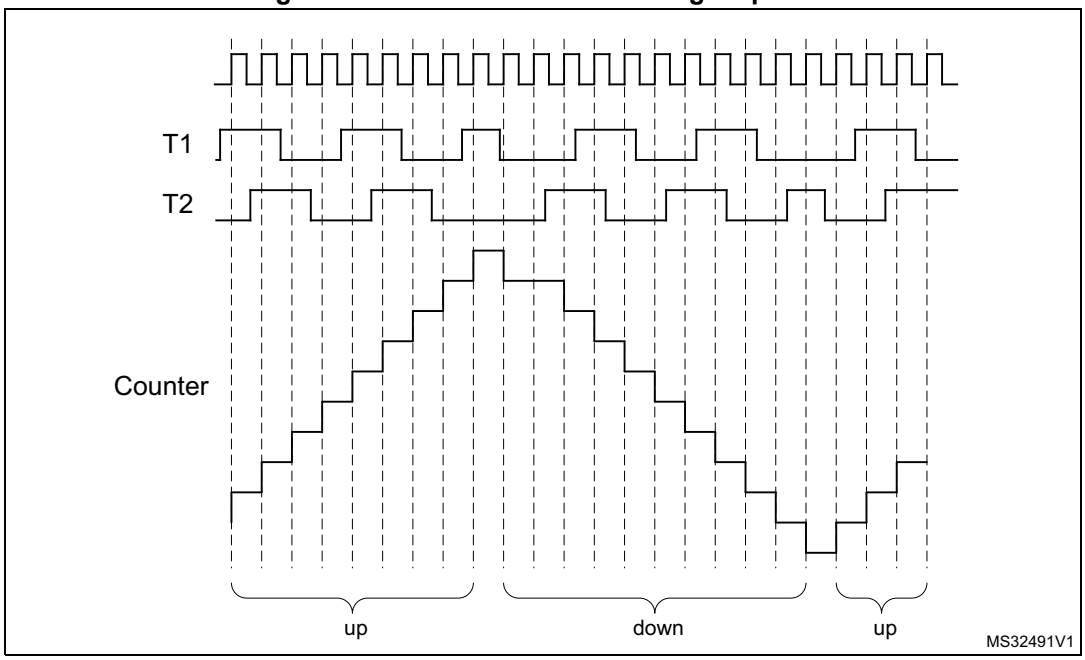
**Table 144. Encoder counting scenarios**

| Active edge | Level on opposite signal (Input1 for Input2, Input2 for Input1) | Input1 signal | | Input2 signal | |
|---|---|---|---|---|---|
| | | Rising | Falling | Rising | Falling |
| Rising Edge | High | Down | No count | Up | No count |
| | Low | Up | No count | Down | No count |
| Falling Edge | High | No count | Up | No count | Down |
| | Low | No count | Down | No count | Up |
| Both Edges | High | Down | Up | Up | Down |
| | Low | Up | Down | Down | Up |

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

**Caution:** In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to ‘0’. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be ‘000’).

Figure 255. Encoder mode counting sequence




### 22.4.16 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the DBG_LPTIM_STOP configuration bit in the DBG module.

## 22.5 LPTIM low-power modes

Table 145. Effect of low-power modes on the LPTIM

| Mode | Description |
|---|---|
| Sleep | No effect. LPTIM interrupts cause the device to exit Sleep mode. |
| Stop | If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode (refer to *Section 22.3: LPTIM implementation*). |
| Standby | The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode. |

## 22.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_IER register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).

*Note: If any bit in the LPTIM_IER register (Interrupt Enable Register) is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.*

**Table 146. Interrupt events**

| Interrupt event | Description |
|---|---|
| Compare match | Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the compare register (LPTIM_CMP). |
| Auto-reload match | Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the Auto-reload register (LPTIM_ARR). |
| External trigger event | Interrupt flag is raised when an external trigger event is detected |
| Auto-reload register update OK | Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete. |
| Compare register update OK | Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete. |
| Direction change | Used in Encoder mode. Two interrupt flags are embedded to signal direction change:<br>– UP flag signals up-counting direction change<br>– DOWN flag signals down-counting direction change. |

## 22.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

### 22.7.1 LPTIM interrupt and status register (LPTIM_ISR)

Address offset: 0x000

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. | DOWN | UP | ARR OK | CMP OK | EXT TRIG | ARRM | CMPM |
| | | | | | | | | | r | r | r | r | r | r | r |

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

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 22.3: LPTIM implementation.*

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 22.3: LPTIM implementation.*

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMPOK**: Compare register update OK

CMPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_CMP register has been successfully completed. CMPOK flag can be cleared by writing 1 to the CMPOKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CMPM**: Compare match

The CMPM bit is set by hardware to inform application that LPTIM_CNT register value reached the LPTIM_CMP register's value. CMPM flag can be cleared by writing 1 to the CMPMCF bit in the LPTIM_ICR register.

### 22.7.2 LPTIM interrupt clear register (LPTIM_ICR)

Address offset: 0x004

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. | DOWNCF | UPCF | ARROKCF | CMPOKCF | EXTTRIGCF | ARRMCF | CMPMCF |
| | | | | | | | | | w | w | w | w | w | w | w |

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

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 22.3: LPTIM implementation.*

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 22.3: LPTIM implementation.*

Bit 4 **ARROKCF**: Autoreload register update OK clear flag

Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMPOKCF**: Compare register update OK clear flag

Writing 1 to this bit clears the CMPOK flag in the LPTIM_ISR register

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag

Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag

Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CMPMCF**: Compare match clear flag

Writing 1 to this bit clears the CMPM flag in the LPTIM_ISR register

### 22.7.3 LPTIM interrupt enable register (LPTIM_IER)

Address offset: 0x008

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. | DOWNIE | UPIE | ARROKIE | CMPOKIE | EXTTRIGIE | ARRMIE | CMPMIE |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

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

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
- 0: DOWN interrupt disabled
- 1: DOWN interrupt enabled

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 22.3: LPTIM implementation.*

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
- 0: UP interrupt disabled
- 1: UP interrupt enabled

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 22.3: LPTIM implementation.*

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
- 0: ARROK interrupt disabled
- 1: ARROK interrupt enabled

Bit 3 **CMPOKIE**: Compare register update OK Interrupt Enable
- 0: CMPOK interrupt disabled
- 1: CMPOK interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
- 0: EXTTRIG interrupt disabled
- 1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
- 0: ARRM interrupt disabled
- 1: ARRM interrupt enabled

Bit 0 **CMPMIE**: Compare match Interrupt Enable
- 0: CMPM interrupt disabled
- 1: CMPM interrupt enabled

**Caution:** The LPTIM_IER register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’)

## 22.7.4 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

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. | ENC | COUNT MODE | PRELOAD | WAVPOL | WAVE | TIMOUT | TRIGEN[1:0] | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRIGSEL[2:0] | | | Res. | PRESC[2:0] | | | Res. | TRGFLT[1:0] | | Res. | CKFLT[1:0] | | CKPOL[1:0] | | CKSEL |
| rw | rw | rw | | rw | rw | rw | | rw | rw | | rw | rw | rw | rw | rw |

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

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable

The ENC bit controls the Encoder mode

0: Encoder mode disabled

1: Encoder mode enabled

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 22.3: LPTIM implementation.*

Bit 23 **COUNTMODE**: counter mode enabled

The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:

0: the counter is incremented following each internal clock pulse

1: the counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode

The PRELOAD bit controls the LPTIM_ARR and the LPTIM_CMP registers update modality

0: Registers are updated after each APB bus write access

1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity

The WAVEPOL bit controls the output polarity

0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CMP registers

1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CMP registers

Bit 20 **WAVE**: Waveform shape

The WAVE bit controls the output shape

0: Deactivate Set-once mode

1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable

The TIMOUT bit controls the Timeout feature

0: A trigger event arriving when the timer is already started will be ignored

1: A trigger event arriving when the timer is already started will reset and restart the counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity

The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:

00: software trigger (counting start is initiated by software)

01: rising edge is the active edge

10: falling edge is the active edge

11: both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that will serve as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7

See *Section 22.4.3: LPTIM trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any trigger active level change is considered as a valid trigger
01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any external clock signal level change is considered as a valid transition

01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

If LPTIM is clocked by an external clock source:

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00:the rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 1 is active.

01:the falling edge is the active edge used for counting
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 2 is active.

10:both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 3 is active.

11:not allowed

Refer to *Section 22.4.15: Encoder mode* for more details about Encoder mode sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM will use:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

**Caution:** The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

## 22.7.5 LPTIM control register (LPTIM_CR)

Address offset: 0x010

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. | RST ARE | COUN TRST | CNT STRT | SNG STRT | ENA BLE |
| | | | | | | | | | | | rw | rs | rw | rw | rw |

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

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register will asynchronously reset LPTIM_CNT register content.

This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit will trigger a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

**Caution:** COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software should consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer will not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM will stop at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0:LPTIM is disabled

1:LPTIM is enabled

### 22.7.6 LPTIM compare register (LPTIM_CMP)

Address offset: 0x014

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMP[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 **CMP[15:0]:** Compare value
CMP is the compare value used by the LPTIM.

**Caution:** The LPTIM_CMP register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

### 22.7.7 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARR[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 **ARR[15:0]:** Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CMP[15:0] value.

**Caution:** The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

### 22.7.8 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[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 **CNT[15:0]:** Counter value

When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.

It should be noted that for a reliable LPTIM_CNT register read access, two consecutive read accesses must be performed and compared. A read access can be considered reliable when the values of the two consecutive read accesses are equal.

### 22.7.9 LPTIM register map

The following table summarizes the LPTIM registers.

**Table 147. LPTIM register map and reset values**

| Offset | Register name | 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 | LPTIM_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. | DOWN(1) | UP(1) | ARROK | CMPOK | EXTTRIG | ARRM | CMPM |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x004 | LPTIM_ICR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DOWNCF(1) | UPCF(1) | ARROKCF | CMPOKCF | EXTTRIGCF | ARRMCF | CMPMCF |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x008 | LPTIM_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. | DOWNIE(1) | UPIE(1) | ARROKIE | CMPOKIE | EXTTRIGIE | ARRMIE | CMPMIE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00C | LPTIM_CFGR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ENC(1) | COUNTMODE | PRELOAD | WAVEPOL | WAVE | TIMOUT | TRIGEN | | Res. | TRIGSEL[2:0] | | | Res. | PRESC | | | Res. | TRGFLT | | Res. | CKFLT | | CKPOL | | CKSEL |
| | Reset value | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | | 0 | 0 | 0 | 0 | 0 |
| 0x010 | LPTIM_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. | RSTARE | COUNTRST | CNTSTRT | SNGSTRT | ENABLE |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| 0x014 | LPTIM_CMP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CMP[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x018 | LPTIM_ARR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ARR[15:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0x01C | LPTIM_CNT | 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 |

1. If LPTIM does not support encoder mode feature, this bit is reserved. Please refer to *Section 22.3: LPTIM implementation*.

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 23 Infrared interface (IRTIM)

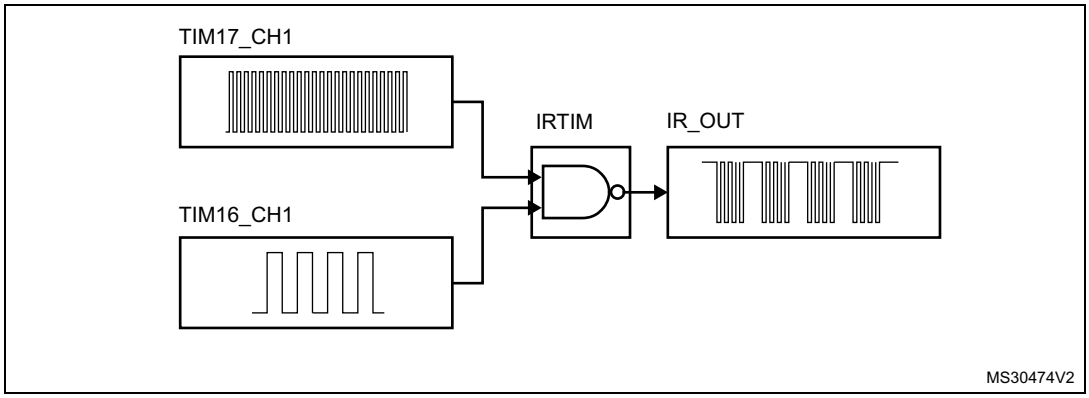
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections withTIM16 and TIM17 as shown in *Figure 256*.

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16_OC1) and TIM17 channel 1 (TIM17_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

**Figure 256. IRTIM internal hardware connections with TIM16 and TIM17**




All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 generates the modulation envelope.

The infrared function is output on the IR_OUT pin. The activation of this function is done through the GPIOx_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 pin) can be activated through the I2C_PB9_FMP bit in the SYSCFG_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

# 24 Real-time clock (RTC)

## 24.1 Introduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

The RTC includes also a periodic programmable wake-up flag with interrupt capability.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (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 Backup domain 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 reset).

## 24.2 RTC main features

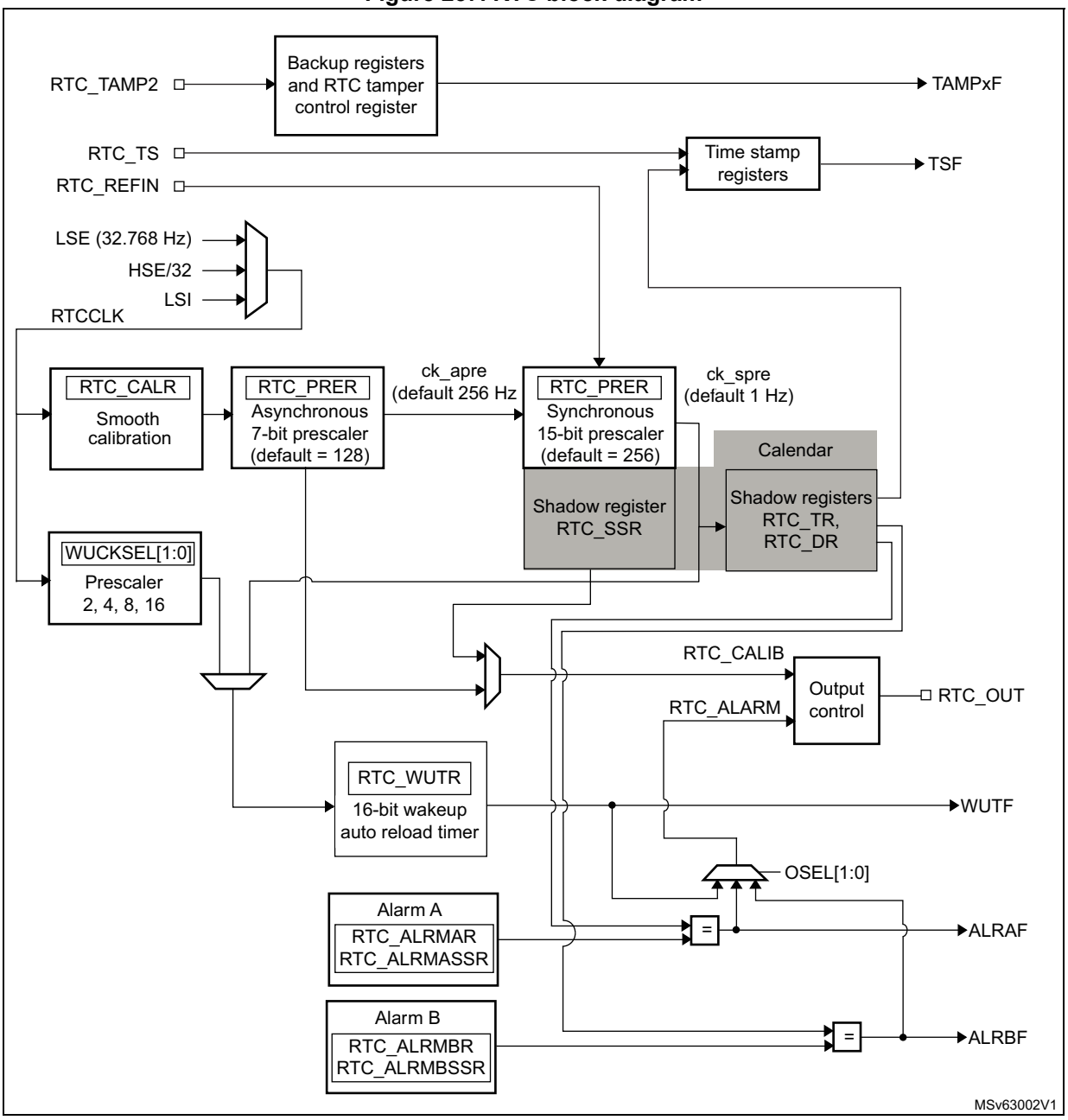
The RTC unit main features are the following (see *Figure 257: RTC block diagram*):

- 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 wake-up unit generating a periodic flag that triggers an automatic wake-up interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- 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
  - Alarm B
  - Wake-up interrupt
  - Time-stamp
  - Tamper detection
- 20 backup registers.

## 24.3 RTC functional description

### 24.3.1 RTC block diagram

**Figure 257. RTC block diagram**




The RTC includes:

- Tamper detection erases the backup registers.
- One timestamp event from I/O
- Tamper event detection can generate a timestamp event
- Timestamp can be generated when a switch to $V_{BAT}$ occurs

### 24.3.2 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to *Section 7: Reset and clock control (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 257: 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 wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 24.3.5: Periodic auto-wake-up* for details).

### 24.3.3 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 RTCCLK period, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ISR register is set (see *Section 24.6.4: RTC initialization and status*

*register (RTC_ISR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 1 RTCCLK period.

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.

### 24.3.4 Programmable alarms

The RTC unit provides programmable alarm: Alarm A and Alarm B. The description below is given for Alarm A, but can be translated in the same way for Alarm B.

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 (MSK1 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 and Alarm B (if enabled by bits OSEL[1:0] 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.

### 24.3.5 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up 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 wake-up 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 wake-up 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 wake-up timer on page 785*), the timer starts counting down.When the wake-up 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 wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the RTC_ALARM output provided it has been enabled through bits OSEL[1:0] 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 modes (Sleep, Stop and Standby) have no influence on the wake-up timer.

### 24.3.6 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 system reset, the RTC registers are protected against parasitic write access by clearing the DBP bit in the PWR_CR register (refer to the power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain 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_TAMPCR, RTC_BKPxR, RTC_OR and RTC_ISR[13:8].

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: 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 Backup domain reset default value (0x00).*

*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. The procedure below is given for Alarm A but can be translated in the same way for Alarm B.

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 wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ISR to make sure the access to wake-up 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 wake-up auto-reload value WUT[15:0], and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting. The WUTWF bit is cleared up to 2 RTCCLK clock cycles after WUTE is cleared, due to clock synchronization.

### 24.3.7 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 APB clock frequency ($f_{PCLK}$) must be equal to or greater than seven times the RTC clock frequency ($f_{RTCCLK}$). This ensures a secure behavior of the synchronization mechanism.

If the APB 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 APB 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_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. 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 1 RTCCLK period: 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 (Stop or Standby), 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 wake-up 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 on page 784*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 24.3.9: 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.*

### 24.3.8 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of 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 Backup domain 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 configuration register (RTC_TAMPCR), the RTC backup registers (RTC_BKPxR), the wake-up timer register (RTC_WUTR), the Alarm A and Alarm B registers (RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR), and the Option register (RTC_OR).

In addition, when it is clocked by the LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to the RTC clock section of the Reset and clock controller for details on the list of RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

### 24.3.9 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 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 output at 1 Hz. In this way, the frequency of the asynchronous prescaler 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 will delay 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 will advance the clock.

**Caution:** Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

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.

**Caution:** This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON=1.

### 24.3.10 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the synchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

*Note: RTC_REFIN clock detection is not available in Standby mode.*

### 24.3.11 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 adjustments (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, cal_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 CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[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); CALM[2]=1*

*causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[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 with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. 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, between 255 and 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.

### 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.

### 24.3.12 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 VBAT 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.*

*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.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 24.6.16: RTC tamper configuration register (RTC_TAMPCR)*.

## 24.3.13 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 the following purposes:

- erase the RTC backup registers (default configuration)
- generate an interrupt, capable to wake-up from Stop and Standby modes
- generate a hardware trigger for the low-power timers

### RTC backup registers

The backup registers (RTC_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs (see *Section 24.6.20: RTC backup registers (RTC_BKPxR)* and *Tamper detection initialization on page 791*) except if the TAMPxNOERASE bit is set, or if TAMPxMF is set in the RTC_TAMPCR register.

### 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 occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

When TAMPxMF is set:

A new tamper occurring 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). Setting TAMPIE is not allowed when one or more TAMPxMF 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. Setting TAMPxIE is not allowed when the corresponding TAMPxMF is set.

### Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMF bit in cleared in RTC_TAMPCR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMF bit is set, the TAMPxF flag is masked, and kept cleared in RTC_ISR register. This configuration allows to trig automatically the low-power timers in Stop mode, without requiring the system wakeup to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

### 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:** When using the edge detection, it is recommended to check by software the tamper pin level just after enabling the tamper detection (by reading the GPIO registers), and before writing sensitive values in the backup registers, to ensure that an active edge did not occur before enabling the tamper event detection.
When TAMPFLT="00" and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the RTC_TAMPx 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 input.

*Note: Tamper detection is still active when $V_{DD}$ power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the RTC_TAMPx 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 precharged 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 trade-off 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.*

### 24.3.14 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.

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 $f_{RTCCLK}/(256 * (PREDIV_A+1))$. 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. The 1 Hz output is affected when a shift operation is on going and may toggle during the shift operation (SHPF=1).

*Note: When COSEL bit is cleared, the RTC_CALIB output is the output of the 6th stage of the asynchronous prescaler.*

*When COSEL bit is set, the RTC_CALIB output is the output of the 8th stage of the synchronous prescaler.*

### 24.3.15 Alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm 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 output

The RTC_ALARM pin can be configured in output open drain or output push-pull using RTC_OR register.

*Note: Once the RTC_ALARM output is enabled, it has priority over RTC_CALIB (COE bit is don't care and must be kept cleared).*

## 24.4 RTC low-power modes

**Table 148. Effect of low-power modes on RTC**

| Mode | Description |
|---|---|
| Sleep | No effect<br>RTC interrupts cause the device to exit the Sleep mode. |
| Low-power run | No effect. |
| Low-power sleep | No effect. RTC interrupts cause the device to exit the Low-power sleep mode. |
| Stop 0 | The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Stop mode. |
| Stop 1 | |
| Stop 2 | |
| Standby | The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Standby mode. |
| Shutdown | The RTC remains active when the RTC clock source is LSE. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Shutdown mode. |

## 24.5 RTC interrupts

All RTC interrupts are connected to the EXTI controller. Refer to *Section 14: Extended interrupt and event controller (EXTI)*.

**Table 149. Interrupt control bits**

| Interrupt event | Event flag | Enable control bit | Exit from Sleep mode | Exit from Stop mode | Exit from Standby mode |
|---|---|---|---|---|---|
| Alarm A | ALRAF | ALRAIE | Yes | Yes(1) | Yes(1) |
| Alarm B | ALRBF | ALRBIE | Yes | Yes(1) | Yes(1) |
| RTC_TS input (timestamp) | TSF | TSIE | Yes | Yes(1) | Yes(1) |
| RTC_TAMP2 input detection | TAMP2F | TAMPIE | Yes | Yes(1) | Yes(1) |
| Wakeup timer interrupt | WUTF | WUTIE | Yes | Yes(1) | Yes(1) |

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.

## 24.6 RTC registers

Refer to *Section 1.2 on page 51* of the reference manual for a list of abbreviations used in register descriptions.

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

### 24.6.1 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 on page 784* and *Reading the calendar on page 785*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 784*.

Address offset: 0x00

Backup domain 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

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

### 24.6.2 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 on page 784* and *Reading the calendar on page 785*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 784*.

Address offset: 0x04

Backup domain 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[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

Bits 3:0 **DU[3:0]**: Date units in BCD format

### 24.6.3 RTC control register (RTC_CR)

Address offset: 0x08

Backup domain 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 | ALRBIE | ALRAIE | TSE | WUTE | ALRBE | ALRAE | Res. | FMT | BYPS HAD | REFCKON | TSEDGE | WUCKSEL[2:0] | | |
| rw | rw | rw | 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 disabled
1: internal event timestamp enabled

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: Alarm B output enabled
11: Wake-up 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/ALRBF/WUTF is asserted (depending on OSEL[1:0])
1: The pin is low when ALRAF/ALRBF/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 (with default prescaler setting)
1: Calibration output is 1 Hz (with default prescaler setting)

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 24.3.14: 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, 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 outside initialization mode.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set, 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 outside initialization mode.

Bit 15 **TSIE**: Time-stamp interrupt enable

0: Time-stamp Interrupt disable
1: Time-stamp Interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable

0: Wake-up timer interrupt disabled
1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: *Alarm B interrupt enable*

0: Alarm B Interrupt disable
1: Alarm B Interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable

0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wake-up timer enable

0: Wake-up timer disabled
1: Wake-up timer enabled

*Note: When the wake-up timer is disabled, wait for WUTWF=1 before enabling it again.*

Bit 9 **ALRBE**: *Alarm B enable*

0: Alarm B disabled
1: Alarm B enabled

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 APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to '1'.*

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)

0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled

*Note: PREDIV_S must be 0x00FF.*

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]**: Wake-up 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: Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).*

*WUT = Wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].*

*Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1.*

*It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.*

*ADD1H and SUB1H changes are effective in the next second.*

*This register is write protected. The write access procedure is described in RTC register write protection on page 784.*

**Caution:** TSE must be reset when TSEDGE is changed to avoid spuriously setting of TSF.

### 24.6.4 RTC initialization and status register (RTC_ISR)

This register is write protected (except for RTC_ISR[13:8] bits). The write access procedure is described in *RTC register write protection on page 784*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to ‘0’

| 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. | TAMP2F | Res. | TSOVF | TSF | WUTF | ALRBF | ALRAF | INIT | INITF | RSF | INITS | SHPF | WUTWF | ALRBWF | ALRAWF |
| | rc_w0 | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rw | r | rc_w0 | r | r | r | r | r |

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

Bit 17 **ITSF**: Internal tTime-stamp flag

This flag is set by hardware when a time-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 **TAMP2F**: RTC_TAMP2 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP2 input.

It is cleared by software writing 0

Bit 13 Reserved, must be kept at reset value.

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 10 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up 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 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm B register (RTC_ALRMBR).

This flag is cleared by software by writing 0.

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_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register 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 (Backup domain 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**: Wake-up timer write flag

This bit is set by hardware up to 2 RTCCLK cycles after the WUTE bit has been set to 0 in RTC_CR, and is cleared up to 2 RTCCLK cycles after the WUTE bit has been set to 1. The wake-up timer values can be changed when WUTE bit is cleared and WUTWF is set.

0: Wake-up timer configuration update not allowed

1: Wake-up timer configuration update allowed

Bit 1 **ALRBWF**: Alarm B write flag

This bit is set by hardware when Alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm B update not allowed

1: Alarm B update allowed

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: The bits ALRAF, ALRBF, WUTF and TSF are cleared 2 APB clock cycles after programming them to 0.*

### 24.6.5 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 on page 784*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 784*.

Address offset: 0x10

Backup domain 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)

### 24.6.6 RTC wake-up 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 on page 784*.

Address offset: 0x14

Backup domain 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]**: Wake-up auto-reload value bits

When the wake-up 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 wake-up 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.

### 24.6.7 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 *RTC register write protection on page 784*.

Address offset: 0x1C

Backup domain 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.

### 24.6.8 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 784*.

Address offset: 0x20

Backup domain 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 B date mask
0: Alarm B set if the date and day match
1: Date and day don’t care in Alarm B 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 B hours mask
0: Alarm B set if the hours match
1: Hours don’t care in Alarm B 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 B minutes mask
0: Alarm B set if the minutes match
1: Minutes don’t care in Alarm B 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 B seconds mask
0: Alarm B set if the seconds match
1: Seconds don’t care in Alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

### 24.6.9 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.

### 24.6.10 RTC sub second register (RTC_SSR)

Address offset: 0x28

Backup domain 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 |

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

Bits 15:0 **SS[15:0]**: Sub second value

SS[15:0] is the value in the synchronous prescaler 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.*

### 24.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 784.*

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

| 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 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: 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 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.*

### 24.6.12 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

Backup domain 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.

### 24.6.13 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

Backup domain 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[2:0] | | | 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[2:0]**: 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

Bits 3:0 **DU[3:0]**: Date units in BCD format

### 24.6.14 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

Backup domain 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[15:0]**: Sub second value

SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

### 24.6.15 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 784.*

Address offset: 0x3C

Backup domain 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 |

Bits 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 increased 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 24.3.11: 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 stuck at “00” when CALW8=’1’. Refer to Section 24.3.11: 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 stuck at ‘0’ when CALW16=’1’. Refer to Section 24.3.11: 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 24.3.11: RTC smooth digital calibration on page 788*.

### 24.6.16 RTC tamper configuration register (RTC_TAMPCR)

Address offset: 0x40

Backup domain 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. | TAMP2 MF | TAMP2 NO ERASE | TAMP2 IE | Res. | Res. | Res. |
| | | | | | | | | | | rw | rw | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TAMP PUDIS | TAMPPRCH [1:0] | | TAMPFLT[1:0] | | TAMPFREQ[2:0] | | | TAMP TS | Res. | Res. | TAMP2 TRG | TAMP2 E | TAMPI E | Res. | Res. |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | | | rw | rw | rw | | |

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

Bits 24:22 Reserved, must be kept at reset value.

Bit 21 **TAMP2MF**: Tamper 2 mask flag

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers are not erased.

*Note: The Tamper 2 interrupt must not be enabled when TAMP2MF is set.*

Bit 20 **TAMP2NOERASE**: Tamper 2 no erase

0: Tamper 2 event erases the backup registers.

1: Tamper 2 event does not erase the backup registers.

Bit 19 **TAMP2IE**: Tamper 2 interrupt enable

0: Tamper 2 interrupt is disabled if TAMPIE = 0.

1: Tamper 2 interrupt enabled.

Bits 18:16 Reserved, must be kept at reset value.

Bit 15 **TAMPPUDIS**: RTC_TAMPx pull-up disable

This bit determines if each of the RTC_TAMPx pins are precharged 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 6:5 Reserved, must be kept at reset value.

Bit 4 **TAMP2TRG**: Active level for RTC_TAMP2 input

if TAMPFLT != 00:
0: RTC_TAMP2 input staying low triggers a tamper detection event.
1: RTC_TAMP2 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP2 input rising edge triggers a tamper detection event.
1: RTC_TAMP2 input falling edge triggers a tamper detection event.

Bit 3 **TAMP2E**: RTC_TAMP2 input detection enable

0: RTC_TAMP2 detection disabled
1: RTC_TAMP2 detection enabled

Bit 2 **TAMPIE**: Tamper interrupt enable

0: Tamper interrupt disabled
1: Tamper interrupt enabled.

*Note: This bit enables the interrupt for all tamper pins events, whatever TAMPxIE level. If this bit is cleared, each tamper event interrupt can be individually enabled by setting TAMPxIE.*

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

**Caution:** When TAMPFLT = 0, TAMPxE must be reset when TAMPxTRG is changed to avoid spuriously setting TAMPxF.

### 24.6.17 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 on page 784*

Address offset: 0x44

Backup domain 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 |

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

Bits 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.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

### 24.6.18 RTC alarm B sub second register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register is write protected.The write access procedure is described in *Section : RTC register write protection*.

Address offset: 0x48

Backup domain 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 |

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

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0x0: No comparison on sub seconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
0x1: SS[14:1] are don't care in Alarm B comparison. Only SS[0] is compared.
0x2: SS[14:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.
0x3: SS[14:3] are don't care in Alarm B comparison. Only SS[2:0] are compared.
...
0xC: SS[14:12] are don't care in Alarm B comparison. SS[11:0] are compared.
0xD: SS[14:13] are don't care in Alarm B comparison. SS[12:0] are compared.
0xE: SS[14] is don't care in Alarm B comparison. SS[13:0] are compared.
0xF: 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.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

### 24.6.19 RTC option register (RTC_OR)

Address offset: 0x4C

Backup domain 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 | Res. |
| | | | | | | | | | | | | | | 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 PB2 as follows:

**RTC_OUT_RMP = ‘0’: Reserved**

**RTC_OUT_RMP = ‘1’:**

If OSEL /= ‘00’ and COE = ‘0’: RTC_ALARM is output on PB2

If OSEL = ‘00’ and COE = ‘1’: RTC_CALIB is output on PB2

If OSEL /= ‘00’ and COE = ‘1’: Reserved

Bit 0 Reserved, must be kept at reset value.

### 24.6.20 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0x9C

Backup domain 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 $V_{BAT}$ when $V_{DD}$ is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode.

This register is reset on a tamper detection event, as long as TAMPxF=1.

## 24.6.21 RTC register map

**Table 150. RTC register map and reset values**

| Offset | Register name | 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 | ALRBIE | ALRAIE | TSE | WUTE | ALRBE | ALRAE | Res. | FMT | BYPSHAD | REFCKON | 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 | 0 | 0 |
| 0x0C | **RTC_ISR** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ITSF | RECALPF | Res. | TAMP2F | Res. | TSOVF | TSF | WUTF | ALRBF | ALRAF | INIT | INITF | RSF | INITS | SHPF | WUT WF | ALRBWF | ALRAWF |
| | Reset value | | | | | | | | | | | | | | | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0x10 | **RTC_PRER** | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREDIV_A[6:0] | | | | | | | PREDIV_S[14:0] | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 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 |
| 0x20 | **RTC_ALRMBR** | MSK4 | WDSEL | DT[1:0] | | DU[3:0] | | | | MSK3 | PM | HT[1:0] | | HU[3:0] | | | | MSK2 | MNT[2:0] | | | MNU[3:0] | | | | MSK2 | 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 | | | | | | | |
| | 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 |

**Table 150. RTC register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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. | TAMP2MF | TAMP2NOERASE | TAMP2IE | Res. | Res. | Res. | TAMPPUDIS | TAMPPRCH[1:0] | | TAMPFLT[1:0] | | TAMPFREQ[2:0] | | | TAMPTS | Res. | Res. | TAMP2TRG | TAMP2E | TAMPIE | Res. | Res. |
| | 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 |
| 0x48 | **RTC_ ALRMBSSR** | 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 | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | |
| 0x50 to 0x9C | **RTC_BKP0R** | 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 |
| | to **RTC_BKP19R** | 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 on page 56* for the register boundary addresses.

# 25 Independent watchdog (IWDG)

## 25.1 Introduction

The devices feature an embedded watchdog peripheral (IWDG) that offers a combination of high safety level, timing accuracy, and flexibility of use. This peripheral detects and solves malfunctions due to software failure, and triggers a system reset when the counter reaches a given timeout value.

The independent watchdog is clocked by its own dedicated low-speed clock (LSI), and stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to *Section 26: System window watchdog (WWDG)*.

## 25.2 IWDG main features

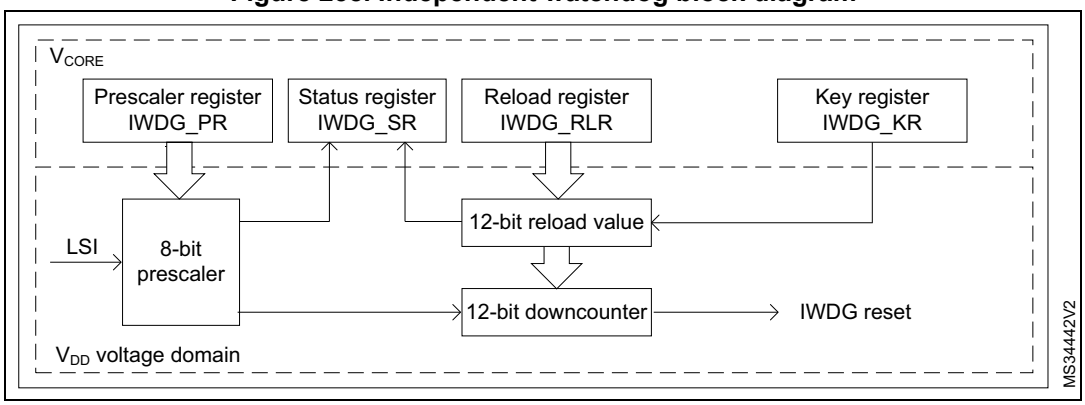
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog is activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog is activated) if the downcounter is reloaded outside the window

## 25.3 IWDG functional description

### 25.3.1 IWDG block diagram

*Figure 258* shows the functional blocks of the independent watchdog module.

**Figure 258. Independent watchdog block diagram**




1. The register interface is located in the $V_{CORE}$ voltage domain. The watchdog function is located in the $V_{DD}$ voltage domain, still functional in Stop and Standby modes.

When the independent watchdog is started by writing the value 0x0000 CCCC in the *IWDG 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 key register (IWDG_KR)*, the IWDG_RLR value is reloaded in the counter, and the watchdog reset is prevented.

Once running, the IWDG cannot be stopped.

### 25.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than the value stored in the *IWDG window register (IWDG_WINR)*, a reset is provided.

The default value of the *IWDG window register (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 to reset the downcounter to the *IWDG reload register (IWDG_RLR)* value, and to 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 key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically refreshes the counter value in the *IWDG reload register (IWDG_RLR)*.

*Note: Writing the window value allows the counter value to be refreshed by the RLR when the IWDG status register (IWDG_SR) is set to 0x0000 0000.*

#### Configuring the IWDG when the window option is disabled

When the window option is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA).

### 25.3.3 Hardware watchdog

If this feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the *IWDG key register (IWDG_KR)* is written by the software before the counter reaches the end of count, or if the downcounter is reloaded inside the window.

### 25.3.4 Low-power freeze

Depending on the IWDG_STOP and IWDG_STBY options configuration, the IWDG can continue counting or not during the Stop mode and the Standby mode, respectively. If the IWDG is kept running during Stop or Standby modes, it can wake up the device from this mode. Refer to *User and read protection option bytes* for more details.

### 25.3.5 Register access protection

Write access to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, and *IWDG window register (IWDG_WINR)* is protected. To modify it, first write the code 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the sequence, and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an ongoing update of the prescaler, or of the downcounter reload value, or of the window value.

### 25.3.6 Debug mode

When the CPU1 enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on the configuration of the corresponding bit in DBGMCU freeze register.

## 25.4 IWDG registers

Refer to *Section 1.2 on page 51* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 25.4.1 IWDG 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 enable access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers (see *Section 25.3.5: Register access protection*)

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

### 25.4.2 IWDG 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 25.3.5: Register access protection*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (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 $V_{DD}$ 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 status register (IWDG_SR) is reset.*

### 25.4.3 IWDG 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.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Register access protection*. 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 key register (IWDG_KR)*. 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 status register (IWDG_SR)* must be reset 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 up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.*

### 25.4.4 IWDG 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 five cycles).

Window value can be updated only when WVU bit is reset.

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 five 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 five cycles).

Prescaler value can be updated only when PVU bit 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.*

### 25.4.5 IWDG 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.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 25.3.5*, they contain the high limit of the window value to be compared with 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 status register (IWDG_SR)* 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 status register (IWDG_SR) is reset.*

### 25.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

**Table 151. IWDG register map and reset values**

| Offset | Register name | 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 on page 56* for the register boundary addresses.

# 26 System window watchdog (WWDG)

## 26.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. An MCU reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

## 26.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 260*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

## 26.3 WWDG functional description

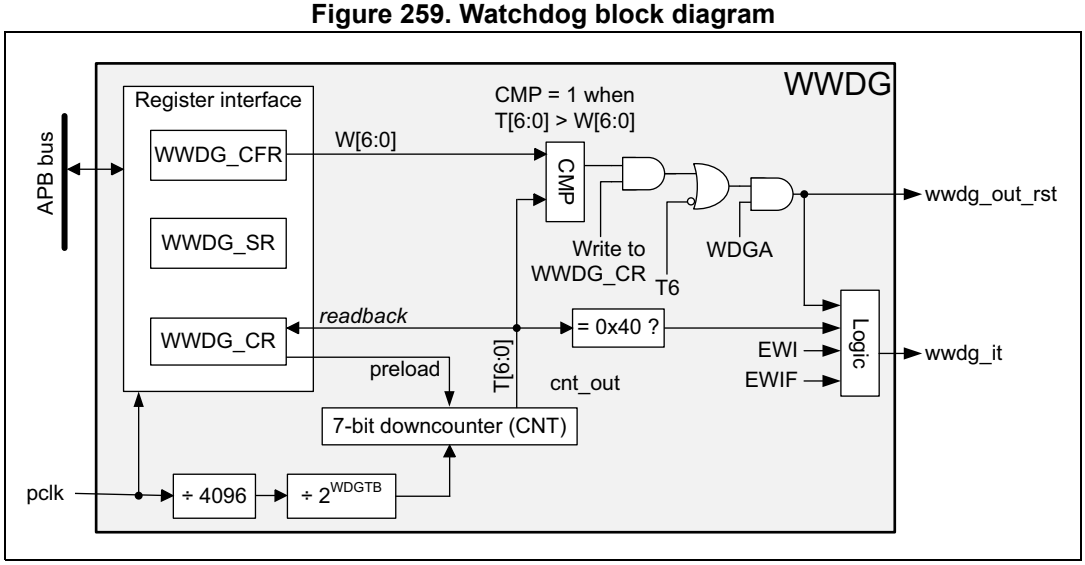
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 259* for the WWDG block diagram.

### 26.3.1 WWDG block diagram

**Figure 259. Watchdog block diagram**




### 26.3.2 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

### 26.3.3 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 260*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 260* describes the window watchdog process.

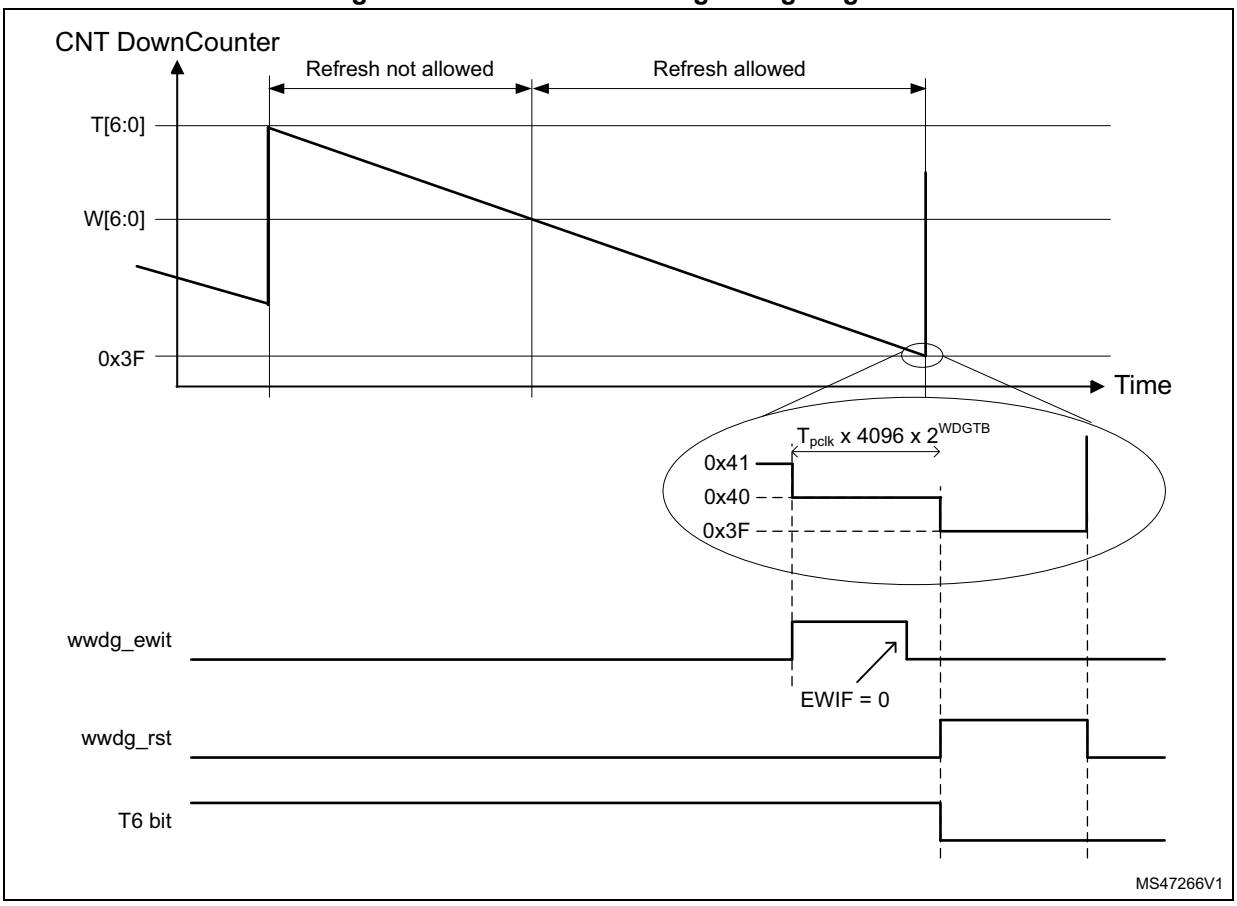
*Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).*

### 26.3.4 How to program the watchdog timeout

Use the formula in *Figure 260* to calculate the WWDG timeout.

**Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.**

**Figure 260. Window watchdog timing diagram**




The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \qquad (ms)$$

where:

$t_{WWDG}$: WWDG timeout

$t_{PCLK}$: APB clock period measured in ms

4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of $t_{WWDG}$.

### 26.3.5 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to *Section 32: Debug support (DBG)*.

## 26.4 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

*Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.*

## 26.5 WWDG registers

Refer to *Section 1.2 on page 51* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

### 26.5.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

| 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. | WDGA | T[6:0] | | | | | | |
| | | | | | | | | rs | rw | rw | rw | rw | rw | rw | rw |

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

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

## 26.5.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

| 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. | WDGTB[2:0] | | | Res. | EWI | Res. | Res. | W[6:0] | | | | | | |
| | | rw | rw | rw | | rs | | | rw | rw | rw | rw | rw | rw | rw |

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

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

### 26.5.3 WWDG status register (WWDG_SR)

Address offset: 0x008

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. | EWIF |
| | | | | | | | | | | | | | | | rc_w0 |

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

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

### 26.5.4 WWDG register map

The following table gives the WWDG register map and reset values.

**Table 152. WWDG register map and reset values**

| Offset | Register name | 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 | WWDG_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. | WDGA | T[6:0] | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0x004 | WWDG_CFR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WDGTB [2:0] | | | Res. | EWI | Res. | Res. | W[6:0] | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0x008 | WWDG_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. | Res. | Res. | EWIF |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 27 Inter-integrated circuit (I2C) interface

## 27.1 Introduction

The $I^2C$ (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial $I^2C$ bus. It provides multimaster 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+).

The $I^2C$ bus interface is also SMBus (system management bus) and PMBus® (power management bus) compatible.

DMA can be used to reduce CPU overload.

## 27.2 I2C main features

- $I^2C$ bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster 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 slave 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 features are also available, depending upon product implementation (see *Section 27.3*):

- SMBus specification rev 3.0 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 1.3 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming

- Wake-up from Stop mode on address match

## 27.3 I2C implementation

**Table 153. STM32WB50CG/30CE I2C implementation**

| I2C features[1] | I2C1 |
|---|---|
| 7-bit addressing mode | X |
| 10-bit addressing mode | X |
| Standard mode (up to 100 kbit/s) | X |
| Fast mode (up to 400 kbit/s) | X |
| Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s) | X |
| Independent clock | X |
| SMBus/PMbus | X |
| Wake-up from Stop 0 / Stop 1 mode on address match | X |
| Wake-up from Stop 2 mode on address match | - |

1. X = supported.

## 27.4 I2C functional description

In addition to receiving and transmitting data, this interface converts them 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.

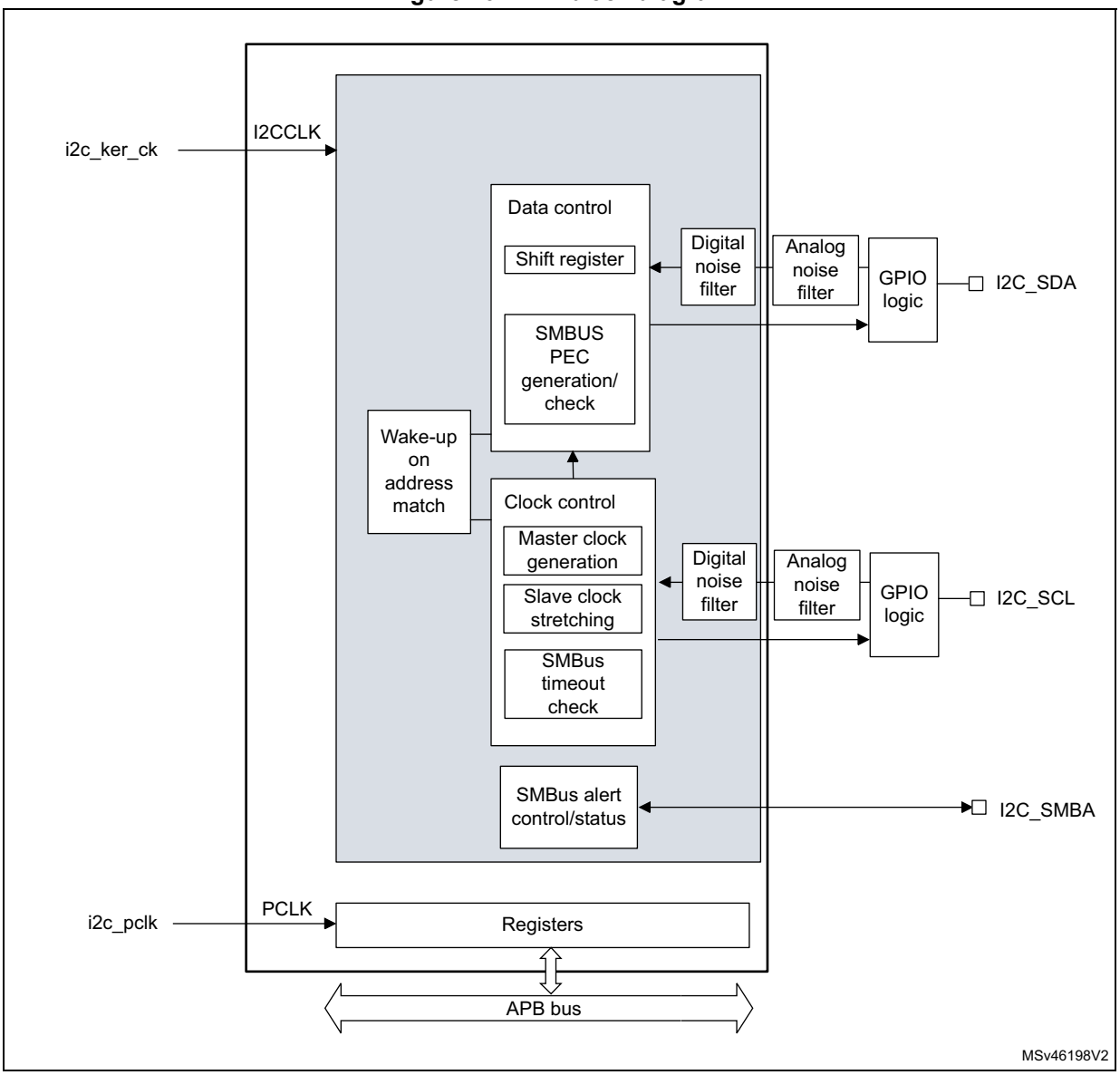
This interface can also be connected to an SMBus with data (SDA) and clock (SCL) pins.

If the SMBus feature is supported, the optional SMBus Alert pin (SMBA) is also available.

### 27.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 261*.

**Figure 261. I2C block diagram**




The I2C is clocked by an independent clock source, which allows the I2C to operate independently from the PCLK frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG). *Refer to Section 27.3: I2C implementation.*

### 27.4.2 I2C pins and internal signals

**Table 154. I2C input/output pins**

| Pin name | Signal type | Description |
|---|---|---|
| I2C_SDA | Bidirectional | I2C data |
| I2C_SCL | Bidirectional | I2C clock |
| I2C_SMBA | Bidirectional | SMBus alert |

**Table 155. I2C internal input/output signals**

| Internal signal name | Signal type | Description |
|---|---|---|
| i2c_ker_ck | Input | I2C kernel clock, also named I2CCLK in this document |
| i2c_pclk | Input | I2C APB clock |
| i2c_it | Output | I2C interrupts, refer to *Table 168* for the list of interrupt sources |
| i2c_rx_dma | Output | I2C receive data DMA request (I2C_RX) |
| i2c_tx_dma | Output | I2C transmit data DMA request (I2C_TX) |

### 27.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period $t_{I2CCLK}$ must respect the following conditions:

- $t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$
- $t_{I2CCLK} < t_{HIGH}$

with:

$t_{LOW}$: SCL low time and $t_{HIGH}$: SCL high time

$t_{filters}$: when enabled, sum of the delays brought by the analog and by the digital filters.

The digital filter delay is DNF x $t_{I2CCLK}$.

The PCLK clock period $t_{PCLK}$ must respect the condition:

- $t_{PCLK} < 4 / 3\ t_{SCL}$ ($t_{SCL}$: SCL period)

**Caution:** When the I2C kernel is clocked by PCLK, this clock must respect the conditions for $t_{I2CCLK}$.

### 27.4.4 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

### Communication flow

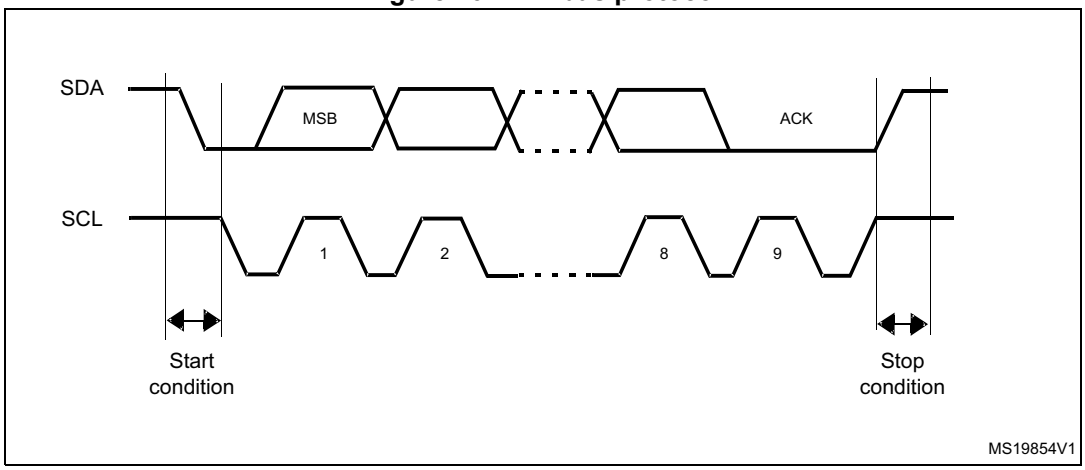
In master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition, and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In slave 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 be enabled also by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contains the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in master mode.

A ninth clock pulse follows the eight clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter (see *Figure 262*).

**Figure 262. I$^2$C bus protocol**




Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

## 27.4.5 I2C initialization

### Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller, 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$^2$C performs a software reset. Refer to *Section 27.4.6* for more details.

### Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This filter is compliant with the I$^2$C specification, which requires the suppression of spikes with pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The user 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 one to fifteen I2CCLK periods.

**Table 156. Comparison of analog vs. digital filters**

| - | Analog filter | Digital filter |
|---|---|---|
| Pulse width of suppressed spikes | ≥ 50 ns | Programmable length, from one to fifteen I2C peripheral clocks |
| Benefits | Available in Stop mode | – Programmable length: extra filtering capability versus standard requirements<br>– Stable length |
| Drawbacks | Variation vs. temperature, voltage, process | Wake-up from Stop mode on address match is not available when the digital filter is enabled |

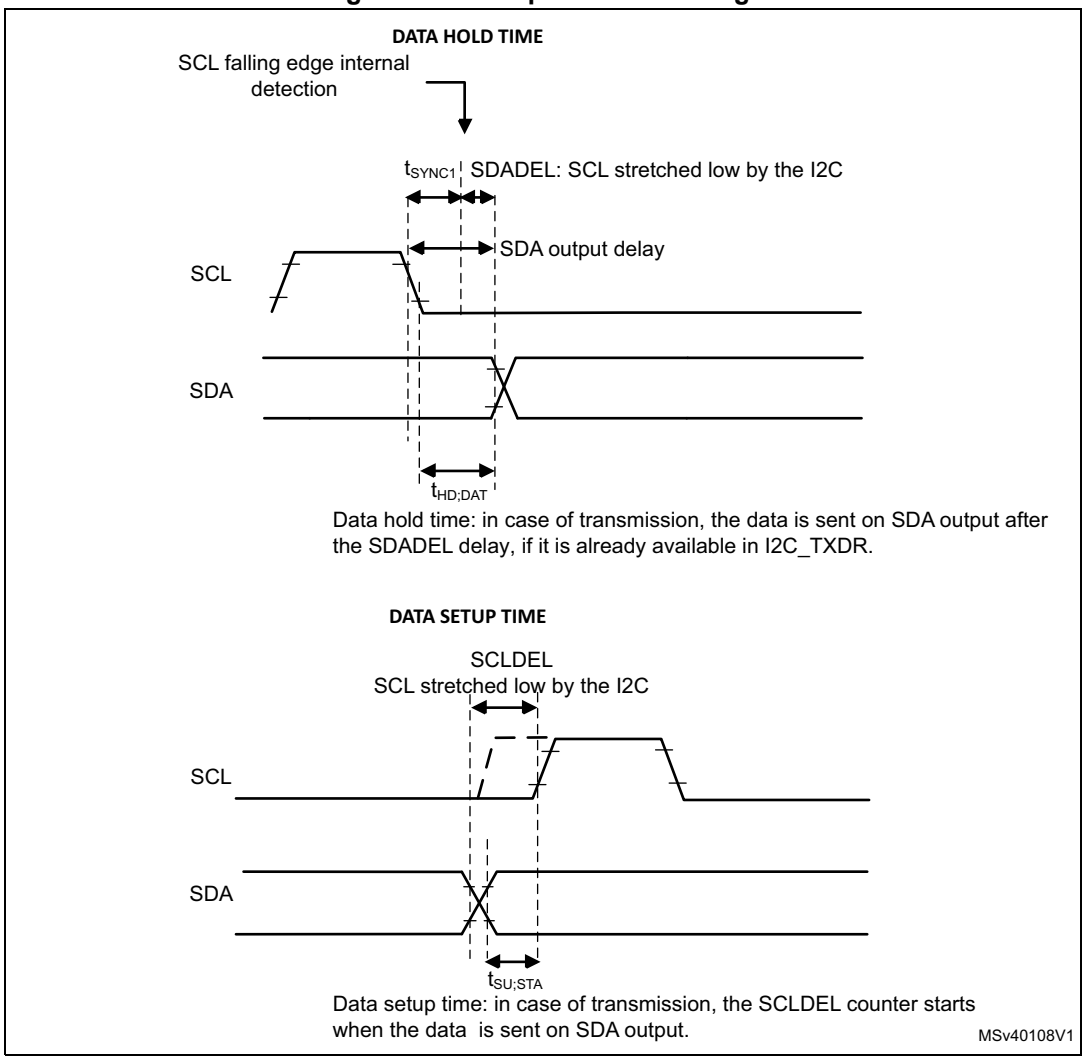
**Caution:** The filter configuration cannot be changed when the I2C is enabled.

### I2C timings

The timings must be configured to guarantee correct data hold and setup times, in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window.

**Figure 263. Setup and hold timings**




When the SCL falling edge is internally detected, a delay ($t_{SDADEL}$, impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output: $t_{SDADEL}$ = SDADEL x $t_{PRESC}$ + $t_{I2CCLK}$, where $t_{PRESC}$ = (PRESC + 1) x $t_{I2CCLK}$.

The total SDA output delay is:

$t_{SYNC1}$ + {[SDADEL x (PRESC + 1) + 1] x $t_{I2CCLK}$}

$t_{SYNC1}$ duration depends upon:

- SCL falling slope
- When enabled, input delay brought by the analog filter: $t_{AF(min)} < t_{AF} < t_{AF(max)}$
- When enabled, input delay brought by the digital filter: $t_{DNF}$ = DNF x $t_{I2CCLK}$
- Delay due to SCL synchronization to I2CCLK clock (two to three I2CCLK periods)

To bridge the undefined region of the SCL falling edge, the user 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 the device datasheet for* $t_{AF}$ *values.*

The maximum $t_{HD;DAT}$ can be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, 0.45 µs for Fast-mode Plus. It must be lower 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. In this case the previous equation becomes:

$$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}.$$

*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 157* for $t_f$, $t_r$, $t_{HD;DAT}$, and $t_{VD;DAT}$ standard values.

- After $t_{SDADEL}$, or after sending SDA output when the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, 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 $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

$$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1)] \times t_{I2CCLK}]\} - 1 \leq SCLDEL$$

Refer to *Table 157* for $t_r$ and $t_{SU;DAT}$ standard values.

The SDA and SCL transition time values to use 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.

*Note:* *At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x* $t_{I2CCLK}$*, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.*

If NOSTRETCH = 1 in slave mode, the SCL is not stretched, hence the SDADEL must be programmed so that it guarantees a sufficient setup time.

**Table 157. I²C-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 master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bit fields 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 $t_{HIGH}$.

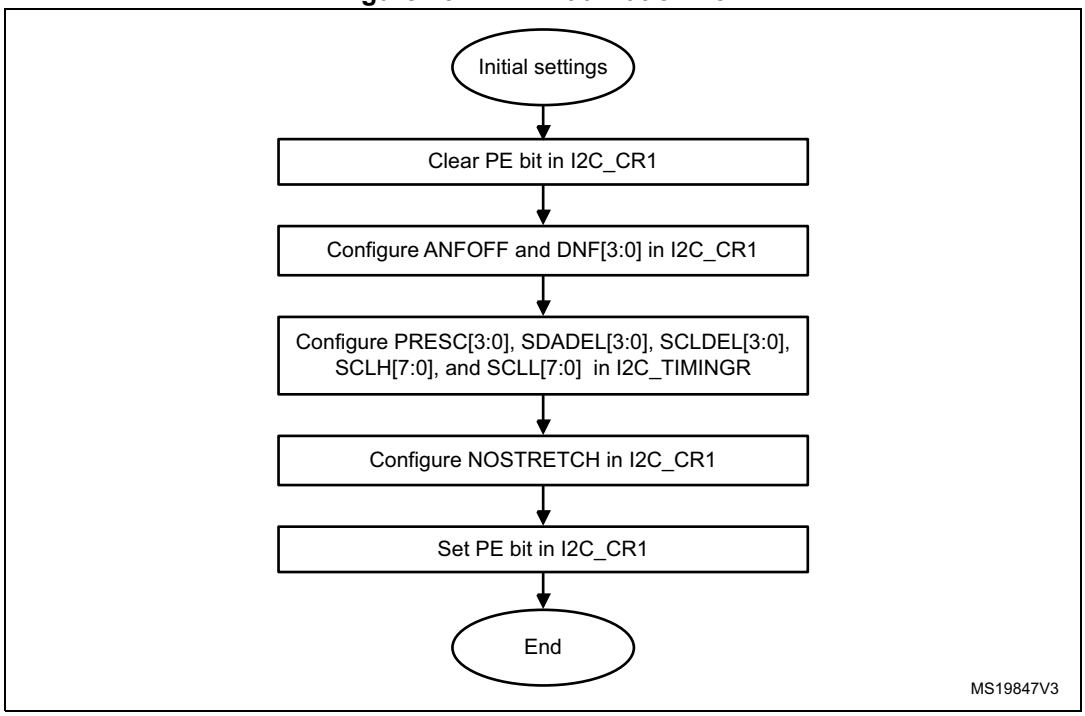
Refer to *I2C master initialization* for more details.

**Caution:** Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to *I2C slave initialization* for more details.

**Caution:** Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

**Figure 264. I2C initialization flow**




## 27.4.6 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.

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

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 three APB clock cycles to perform the software reset. This is ensured by the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

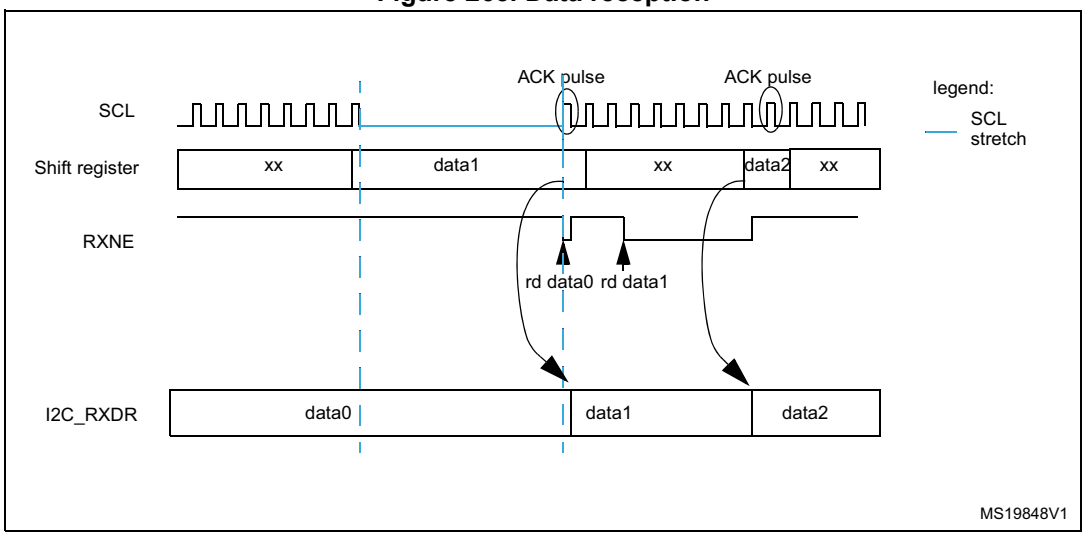
## 27.4.7 Data transfer

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 eighth 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 eighth and ninth SCL pulse (before the acknowledge pulse).

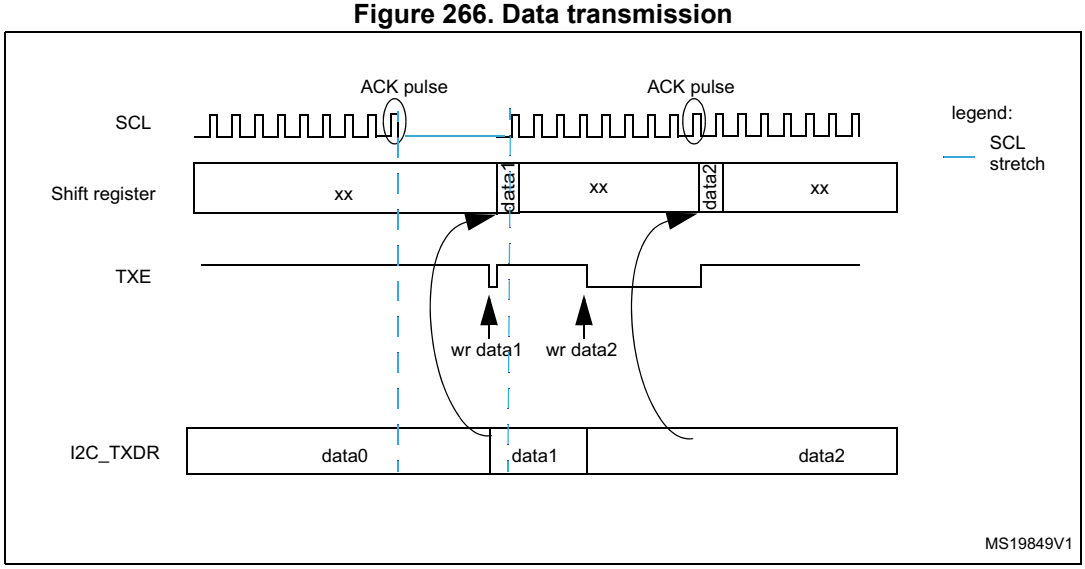
**Figure 265. Data reception**



### Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth 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 ninth SCL pulse.

**Figure 266. Data transmission**




### Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default, it is disabled in slave mode.It can be enabled by software by setting the SBC (slave byte control) bit in the I2C_CR1 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, the TCR flag is set when the number of bytes programmed in NBYTES is 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 master mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field is transferred.
- **Software end** (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 is 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 master wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

**Table 158. I2C configuration**

| Function | SBC bit | RELOAD bit | AUTOEND bit |
|---|---|---|---|
| Master Tx/Rx NBYTES + STOP | x | 0 | 1 |
| Master Tx/Rx + NBYTES + RESTART | x | 0 | 0 |
| Slave Tx/Rx, all received bytes ACKed | 0 | x | x |
| Slave Rx with ACK control | 1 | 1 | x |

## 27.4.8 I2C slave mode

### I2C slave initialization

To work in slave mode, the user must enable at least one slave address. Registers I2C_OAR1 and I2C_OAR2 are available to program the slave 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 slave addresses are required, the second slave address OA2 can be configured. Up to seven 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 slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, to perform software actions. If the master 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, the user must read the ADDCODE[6:0] bits in the I2C_ISR register to check which address matched. DIR flag must also be checked to know the transfer direction.

### Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave 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 Slave 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 $[(SDADEL + SCLDEL + 1) \times (PRESC+ 1) + 1] \times t_{I2CCLK}$.

### Slave without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave 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 the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures 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 ninth 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.

### Slave byte control mode

To allow byte ACK control in slave reception mode, the Slave 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.

The Reload mode must be selected to allow byte ACK control in slave 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 eighth and ninth SCL pulses. The user 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 the next byte can be received.

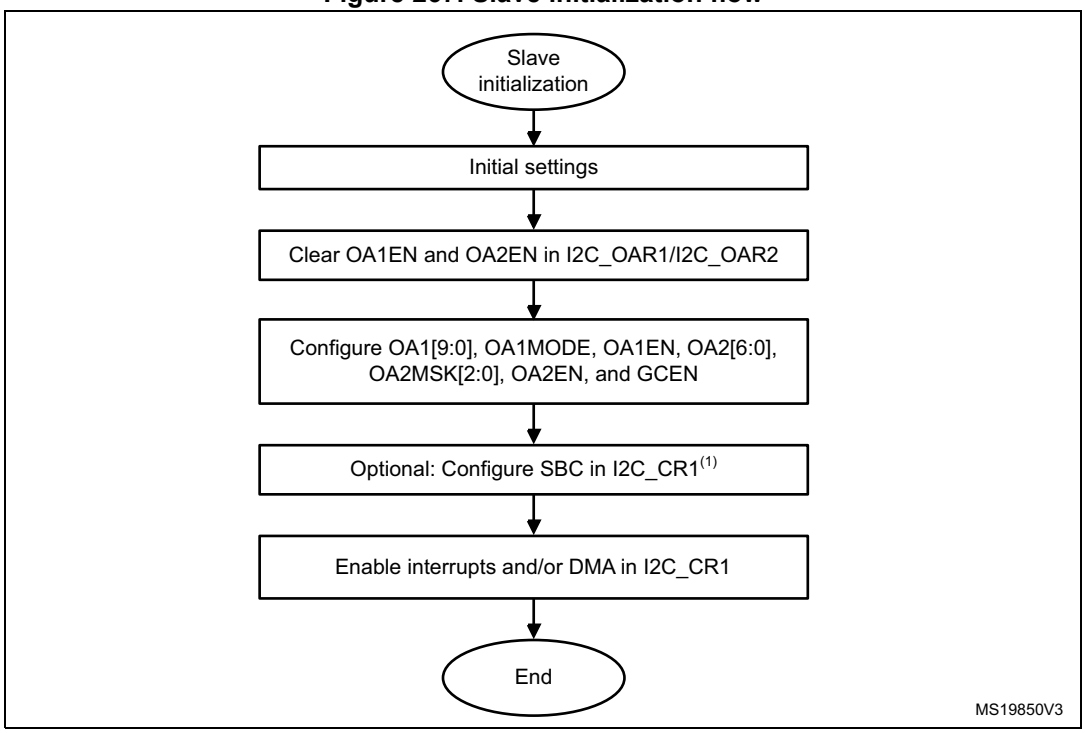
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

*Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR = 1.*

*The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.*

**Caution:** The Slave byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

**Figure 267. Slave initialization flow**




1. SBC must be set to support SMBus features.

### Slave 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 slave automatically releases the SCL and SDA lines to let the master 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 slave address is received (ADDR = 1), the user 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 Slave 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 the user cannot flush the I2C_TXDR register content in the ADDR subroutine, 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 one 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 a TXIS event (transmit interrupt or transmit DMA request) is needed, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

**Figure 268. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0**

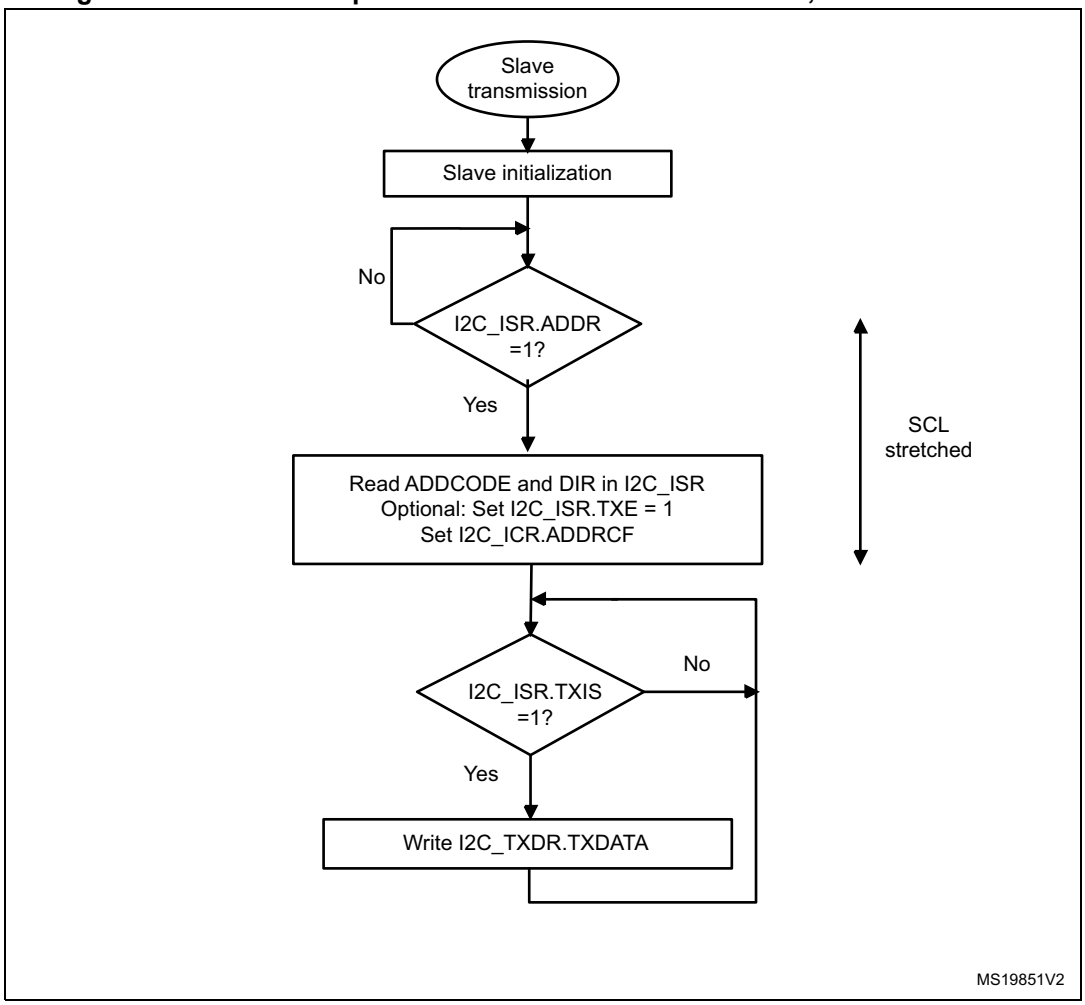


**Figure 269. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1**

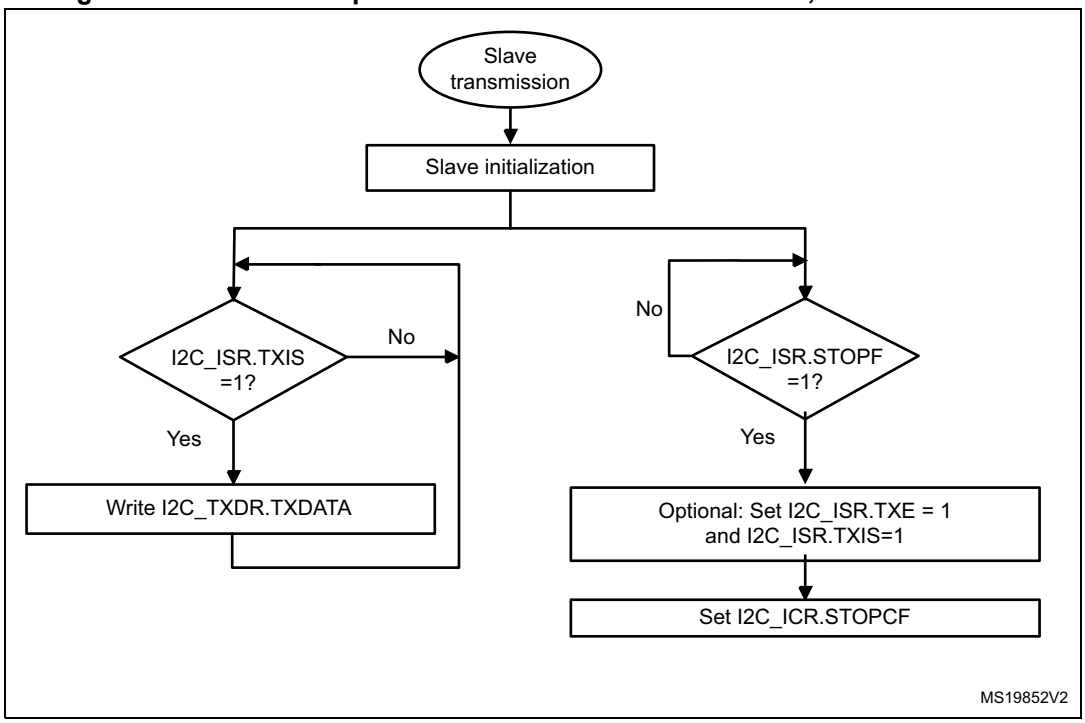
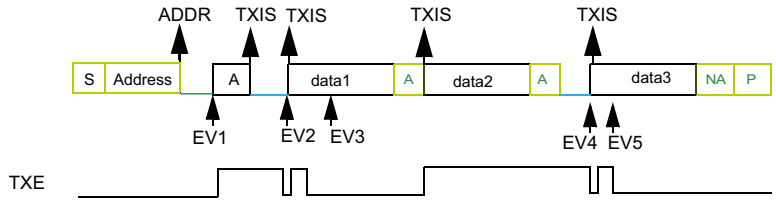


**Figure 270. Transfer bus diagrams for I2C slave transmitter (mandatory events only)**

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF

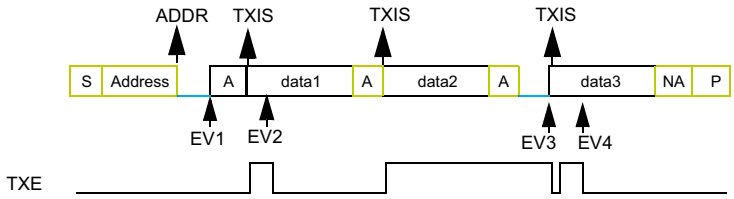
EV2: TXIS ISR: wr data1

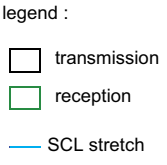
EV3: TXIS ISR: wr data2

EV4: TXIS ISR: wr data3

EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



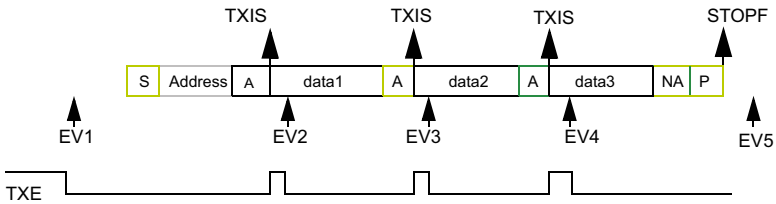

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF

EV2: TXIS ISR: wr data2

EV3: TXIS ISR: wr data3

EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:



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

MS19853V2

### Slave 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 271. Transfer sequence flow for slave receiver with NOSTRETCH = 0**

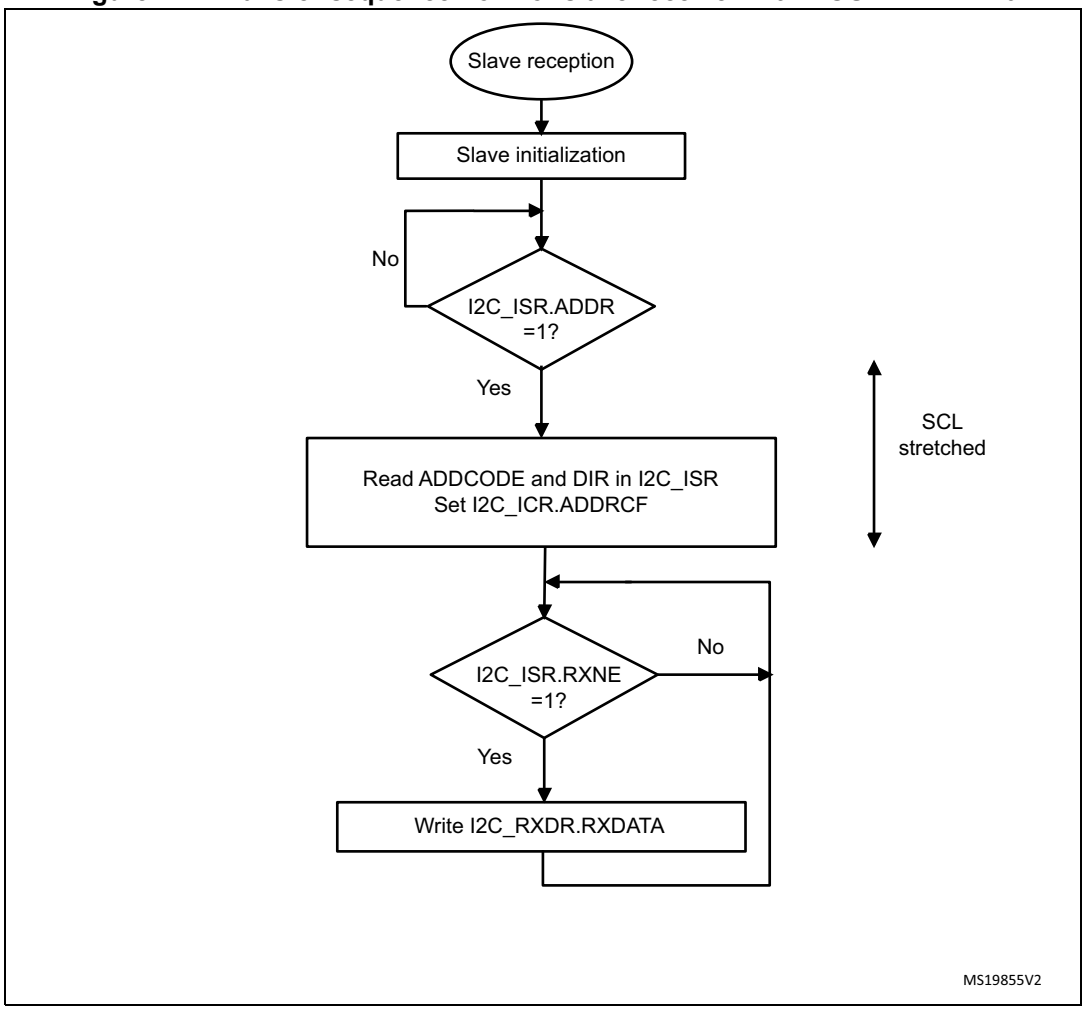


**Figure 272. Transfer sequence flow for slave receiver with NOSTRETCH = 1**

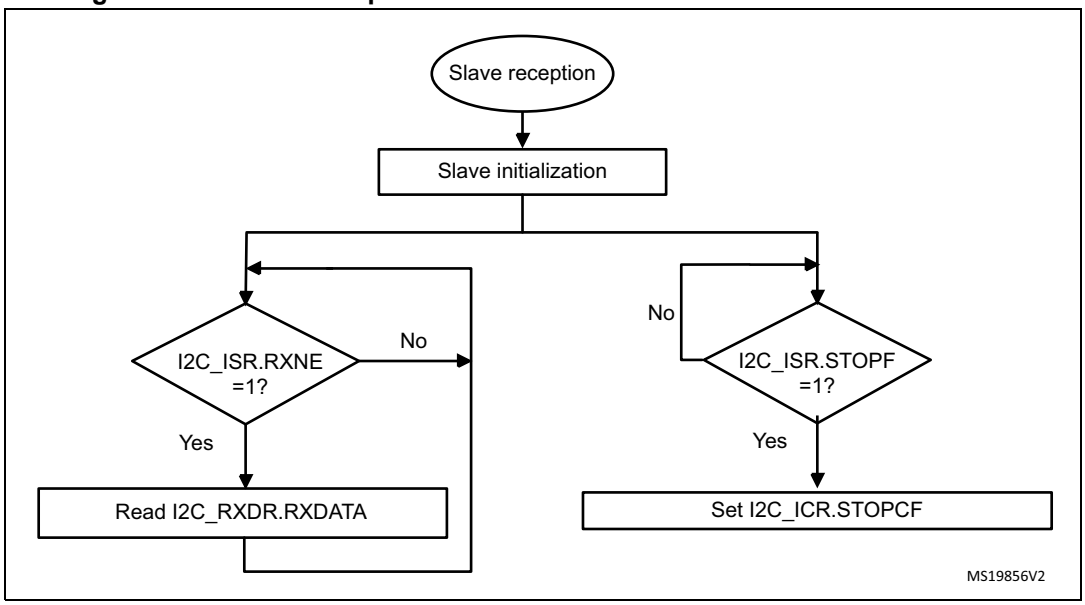
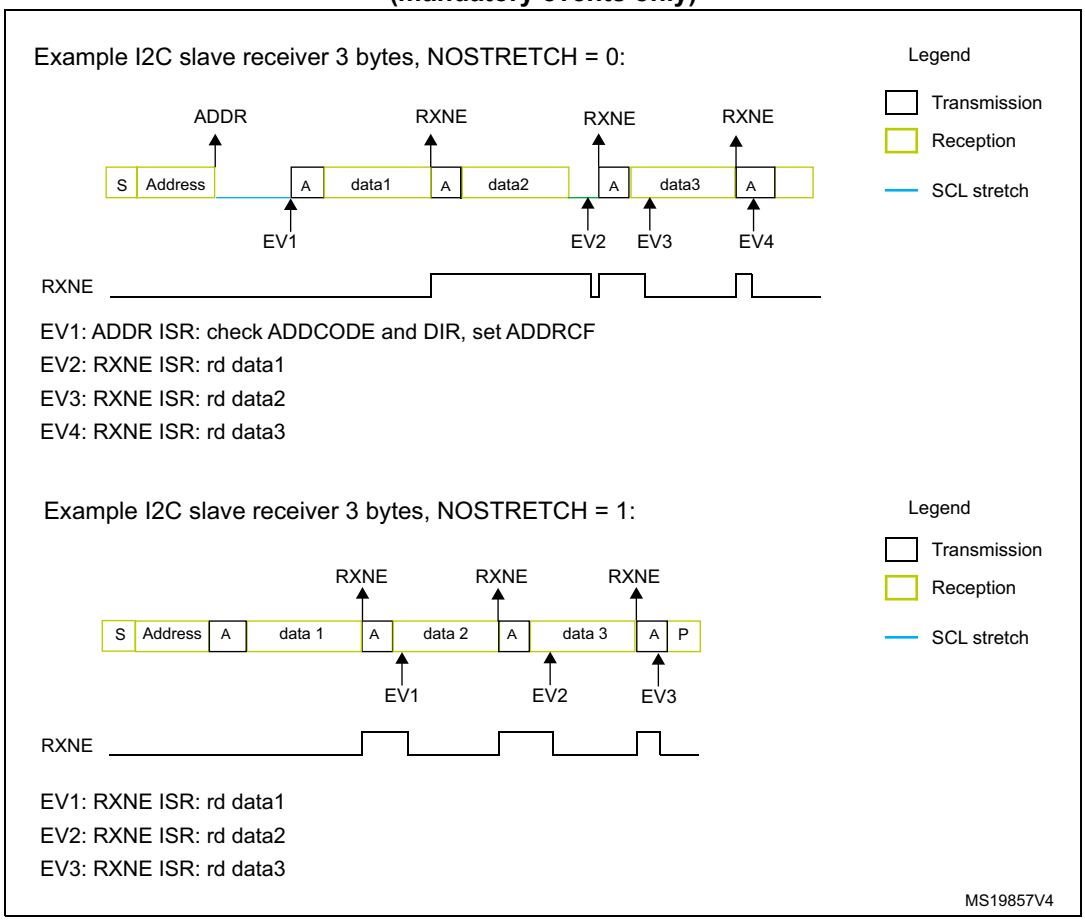



**Figure 273. Transfer bus diagrams for I2C slave receiver (mandatory events only)**



## 27.4.9 I2C master mode

### I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave 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 and 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 reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the master clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

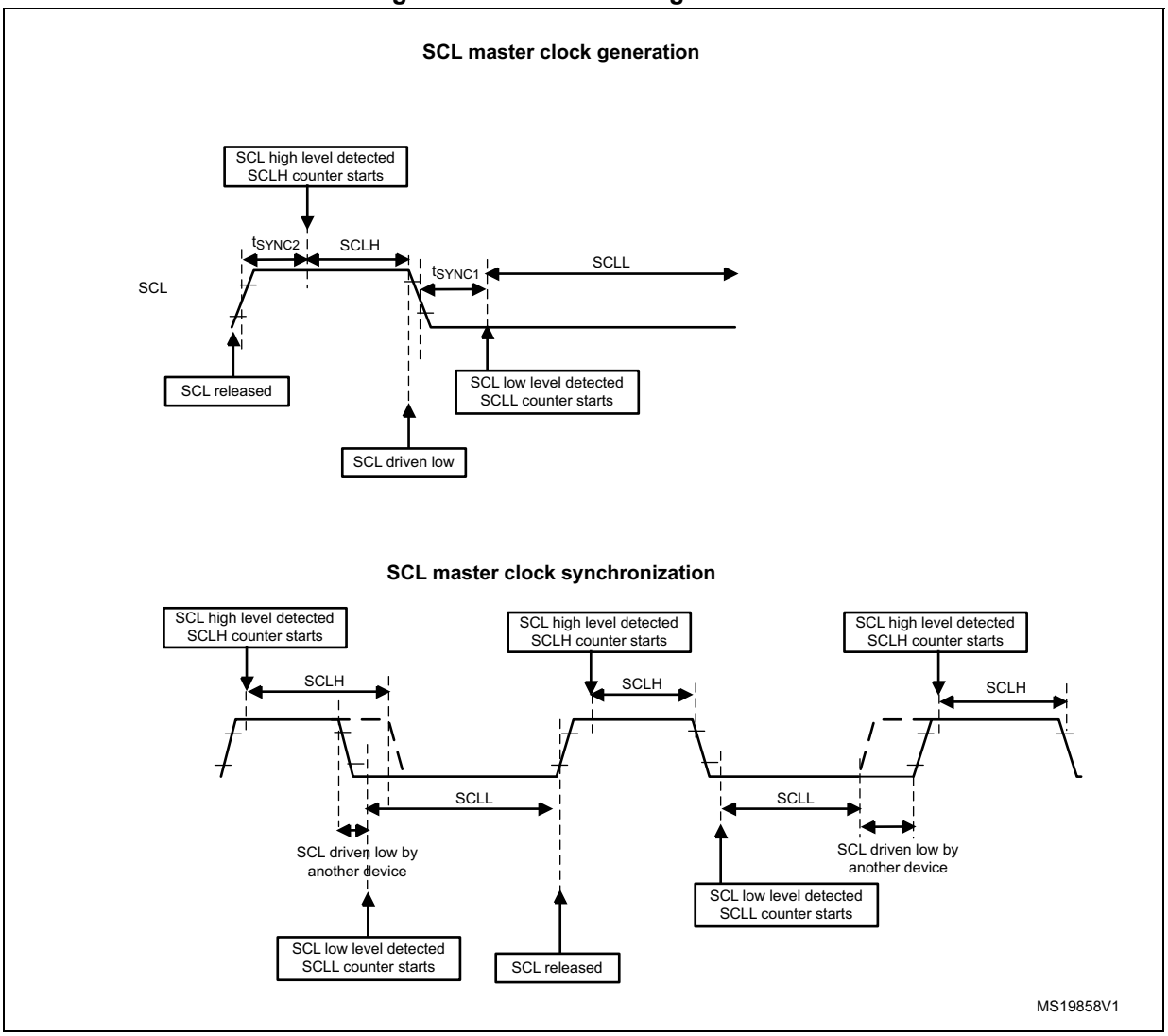
The duration of $t_{SYNC1}$ depends upon:

- 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 (two to three I2CCLK periods)

The duration of $t_{SYNC2}$ depends upon:

- 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 (two to three I2CCLK periods)

**Figure 274. Master clock generation**



**Caution:** To be I²C or SMBus compliant, the master clock must respect the timings given in the following table.

**Table 159. I²C-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 | - | |
| $t_{SU:STO}$ | Set-up time for STOP condition | 4.0 | - | 0.6 | - | 0.26 | - | 4.0 | - | |
| $t_{BUF}$ | Bus free time between a STOP and START condition | 4.7 | - | 1.3 | - | 0.5 | - | 4.7 | - | |
| $t_{LOW}$ | Low period of the SCL clock | 4.7 | - | 1.3 | - | 0.5 | - | 4.7 | - | |
| $t_{HIGH}$ | Period of the SCL clock | 4.0 | - | 0.6 | - | 0.26 | - | 4.0 | 50 | |
| $t_r$ | 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 | |

*Note: SCLL and SCLH are also used to generate, respectively, the $t_{BUF}$ / $t_{SU:STA}$ and the $t_{HD:STA}$ / $t_{SU:STO}$ timings.*

Refer to *Section 27.4.10* for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

### Master communication initialization (address phase)

To initiate the communication, program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave 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 this number is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

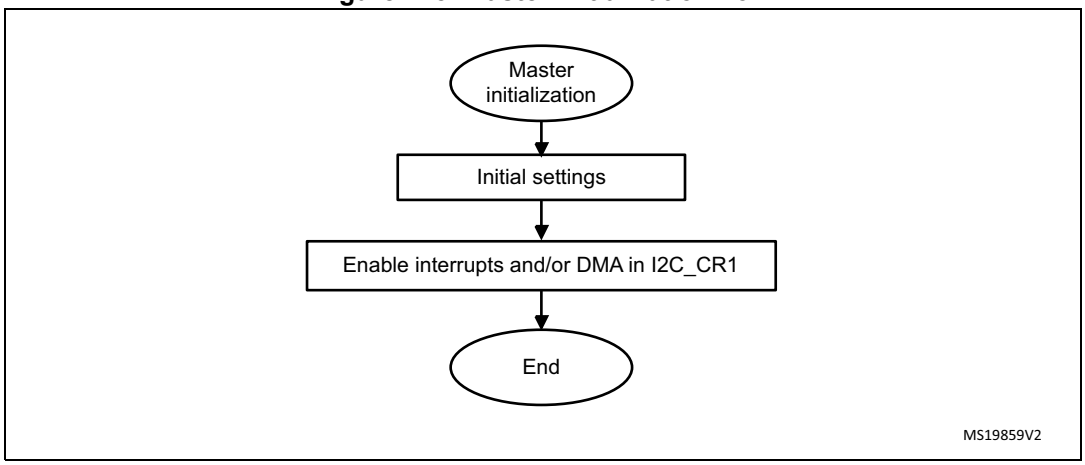
Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a $t_{BUF}$ delay.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

*Note: The START bit is reset by hardware when the slave address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.*
*In 10-bit addressing mode, when the slave address first seven bits are NACKed by the slave, the master relaunches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, to stop sending the slave address.*
*If the I2C is addressed as a slave (ADDR = 1) while the START bit is set, the I2C switches to slave 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.*

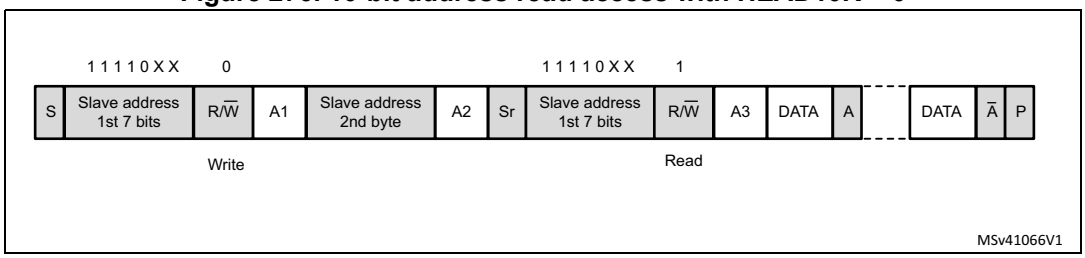
**Figure 275. Master initialization flow**




## Initialization of a master receiver addressing a 10-bit address slave

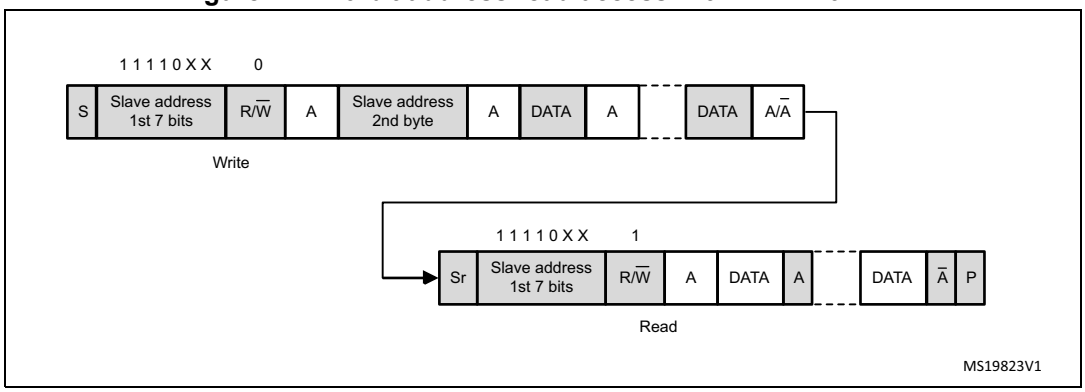
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address second byte + (Re)Start + Slave address 10-bit header Read

**Figure 276. 10-bit address read access with HEAD10R = 0**



- If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10-bit slave address configured with HEAD10R = 1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

**Figure 277. 10-bit address read access with HEAD10R = 1**




## Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth 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, to perform software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave 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 278. Transfer sequence flow for I2C master transmitter for N ≤ 255 bytes**

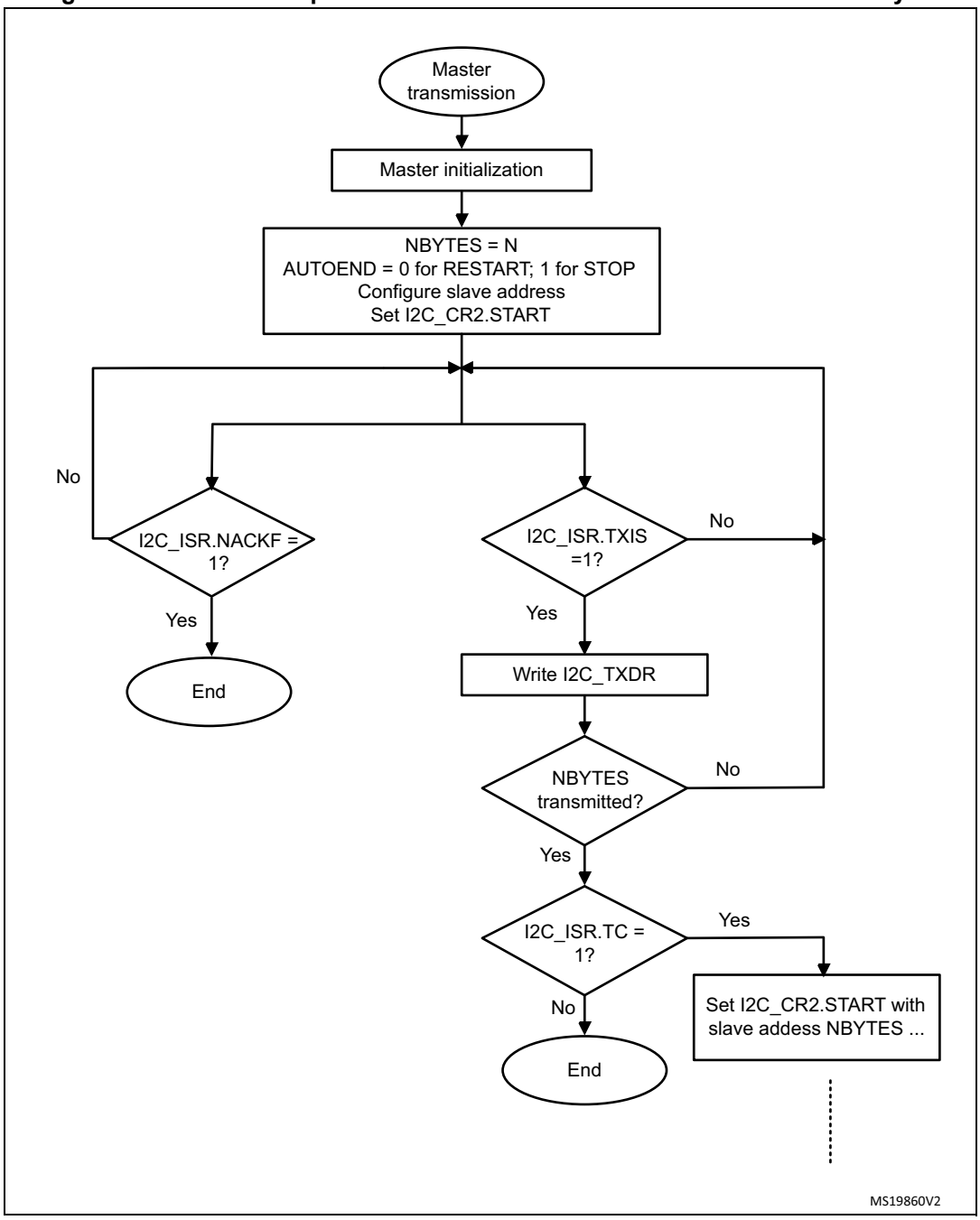


**Figure 279. Transfer sequence flow for I2C master transmitter for N > 255 bytes**

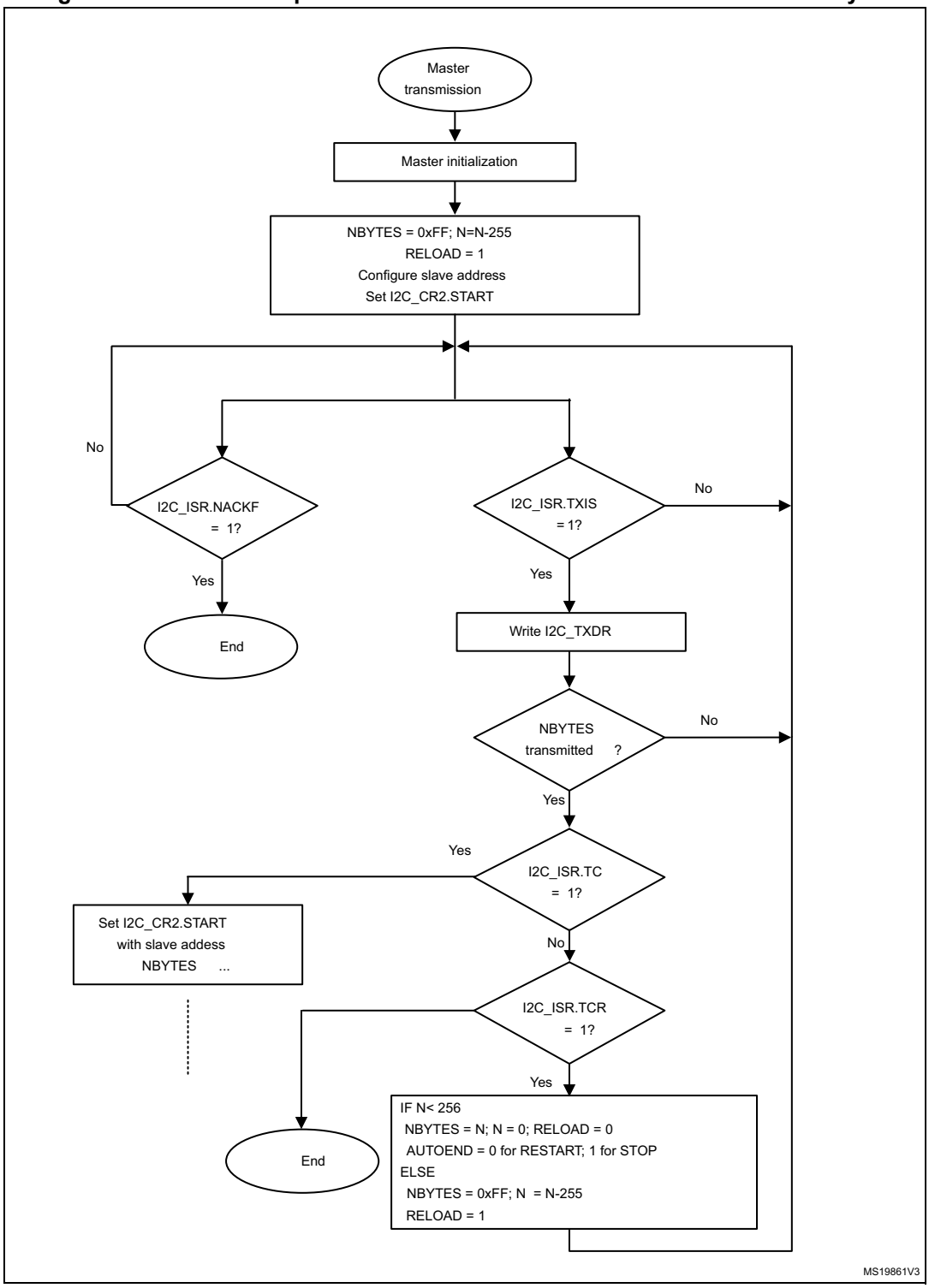
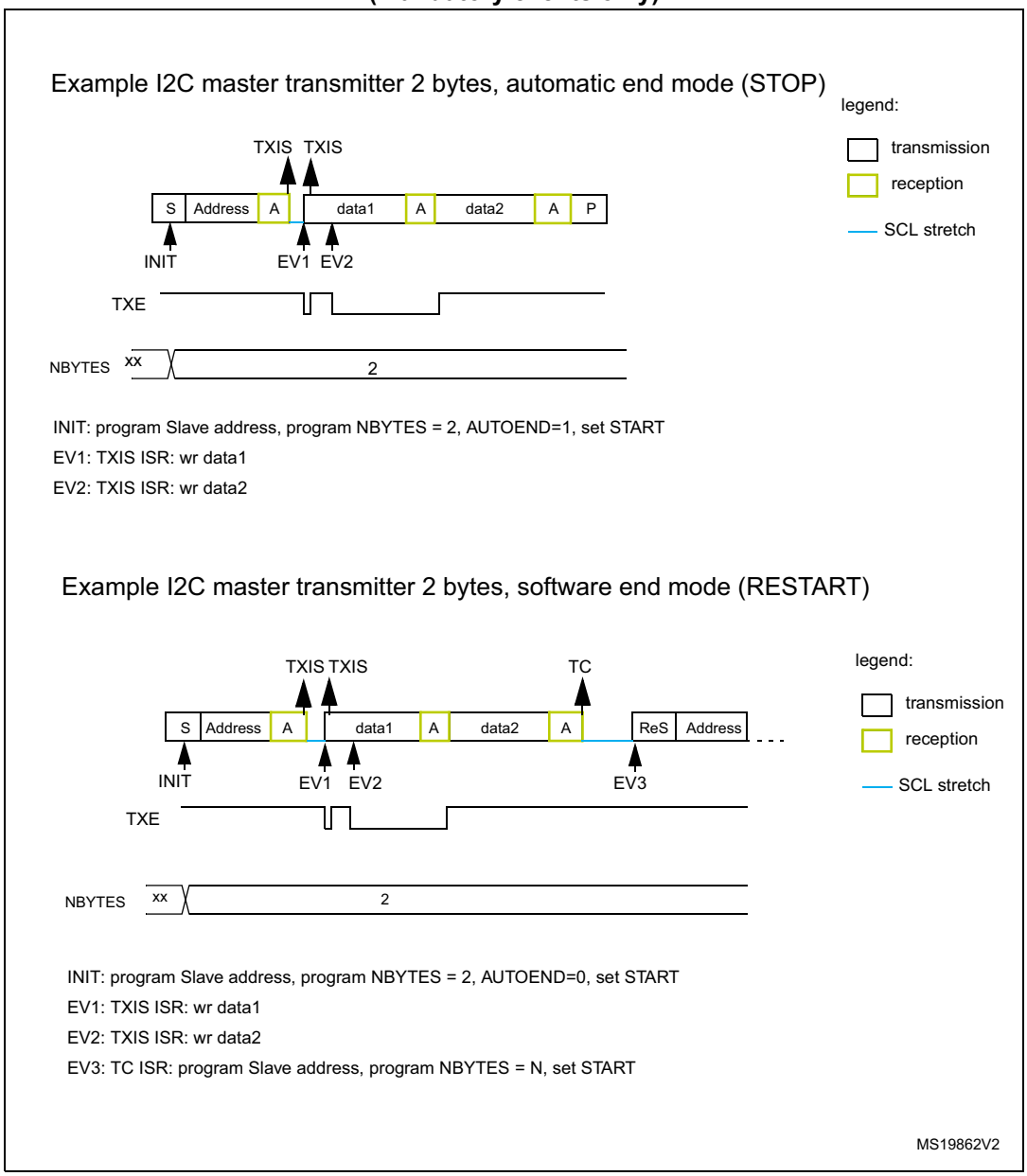


**Figure 280. Transfer bus diagrams for I2C master transmitter (mandatory events only)**



### Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth 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 slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave 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 281. Transfer sequence flow for I2C master receiver for N ≤ 255 bytes**

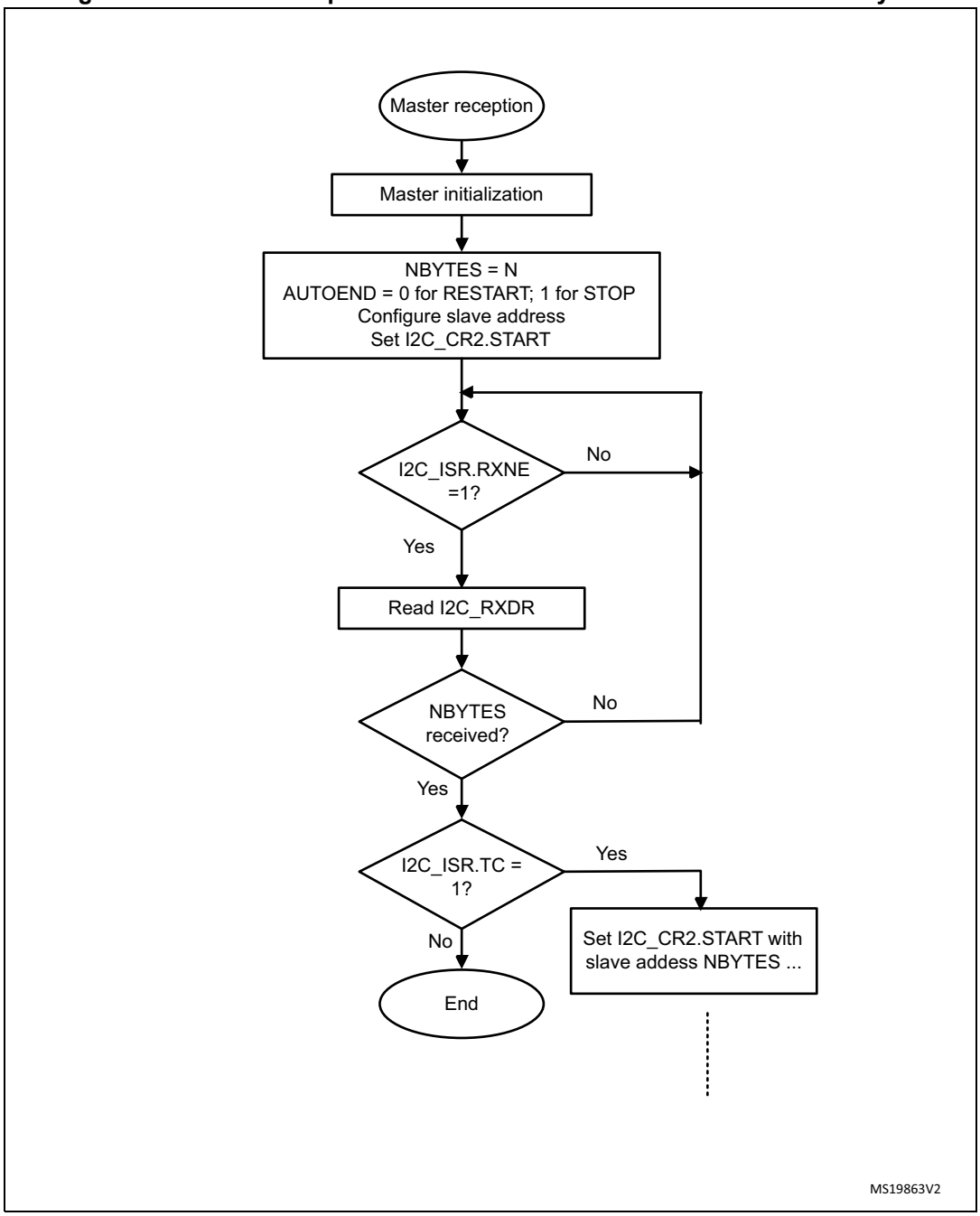


**Figure 282. Transfer sequence flow for I2C master receiver for N > 255 bytes**

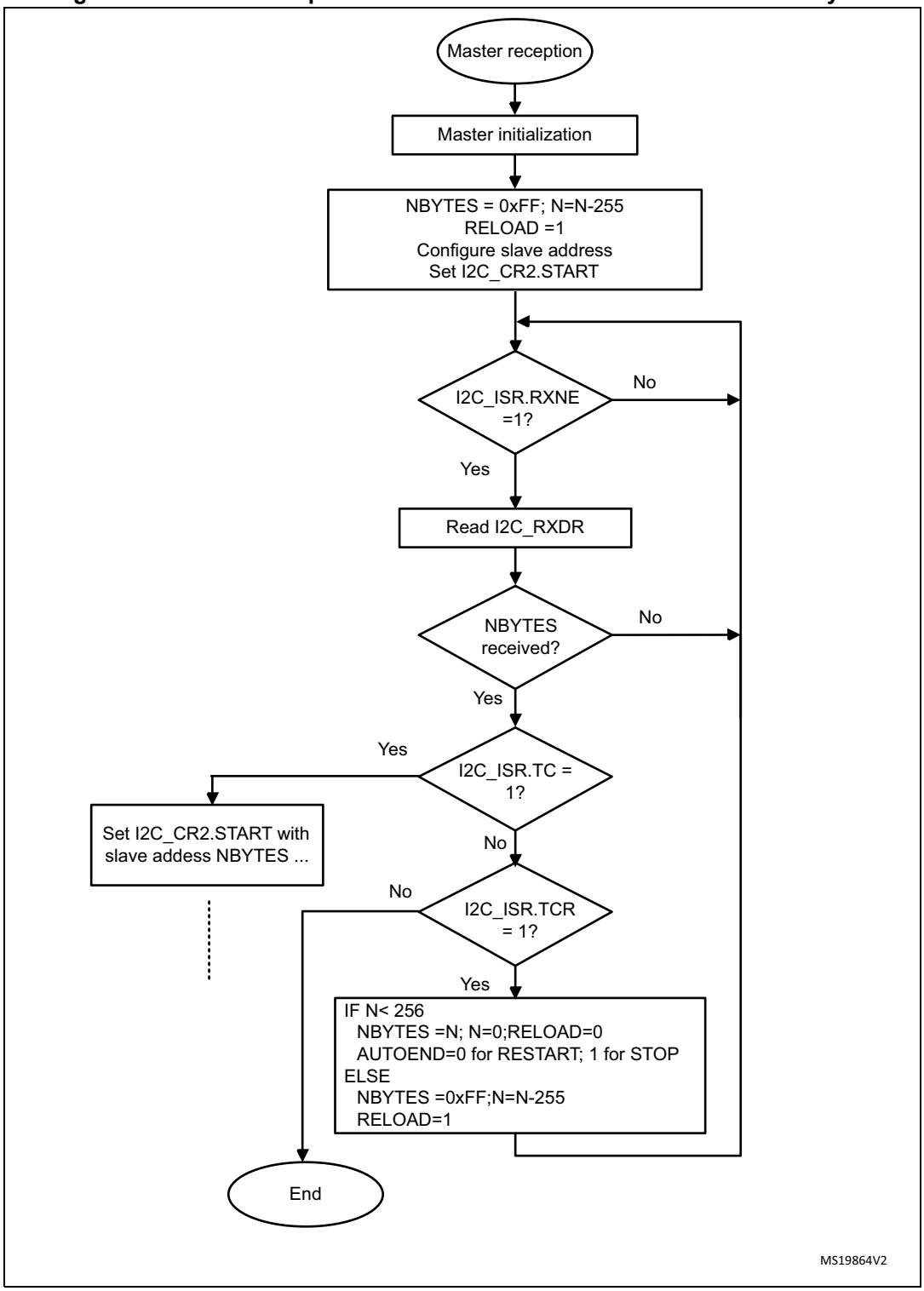
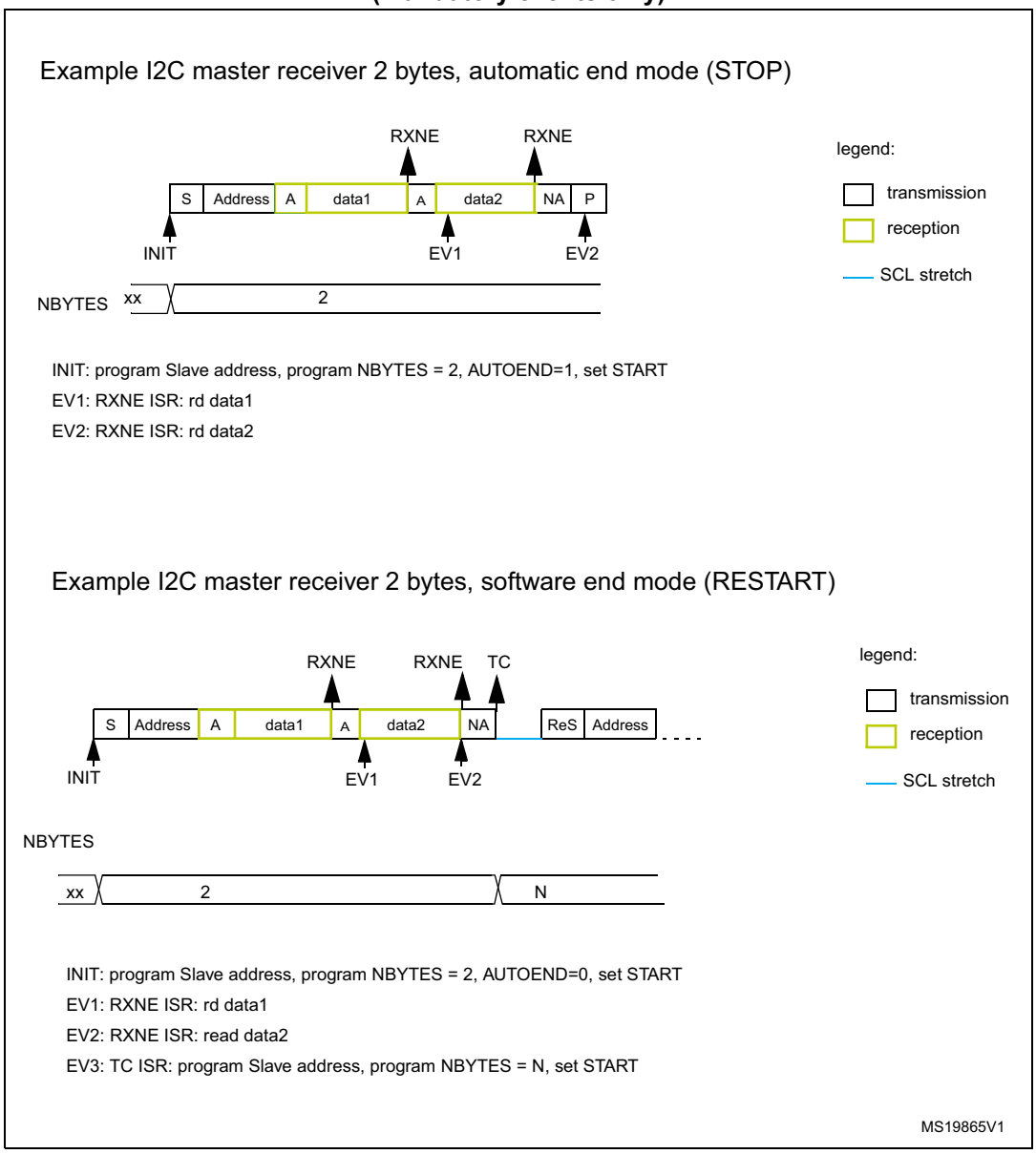


**Figure 283. Transfer bus diagrams for I2C master receiver (mandatory events only)**




## 27.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the $I^2C$ specification. To get more accurate configuration values, use the STM32CubeMX tool (I2C Configuration window).

**Table 160. Examples of timing settings for $f_{I2CCLK}$ = 8 MHz**

| Parameter | Standard-mode (Sm) | | Fast-mode (Fm) | Fast-mode Plus (Fm+) |
|---|---|---|---|---|
| | 10 kHz | 100 kHz | 400 kHz | 500 kHz |
| PRESC | 0x1 | 0x1 | 0x0 | 0x0 |
| SCLL | 0xC7 | 0x13 | 0x9 | 0x6 |
| $t_{SCLL}$ | 200 x 250 ns = 50 µs | 20 x 250 ns = 5.0 µs | 10 x 125 ns = 1250 ns | 7 x 125 ns = 875 ns |
| SCLH | 0xC3 | 0xF | 0x3 | 0x3 |
| $t_{SCLH}$ | 196 x 250 ns = 49 µs | 16 x 250 ns = 4.0 µs | 4 x 125 ns = 500 ns | 4 x 125 ns = 500 ns |
| $t_{SCL}$(1) | ~100 µs(2) | ~10 µs(2) | ~2.5 µs(3) | ~2.0 µs(4) |
| SDADEL | 0x2 | 0x2 | 0x1 | 0x0 |
| $t_{SDADEL}$ | 2 x 250 ns = 500 ns | 2 x 250 ns = 500 ns | 1 x 125 ns = 125 ns | 0 ns |
| SCLDEL | 0x4 | 0x4 | 0x3 | 0x1 |
| $t_{SCLDEL}$ | 5 x 250 ns = 1250 ns | 5 x 250 ns = 1250 ns | 4 x 125 ns = 500 ns | 2 x 125 ns = 250 ns |

1. $t_{SCL}$ 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 $t_{I2CCLK}$ = 500 ns. Example with $t_{SYNC1}$ + $t_{SYNC2}$ = 1000 ns.
3. $t_{SYNC1}$ + $t_{SYNC2}$ minimum value is 4 x $t_{I2CCLK}$ = 500 ns. Example with $t_{SYNC1}$ + $t_{SYNC2}$ = 750 ns.
4. $t_{SYNC1}$ + $t_{SYNC2}$ minimum value is 4 x $t_{I2CCLK}$ = 500 ns. Example with $t_{SYNC1}$ + $t_{SYNC2}$ = 655 ns.

**Table 161. Examples of timing 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 | 0x3 | 0x3 | 0x1 | 0x0 |
| 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) | ~2.5 µs(3) | ~1.0 µs(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. $t_{SCL}$ 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.

### 27.4.11 SMBus specific features

This section is relevant only when the SMBus feature is supported (refer to *Section 27.3*).

### 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. The SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks, and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

### Bus protocols

There are eleven possible command protocols for any given device. A device can use any or all of them 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 must be implemented by the user software.

For more details on these protocols, refer to SMBus specification (http://smbus.org).

### Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. 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 must be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus address resolution protocol, refer to SMBus specification (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 slave mode, the Slave byte control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to *Slave byte control mode* 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 master and the host as a slave.

### SMBus alert

The SMBus ALERT optional signal is supported. A slave-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 slave 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. The 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 a not acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

### Timeouts

This peripheral embeds hardware timers to be compliant with the three timeouts defined in the SMBus specification.

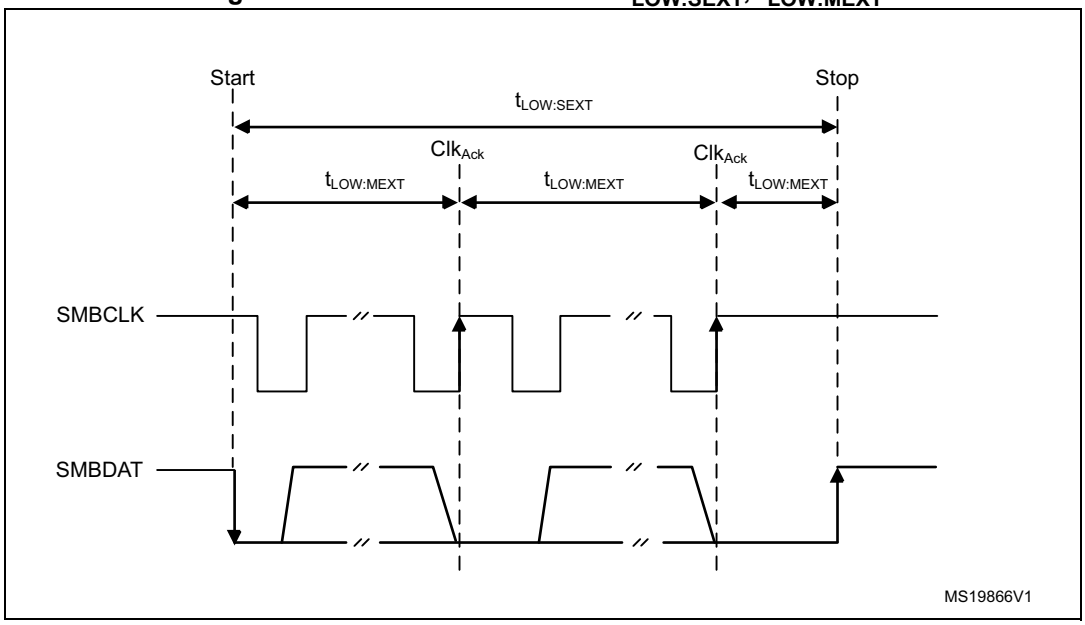
**Table 162. SMBus timeout specifications**

| Symbol | Parameter | Limits | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| $t_{TIMEOUT}$ | Detect clock low timeout | 25 | 35 | ms |
| $t_{LOW:SEXT}$(1) | Cumulative clock low extend time (slave device) | - | 25 | |
| $t_{LOW:MEXT}$(2) | Cumulative clock low extend time (master device) | - | 10 | |

1. $t_{LOW:SEXT}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another slave device or the master also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.

2. $t_{LOW:MEXT}$ is the cumulative time a master 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 slave device or another master also extends 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 slave device as the sole target of the master.

Figure 284. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$




### Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH,MAX}$ (refer to *I2C timings*).

This timing parameter covers the condition where a master 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 master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

## 27.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported (see *Section 27.3*).

In addition to I2C initialization, some other specific initialization must be done to perform SMBus communication.

### Received command and data acknowledge control (slave mode)

A SMBus receiver must be able to NACK each received command or data. To allow ACK control in slave mode, the Slave byte control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to *Slave byte control mode* for more details.

### Specific address (slave mode)

The specific SMBus addresses must be enabled if needed. Refer to *Bus idle detection* 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 the 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 slave 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 163. SMBus with PEC configuration**

| Mode | SBC bit | RELOAD bit | AUTOEND bit | PECBYTE bit |
|---|---|---|---|---|
| Master Tx/Rx NBYTES + PEC+ STOP | x | 0 | 1 | 1 |
| Master Tx/Rx NBYTES + PEC + ReSTART | x | 0 | 0 | 1 |
| Slave 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.

- $t_{TIMEOUT}$ check

  To enable the $t_{TIMEOUT}$ check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value, to check the $t_{TIMEOUT}$ parameter. The TIDLE bit must be configured to 0 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 164*.

**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 master or as a slave, the 12-bit TIMEOUTB timer must be configured to check $t_{LOW:SEXT}$ for a slave, and $t_{LOW:MEXT}$ for a master. As the standard specifies only a maximum, the user can choose the same value for both. The timer is then 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* section, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 165*

**Caution:** Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

### Bus idle detection

To enable the $t_{IDLE}$ check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value, to obtain the $t_{IDLE}$ parameter. The TIDLE bit must be configured to '1 to detect both SCL and SDA high level timeout. The timer is then 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 166*.

**Caution:** Changing TIMEOUTA and TIDLE configuration is not allowed when TIMEOUTEN is set.

## 27.4.13 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 27.3*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

**Table 164. Examples of TIMEOUTA settings (max $t_{TIMEOUT}$ = 25 ms)**

| $f_{I2CCLK}$ | TIMEOUTA[11:0] bits | TIDLE bit | TIMEOUTEN bit | $t_{TIMEOUT}$ |
|---|---|---|---|---|
| 8 MHz | 0x61 | 0 | 1 | 98 x 2048 x 125 ns = 25 ms |
| 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 165. Examples of TIMEOUTB settings**

| $f_{I2CCLK}$ | TIMEOUTB[11:0] bits | TEXTEN bit | $t_{LOW:EXT}$ |
|---|---|---|---|
| 8 MHz | 0x1F | 1 | 32 x 2048 x 125 ns = 8 ms |
| 16 MHz | 0x3F | 1 | 64 x 2048 x 62.5 ns = 8 ms |

- Configuring the maximum duration of $t_{IDLE}$ to 50 µs

**Table 166. Examples of TIMEOUTA settings (max $t_{IDLE}$ = 50 µs)**

| $f_{I2CCLK}$ | TIMEOUTA[11:0] bits | TIDLE bit | TIMEOUTEN bit | $t_{TIDLE}$ |
|---|---|---|---|---|
| 8 MHz | 0x63 | 1 | 1 | 100 x 4 x 125 ns = 50 µs |
| 16 MHz | 0xC7 | 1 | 1 | 200 x 4 x 62.5 ns = 50 µs |

## 27.4.14 SMBus slave mode

This section is relevant only when the SMBus feature is supported (refer to *Section 27.3*).

In addition to I2C slave transfer management (refer to *Section 27.4.8*), additional software flows are provided to support the SMBus.

### SMBus slave transmitter

When the IP is used in SMBus, SBC must be programmed to 1 to enable 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 is NBYTES - 1, and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES - 1 data transfer.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 285. Transfer sequence flow for SMBus slave transmitter N bytes + PEC**

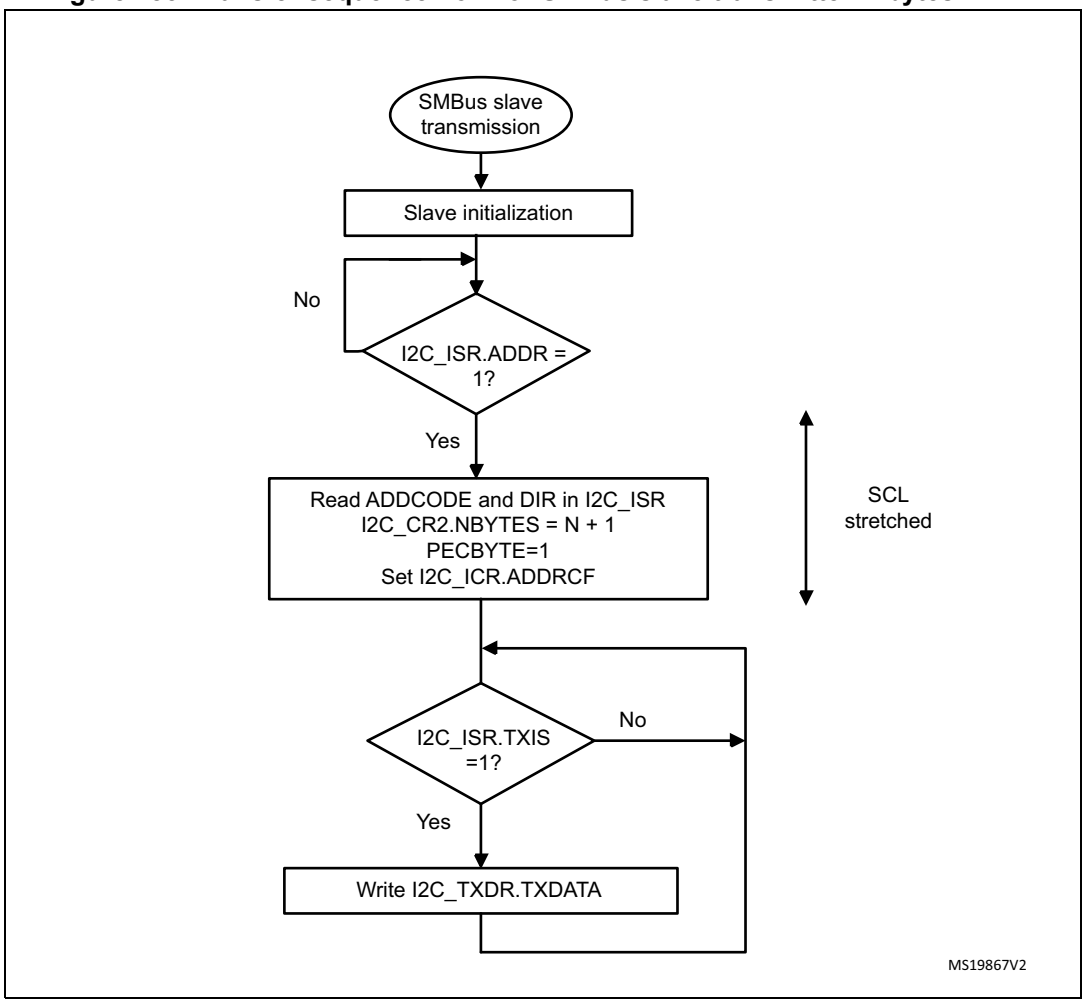
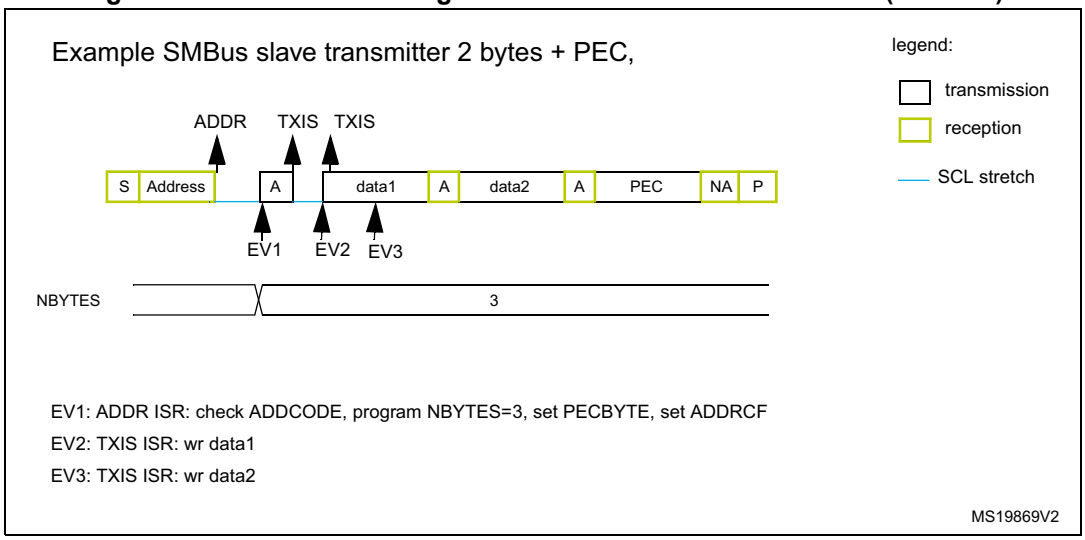


**Figure 286. Transfer bus diagrams for SMBus slave transmitter (SBC = 1)**




### SMBus slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to 1 to allow the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Slave byte control mode* for more details.

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, the user 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 287. Transfer sequence flow for SMBus slave receiver N bytes + PEC**

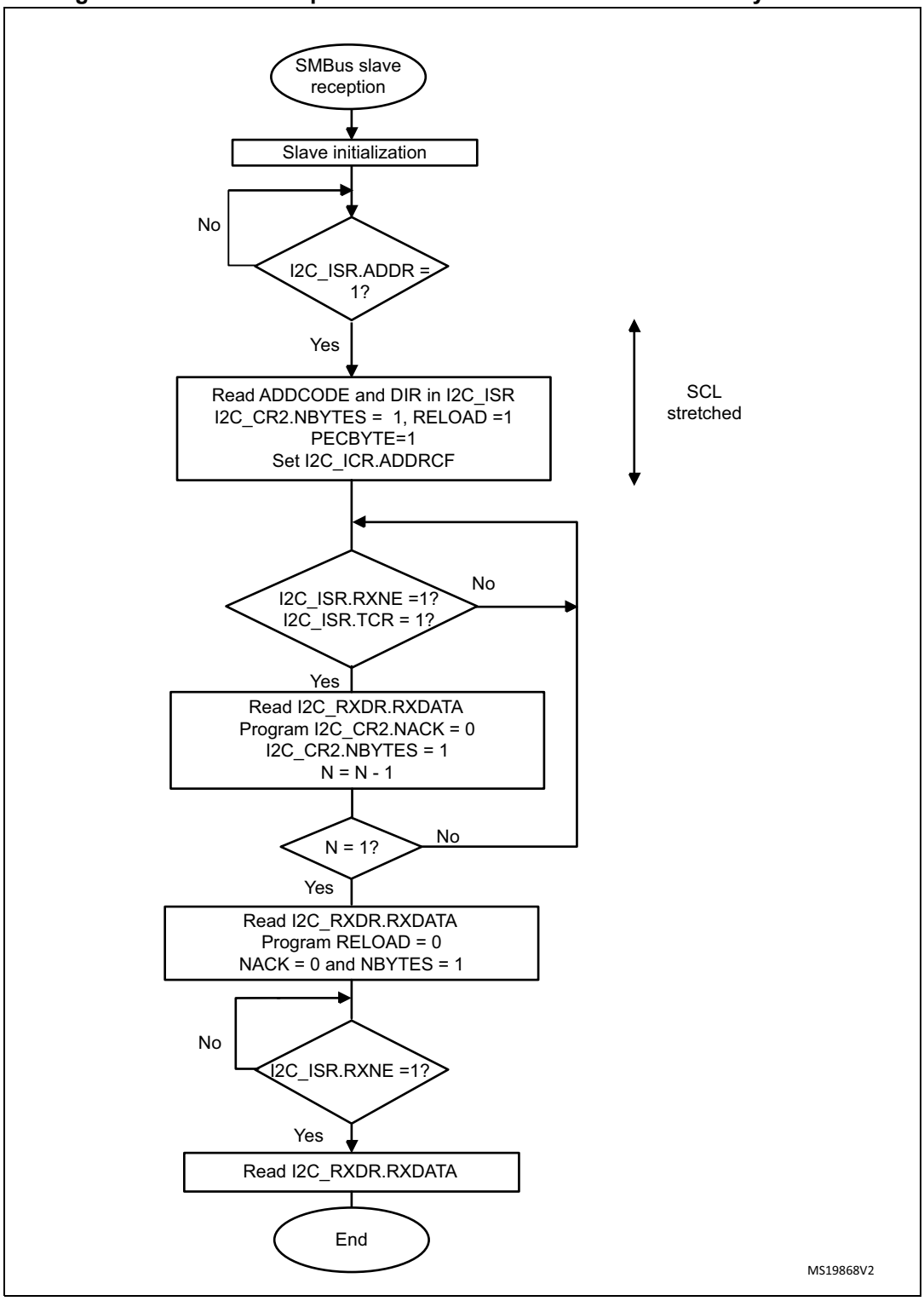
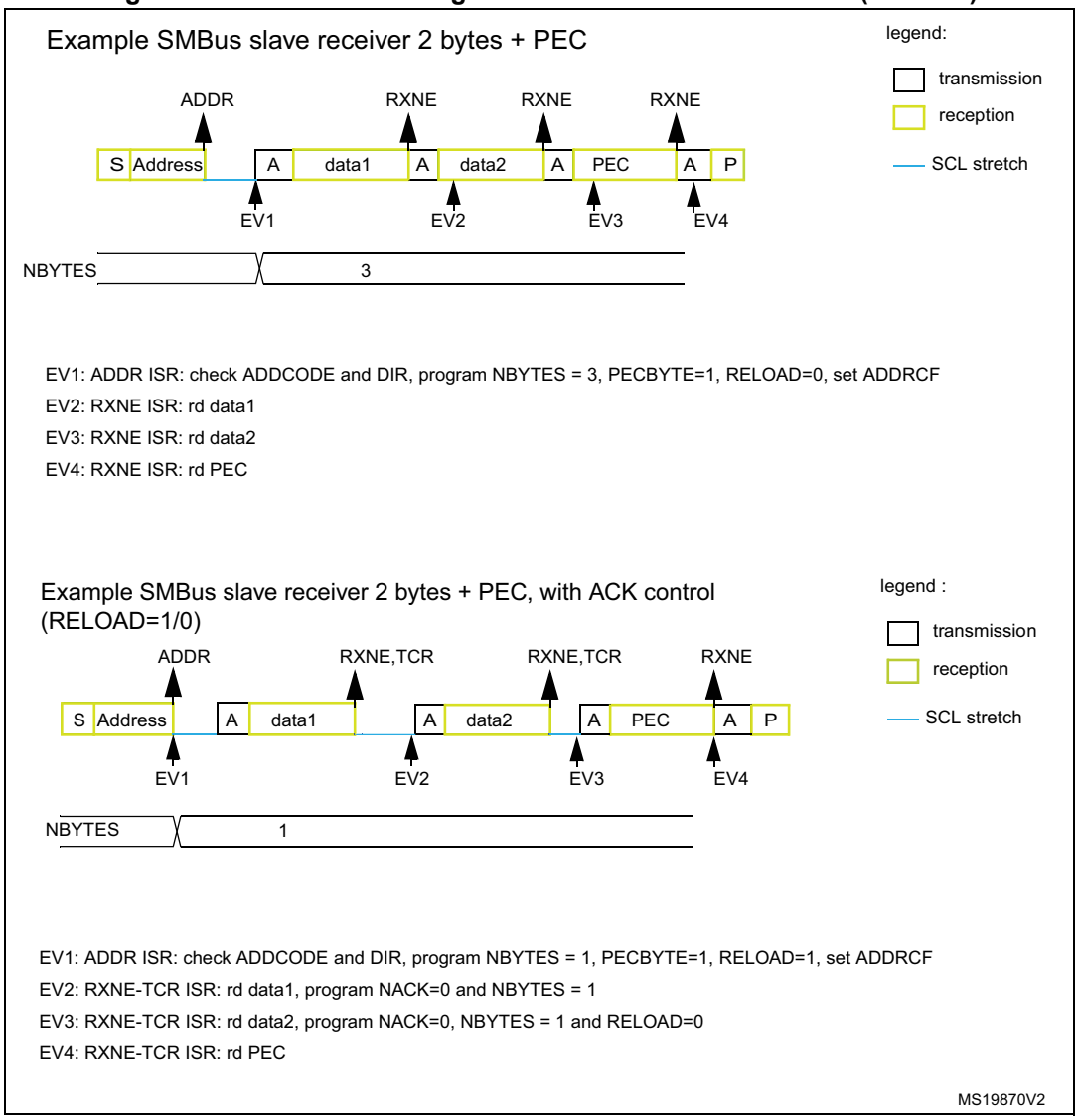


**Figure 288. Bus transfer diagrams for SMBus slave receiver (SBC = 1)**




This section is relevant only when the SMBus feature is supported (refer to *Section 27.3*).

In addition to I2C master transfer management (refer to *Section 27.4.9*), additional software flows are provided to support the SMBus.

### SMBus master transmitter

When the SMBus master 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.

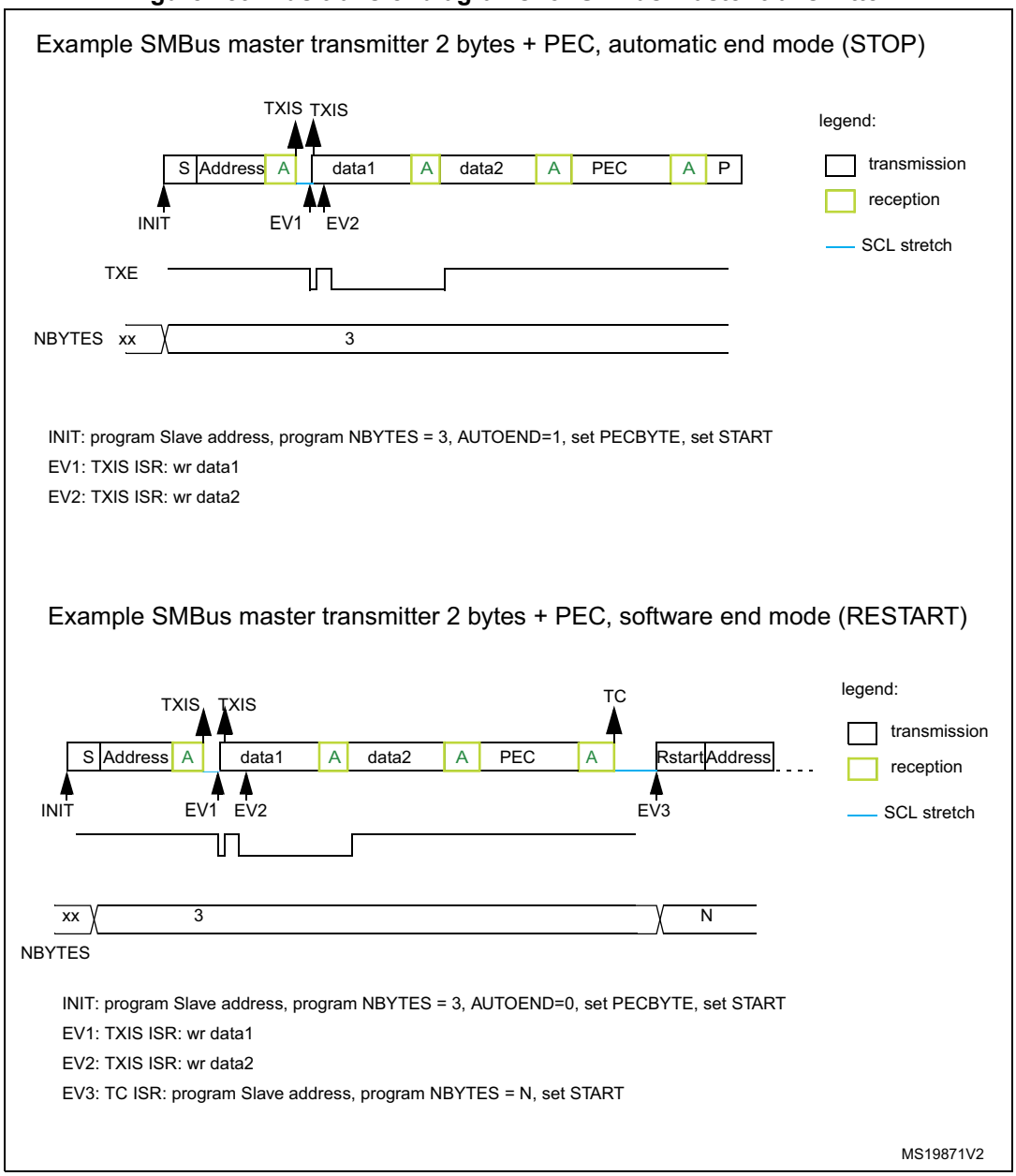
If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master 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 289. Bus transfer diagrams for SMBus master transmitter**



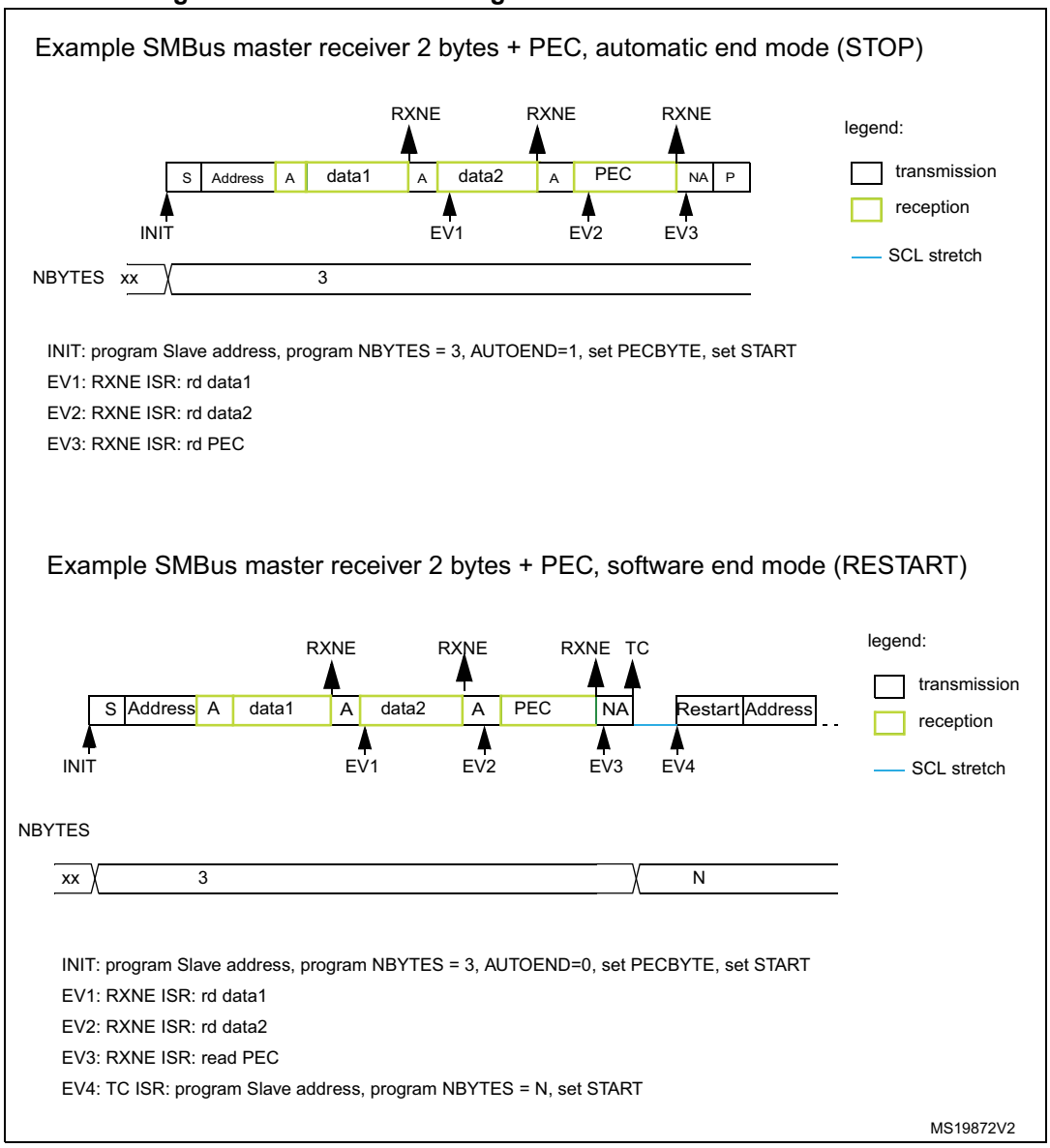
### SMBus master receiver

When the SMBus master 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 slave 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 master 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 slave 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 290. Bus transfer diagrams for SMBus master receiver**




## 27.4.15 Wake-up from Stop mode on address match

This section is relevant only when wake-up from Stop mode feature is supported (refer to *Section 27.3*).

The I2C is able to wake-up the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wake-up from Stop mode is enabled by setting the WUPEN bit in the I2C_CR1 register. The HSI oscillator must be selected as the clock source for I2CCLK in order to allow wake-up from Stop mode.

During Stop mode, the HSI is switched off. When a START is detected, the I2C interface switches the HSI on, and stretches SCL low until HSI is woken up.

HSI is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wake-up time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI is switched off again and the MCU is not woken up.

*Note:* *If the I2C clock is the system clock, or if WUPEN = 0, the* HSI *is not switched on after a START is received.*

*Only an ADDR interrupt can wake up the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.*

**Caution:** The digital filter is not compatible with the wake-up from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

**Caution:** This feature is available only when the I2C clock source is the HSI oscillator.

**Caution:** Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

**Caution:** If wake up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

## 27.4.16 Error conditions

The following errors are the conditions that can cause a communication 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 nine SCL clock pulses. A START or a STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave 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 master 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 master switches automatically to slave mode.
- In slave 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 slave 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 must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must 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 27.3)*.

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 27.3)*.

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 an 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.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{LOW:MEXT}$ parameter).
- Slave 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 master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave 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 27.3*).

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.

### 27.4.17 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 ) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave 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 *Master transmitter*.
- In slave 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 slave transmitter* and *SMBus master transmitter*.

*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 ) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave 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 slave 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 27.3*) the PEC transfer is managed with the NBYTES counter. Refer to *SMBus slave receiver* and *SMBus master receiver*.

*Note: If DMA is used for reception, the RXIE bit does not need to be enabled.*

### 27.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_ configuration bits in the DBG module.

## 27.5 I2C low-power modes

**Table 167. Effect of low-power modes on the I2C**

| Mode | Description |
|---|---|
| Sleep | No effect. I2C interrupts cause the device to exit the Sleep mode. |
| Stop[1] | The I2C registers content is kept.<br>– WUPEN = 1 and I2C is clocked by an internal oscillator (HSI): the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode.<br>– WUPEN = 0: the I2C must be disabled before entering Stop mode. |
| Standby | The I2C peripheral is powered down and must be reinitialized after exiting Standby mode. |

1. Refer to *Section 27.3* for information about the Stop modes supported by each instance. If wake-up from a specific Stop mode is not supported, the instance must be disabled before entering this Stop mode.

## 27.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

**Table 168. I2C interrupt requests**

| Interrupt acronym | Interrupt event | Event flag | Enable control bit | Interrupt clear method | Exit Sleep mode | Exit Stop modes | Exit Standby modes |
|---|---|---|---|---|---|---|---|
| I2C_EV | Receive buffer not empty | RXNE | RXIE | Read I2C_RXDR register | Yes | No | No |
| | Transmit buffer interrupt status | TXIS | TXIE | Write I2C_TXDR register | | | |
| | Stop detection interrupt flag | STOPF | STOPIE | Write STOPCF = 1 | | | |
| | Transfer complete reload | TCR | TCIE | Write I2C_CR2 with NBYTES[7:0] ≠ 0 | | | |
| | Transfer complete | TC | | Write START = 1 or STOP = 1 | | | |
| | Address matched | ADDR | ADDRIE | Write ADDRCF = 1 | | Yes(1) | |
| | NACK reception | NACKF | NACKIE | Write NACKCF = 1 | | No | |
| I2C_ER | Bus error | BERR | ERRIE | Write BERRCF = 1 | Yes | No | No |
| | Arbitration loss | ARLO | | Write ARLOCF = 1 | | | |
| | Overrun/underrun | OVR | | Write OVRCF = 1 | | | |
| I2C_ER | PEC error | PECERR | ERRIE | Write PECERRCF = 1 | Yes | No | No |
| | Timeout/ $t_{LOW}$ error | TIMEOUT | | Write TIMEOUTCF = 1 | | | |
| | SMBus alert | ALERT | | Write ALERTCF = 1 | | | |

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the wake-up from Stop mode feature. Refer to *Section 27.3*.

## 27.7 I2C registers

Refer to *Section 1.2 on page 51* for the list of abbreviations used in register descriptions.

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

### 27.7.1 I2C 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 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. | PEC EN | ALERT EN | SMBD EN | SMBH EN | GC EN | WUP EN | NO STRETCH | SBC |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RXDMA EN | TXDMA EN | Res. | ANF OFF | DNF[3:0] | | | | ERRIE | TCIE | STOP IE | NACK IE | ADDR IE | RXIE | TXIE | PE |
| 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.

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 27.3.*

Bit 22 **ALERTEN**: SMBus alert enable

0: The SMBus alert pin (SMBA) is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the Alert Response Address header is disabled (0001100x followed by NACK).

1: The SMBus alert pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the Alert Response Address header is enabled (0001100x followed by ACK).

*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 27.3.*

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 27.3.*

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 27.3.*

Bit 19 **GCEN**: General call enable

0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable

0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.

*Note: If the wake-up from Stop mode feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 27.3.*

*Note: WUPEN can be set only when DNF = 0000.*

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.
0: Clock stretching enabled
1: Clock stretching disabled

*Note: This bit can be programmed only when the I2C is disabled (PE = 0).*

Bit 16 **SBC**: Slave byte control

This bit is used to enable hardware byte control in slave mode.
0: Slave byte control disabled
1: Slave 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 be programmed only 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 one $t_{I2CCLK}$
...
1111: digital filter enabled and filtering capability up to fifteen $t_{I2CCLK}$

*Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only 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 generates 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 generates 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 (slave 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 disabled
1: Peripheral enabled
*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 three APB clock cycles.*

### 27.7.2 I2C 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 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 | rw | rw | rw | rw | 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_ WRN | SADD[9:0] | | | | | | | | | |
| rs | rs | rs | rw | rw | rw | rw | rw | rw | rw | rw | rw | 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, and in slave mode when SBC = 0.*

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 27.3.*

Bit 25 **AUTOEND**: Automatic end mode (master 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 slave 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 slave mode with SBC = 0.

*Note: Changing these bits when the START bit is set is not allowed.*

Bit 15 **NACK**: NACK generation (slave 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 only in slave mode: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.*

*When an overrun occurs in slave 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 (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

**In master 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 master 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 slave mode.*

*This bit has no effect when RELOAD is set.*

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10-bit slave address read sequence: Start + 2 bytes 10-bit address in write direction + restart + first seven bits of the 10-bit address in read direction.

1: The master sends only the first seven 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 (master mode)

0: The master operates in 7-bit addressing mode

1: The master 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 (master mode)

0: Master requests a write transfer

1: Master requests a read transfer

*Note: Changing this bit when the START bit is set is not allowed.*

Bits 9:0 **SADD[9:0]**: Slave address (master mode)

**In 7-bit addressing mode (ADD10 = 0)**:

SADD[7:1] must be written with the 7-bit slave address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

**In 10-bit addressing mode (ADD10 = 1)**:

SADD[9:0] must be written with the 10-bit slave address to be sent.

*Note: Changing these bits when the START bit is set is not allowed.*

### 27.7.3 I2C 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 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[9:0] | | | | | | | | | |
| rw | | | | | rw | rw | rw | rw | rw | rw | 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 slave address OA1 is NACKed.
1: Own address 1 enabled. The received slave 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 9:0 **OA1[9:0]**: Interface own slave address

7-bit addressing mode: OA1[7:1] contains the 7-bit own slave address. Bits OA1[9], OA1[8] and OA1[0] are don't care.
10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.

*Note: These bits can be written only when OA1EN = 0.*

### 27.7.4 I2C 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 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 2x 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 | rw | rw | rw | rw | rw | 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 slave address OA2 is NACKed.
1: Own address 2 enabled. The received slave 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 ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.*

Bits 7:1 **OA2[7:1]**: Interface address

7-bit addressing mode: 7-bit address

*Note: These bits can be written only when OA2EN = 0.*

Bit 0 Reserved, must be kept at reset value.

## 27.7.5 I2C 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 | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCLH[7:0] | | | | | | | | SCLL[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK to generate the clock period $t_{PRESC}$ used for data setup and hold counters (refer to *I2C timings*), and for SCL high and low level counters (refer to *I2C master initialization*).

$t_{PRESC}$ = (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 $t_{SCLDEL}$ = (SCLDEL + 1) x $t_{PRESC}$ between SDA edge and SCL rising edge. In master and in slave modes with NOSTRETCH = 0, the SCL line is stretched low during $t_{SCLDEL}$.

*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 and SDA edge. In master and in slave modes with NOSTRETCH = 0, the SCL line is stretched low during $t_{SDADEL}$.

$t_{SDADEL}$= SDADEL x $t_{PRESC}$

*Note: SDADEL is used to generate $t_{HD:DAT}$ timing.*

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master 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 (master mode)

This field is used to generate the SCL low period in master 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).*

*Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.*

### 27.7.6 I2C 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 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.

If the SMBus feature is not supported, this register is reserved, and its bits are forced by hardware to 0. Refer to *Section 27.3*.

| 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 | 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 |
| TIMOUTEN | Res. | Res. | TIDLE | TIMEOUTA[11:0] | | | | | | | | | | | |
| rw | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | 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:

– Master mode: the master cumulative clock low extend time ($t_{LOW:MEXT}$) is detected

– Slave mode: the slave cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) 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. hen 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.*

### 27.7.7 I2C 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 | r | r | r | r | 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 (slave 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 two MSBs of the address.

Bit 16 **DIR**: Transfer direction (slave mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave 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, and 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 an 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.*

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 27.3.*

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.*

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 27.3.*

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.*

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 27.3.*

Bit 10 **OVR**: Overrun/underrun (slave mode)

This flag is set by hardware in slave 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 slave mode. It is cleared by software by setting the 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 master mode, or for slave mode when the SBC bit is set.*

Bit 6 **TC**: Transfer complete (master 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:

- as a master, provided that the STOP condition is generated by the peripheral.
- as a slave, 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 (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave 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 only when NOSTRETCH = 1, 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.*

### 27.7.8 I2C 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 | TIMOUT CF | PEC CF | OVR CF | 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

Note: 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 27.3.*

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 27.3.*

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 27.3.*

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 NACKF 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.

### 27.7.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

If the SMBus feature is not supported, this register is reserved, and its bits are forced by hardware to 0. Refer to *Section 27.3*.

| 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 | r | r | r | r | r | r | 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.

### 27.7.10 I2C 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 | r | r | r | r | r | r | 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 $I^2C$ bus

### 27.7.11 I2C 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 | rw | rw | rw | rw | rw | rw | 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 $I^2C$ bus

*Note: These bits can be written only when TXE = 1.*

## 27.7.12 I2C register map

The table below provides the I2C register map and the reset values.

**Table 169. I2C register map and reset values**

| Offset | Register name | 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 | I2C_CR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PECEN | ALERTEN | SMBDEN | SMBHEN | GCEN | WUPEN | 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 | 0 |
| 0x04 | 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 |
| 0x08 | 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 |
| 0x0C | 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 |
| 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 on page 56* for the register boundary addresses.

# 28 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

## 28.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

## 28.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 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: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

## 28.3 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

## 28.4 USART implementation

The table(s) below describe(s) USART implementation. It(they) also include(s) LPUART for comparison.

**Table 170. USART features**

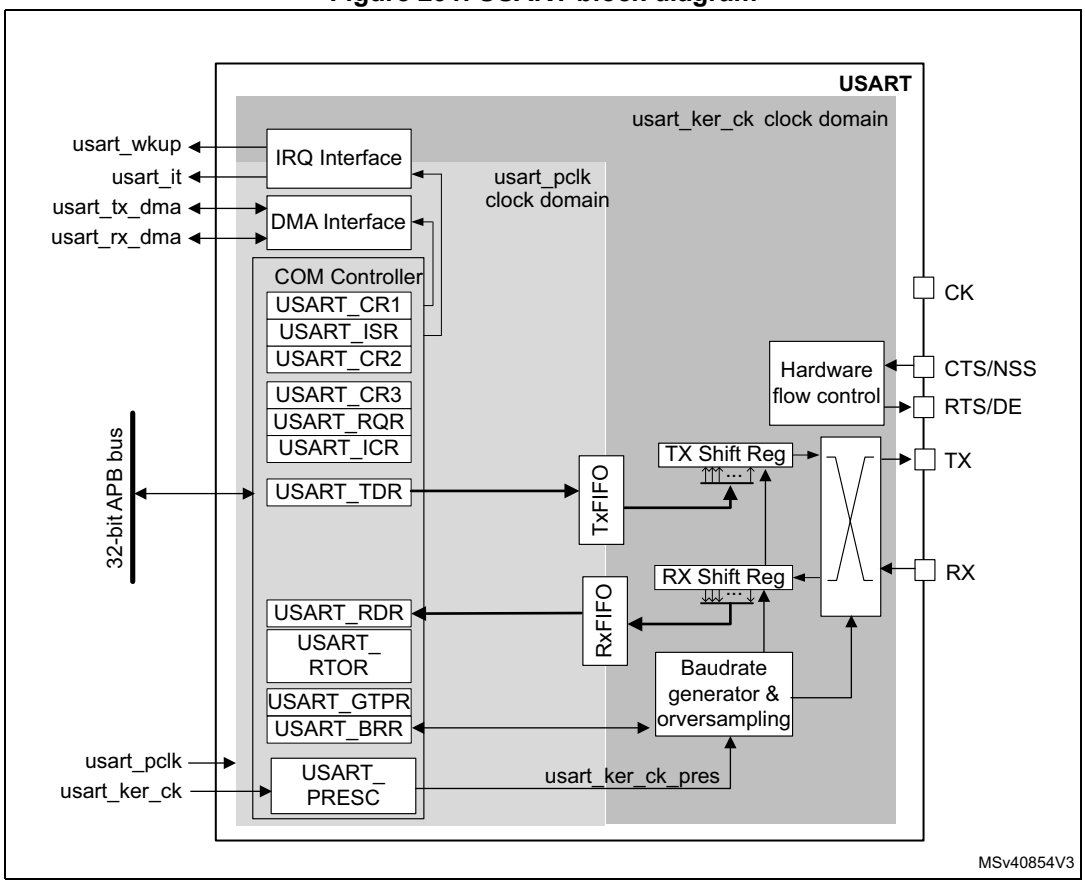
| USART modes/features[1] | USART1 |
|---|---|
| Hardware flow control for modem | X |
| Continuous communication using DMA | X |
| Multiprocessor communication | X |
| Synchronous mode (Master/Slave) | X |
| Smartcard mode | X |
| Single-wire half-duplex communication | X |
| IrDA SIR ENDEC block | X |
| LIN mode | X |
| Dual clock domain and wake-up from low-power mode | X |
| Receiver timeout interrupt | X |
| Modbus communication | X |
| Auto baud rate detection | X |
| Driver Enable | X |
| USART data length | 7, 8 and 9 bits |
| Tx/Rx FIFO | X |
| Tx/Rx FIFO size | 8 |

1. X = supported.

## 28.5 USART functional description

### 28.5.1 USART block diagram

**Figure 291. USART block diagram**




The simplified block diagram given in *Figure 291* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.

- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is disabled, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

## 28.5.2 USART signals

### USART bidirectional communications

USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)

  RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate 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 no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

### RS232 hardware flow control mode

The following pins are required in RS232 hardware flow control mode:

- **CTS** (Clear To Send)

  When driven high, this signal blocks the data transmission at the end of the current transfer.

- **RTS** (Request To Send)

  When it is low, this signal indicates that the USART is ready to receive data.

### RS485 hardware control mode

The following pin is required in RS485 hardware control mode:

- **DE** (Driver Enable)

  This signal activates the transmission mode of the external transceiver.

*Note: DE and RTS share the same pin.*

### Synchronous master/slave mode and smartcard mode

The following pin is required in synchronous master/slave mode and smartcard mode:

- CK

  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 pin. This mechanism can be used to control peripherals featuring shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.

  In smartcard mode, CK output provides the clock to the smartcard.

- NSS

  This pin acts as slave Select input in synchronous slave mode.

*Note: NSS and CTS share the same pin.*

### 28.5.3 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 292*):

- 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-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.*

By default, 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.

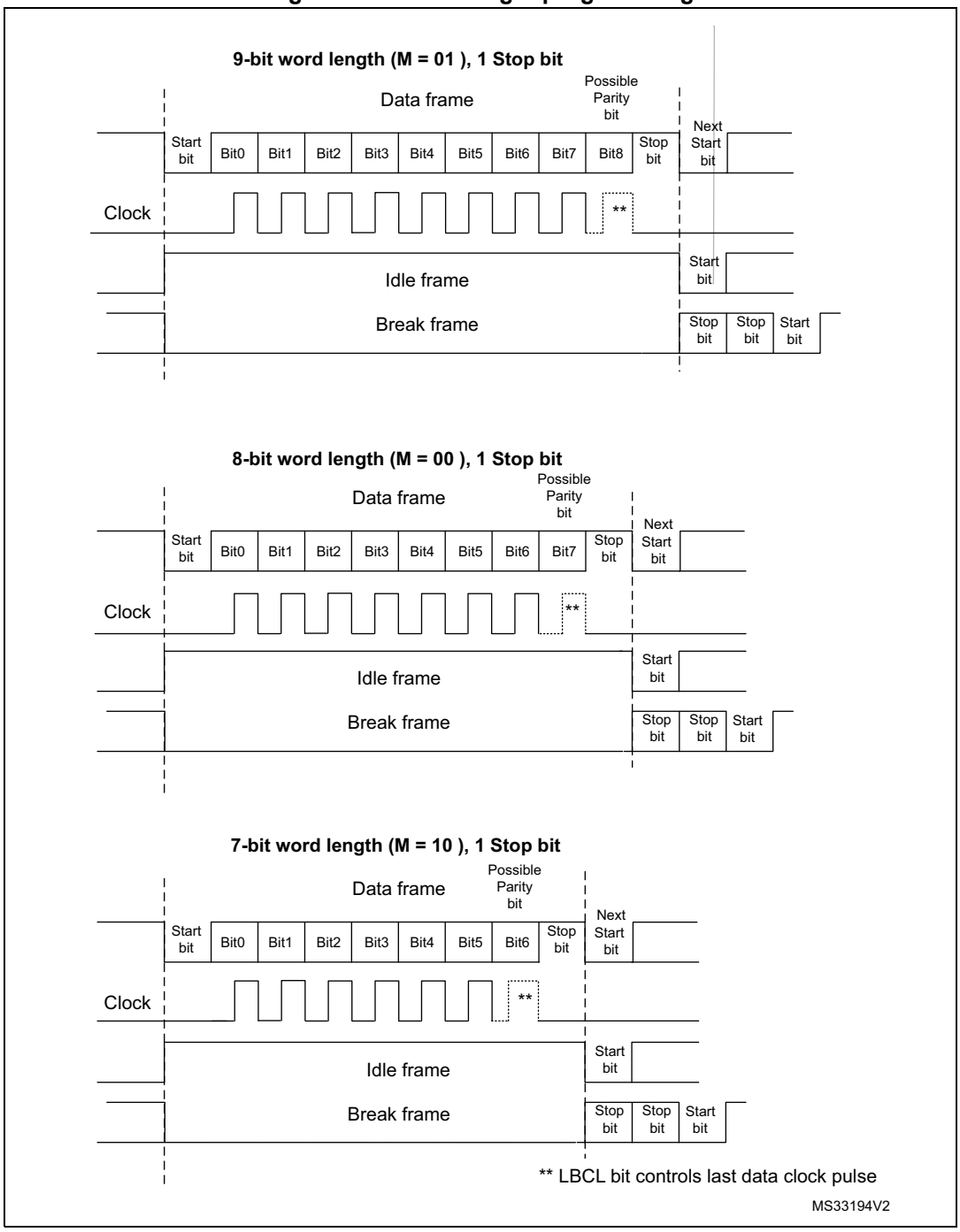
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

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 transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

**Figure 292. Word length programming**



### 28.5.4 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store 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 together with the corresponding flags. However, only the data are read when reading the RDR.*

*The status flags are available in the USART_ISR register.*

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART_CR3 control register.

In this case:

- The RXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

  This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

  RXFTCFG data have been received: one data in USART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101’, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data is not set the overrun flag.
- The TXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

### 28.5.5 USART 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 while the corresponding clock pulses are output on the CK pin.

#### Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

*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 data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.*

*An idle frame is sent when the TE bit is enabled.*

### Configurable stop bits

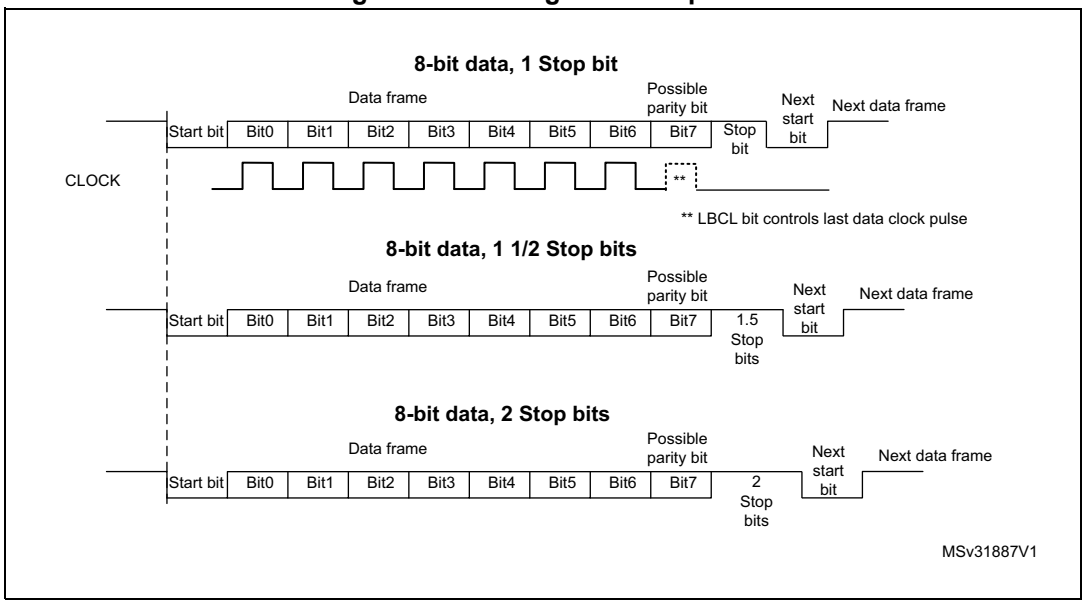
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***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 features 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 293*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 293. Configurable stop bits**



### Character transmission procedure

To transmit a character, follow the sequence below:

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 multibuffer communication must take place. Configure the DMA register as explained in *Section 28.5.19: Continuous communication using USART and DMA*.
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.
   - When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   - 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.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

### Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:

  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;

  – the USART_TDR register is empty;

  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_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 to indicate that:

  – the TXFIFO is not full;

  – the USART_TDR register is empty;

  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

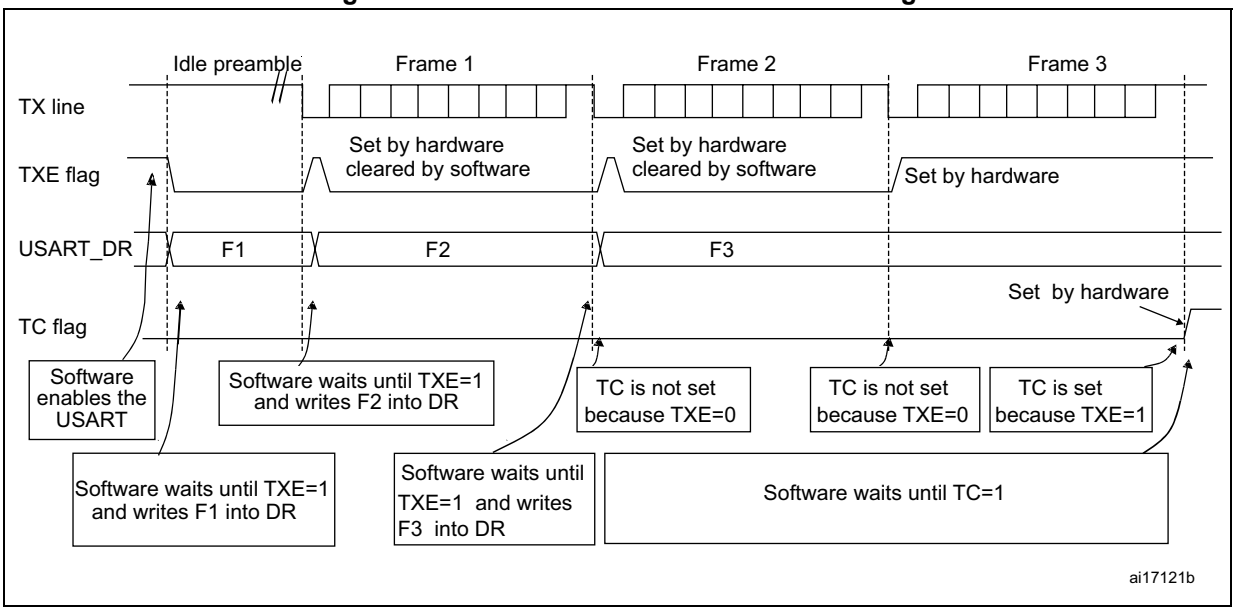
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger 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 to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the device to enter the low-power mode (see *Figure 294: TC/TXE behavior when transmitting*).

**Figure 294. 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 292*).

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.

## 28.5.6 USART receiver

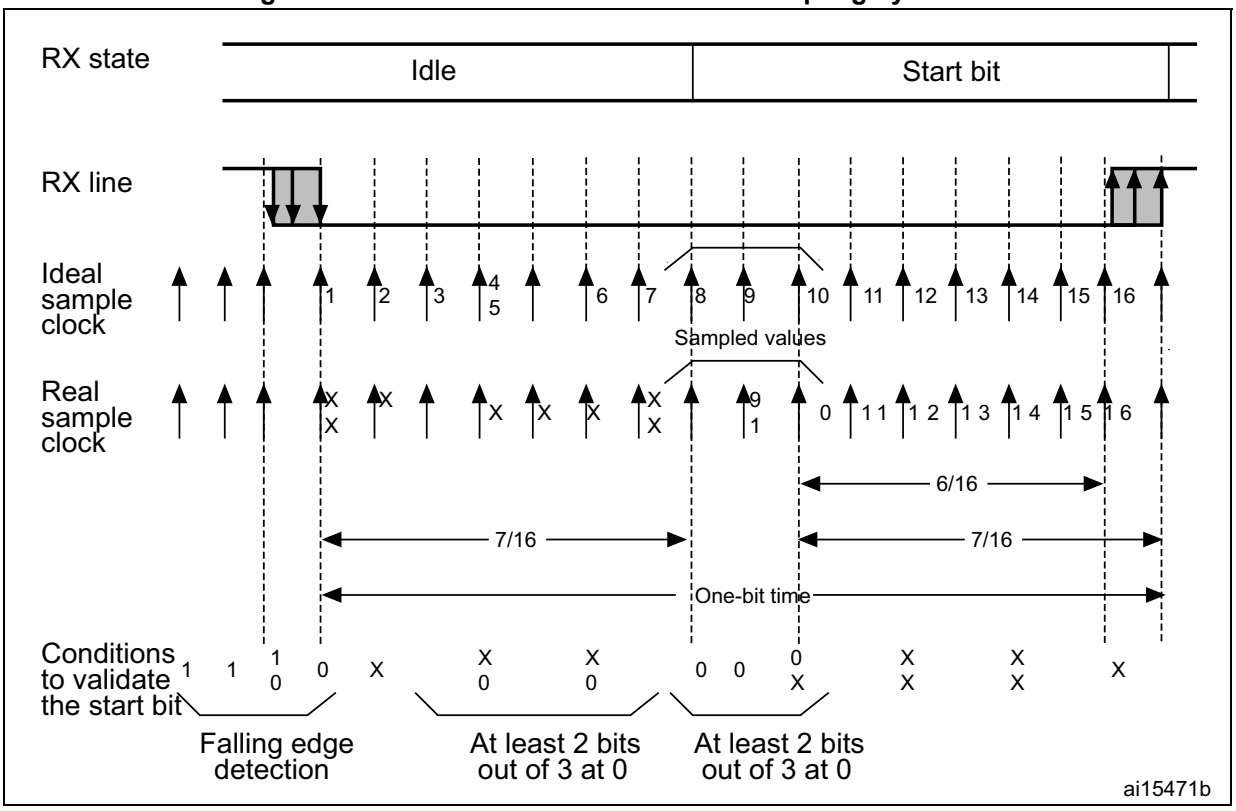
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

### 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 295. 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 and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if 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 NE 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 of the above conditions 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 are shifted out least significant bit first (default configuration) through the RX pin.

#### Character reception procedure

To receive a character, follow the sequence below:

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 *Section 28.5.19: Continuous communication using USART and DMA*.
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 to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads 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. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to ‘1’ in the USART_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 operation from 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 programming RXFRQ bit to ‘1’ in USART_RQR. 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, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

### Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and 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 reception.

  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 by reading the USART_RDR register.
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE or the EIE bit is set.
- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_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 by reading the USART_RDR register.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

*Note: The ORE bit, when set, indicates that at least 1 data has been lost.*

*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, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

### Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see Section *Reset and clock control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

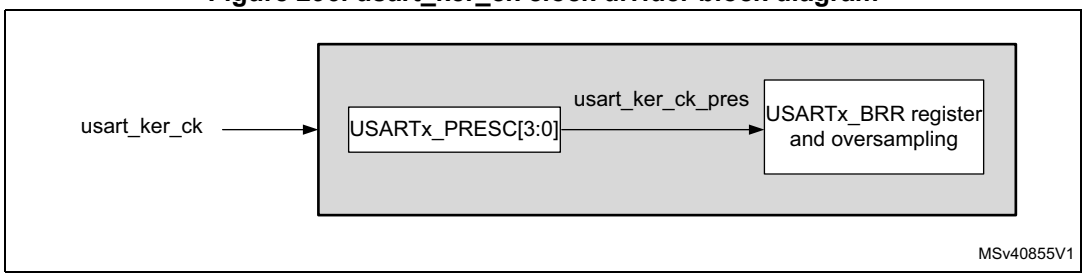
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see Section *Reset and clock control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

**Figure 296. usart_ker_ck clock divider block diagram**




Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

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 enables obtaining the best a trade-off 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 either to 16 or 8 times the baud rate clock (see *Figure 297* and *Figure 298*).

Depending on your application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 28.5.8: Tolerance of the USART receiver to clock deviation on page 919*)
- 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

usart_ker_ck_pres/16 (where usart_ker_ck_pres 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. Two options are available:

- 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 NE bit is set.
- A single sample in the center of the received bit

  Depending on your application:
  - select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Figure 171*) 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 tolerance to clock deviations (see *Section 28.5.8: Tolerance of the USART receiver to clock deviation on page 919*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE 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 NE bit is reset by setting NECF bit in USART_ICR register.

*Note: Noise error is not supported in SPI mode.*

*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 297. Data sampling when oversampling by 16**

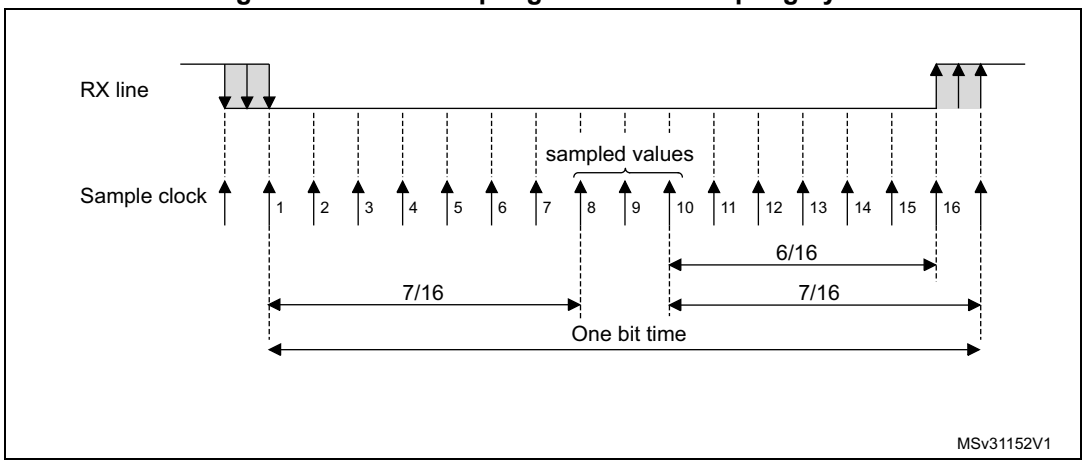


**Figure 298. Data sampling when oversampling by 8**

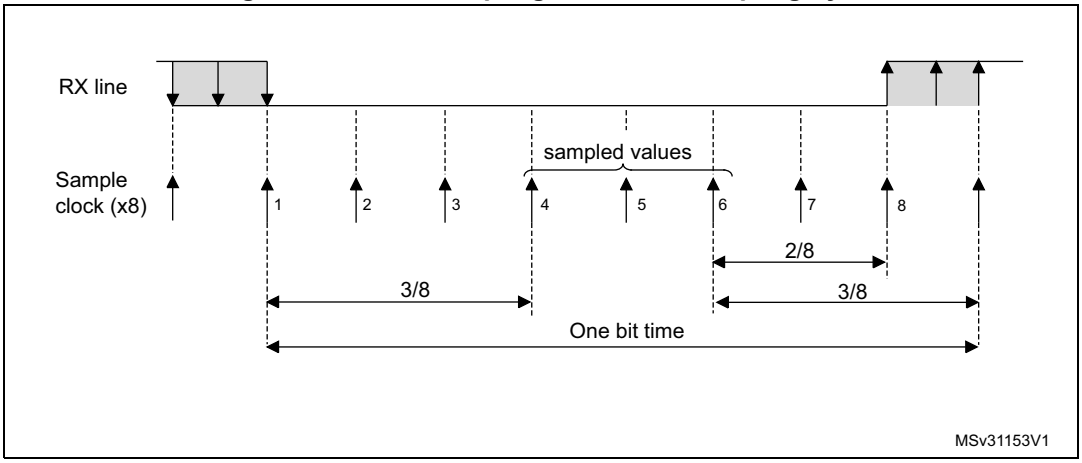



**Table 171. Noise detection from sampled data**

| 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 USART_CR: 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 is 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 are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) 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. The FE flag is then set through RXNE flag (RXFNE if the 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 broken 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 28.5.16: USART receiver timeout on page 933* 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.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

## 28.5.7 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value 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 baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ckpres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ckpres}}{\text{USARTDIV}}$$

**Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)**

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ckpres}}{\text{USARTDIV}}$$

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 16.*

### How to derive USARTDIV from USART_BRR register values

**Example 1**

To obtain 9600 baud with usart_ker_ck_pres = 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >> 1 = 0x1
  BRR = 0x681

**Example 2**

To obtain 921.6 Kbaud with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR = USARTDIV = 0d52 = 0x34
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR = 0x64

## 28.5.8 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates 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 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)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 172*, *Table 173*, depending on the following settings:

- 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 172. Tolerance of the USART receiver when BRR [3:0] = 0000**

| 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 173. Tolerance of the USART receiver when BRR[3:0] is different from 0000**

| 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 172 and Table 173 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).*

### 28.5.9 USART auto baud rate detection

The USART can 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 enables 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 ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. 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 the following:

- **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, to ensure a 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 baud rate 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 baud rate 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). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. 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 the 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 and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

*Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.*

### 28.5.10 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs 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 actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_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 usart_ker_ck cycles), otherwise mute mode might remain active.*

When the mute mode is enabled:

- 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:

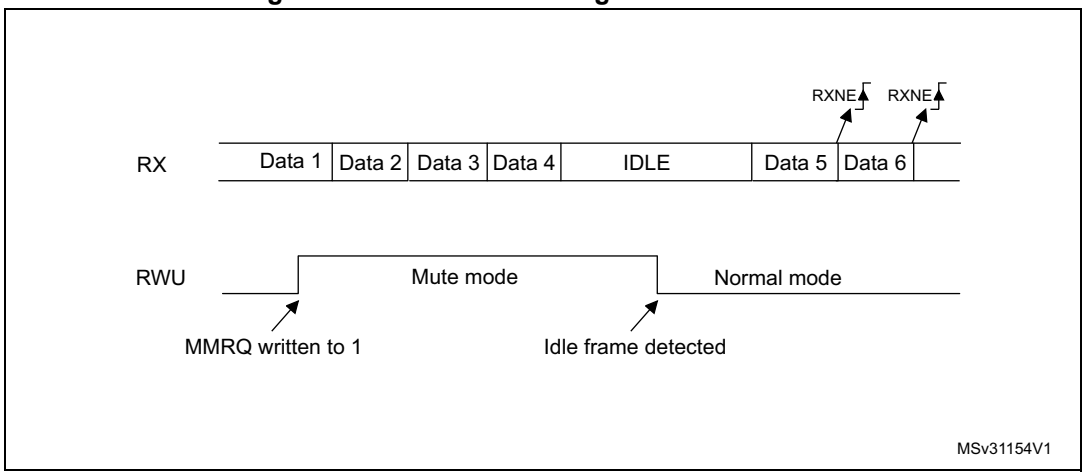
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

#### Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to ‘1’ and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then 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 299*.

**Figure 299. 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.

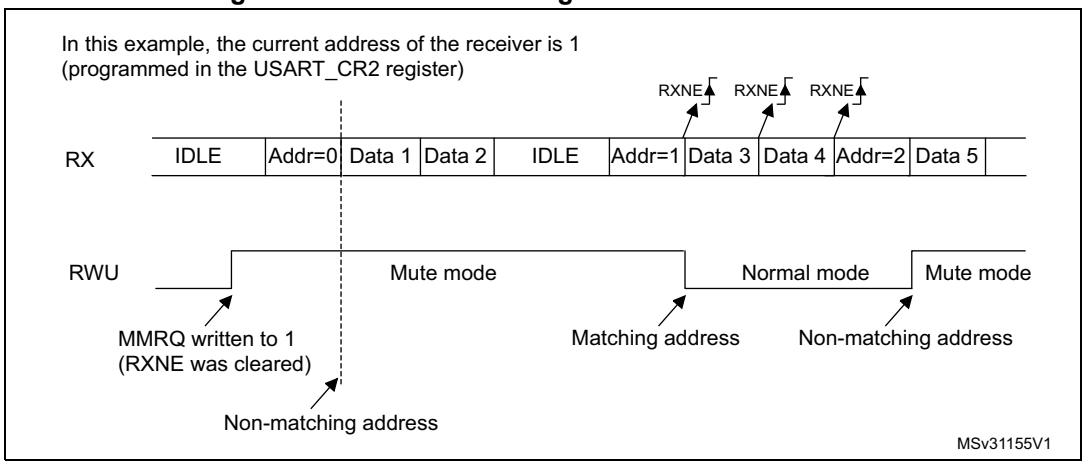
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 300*.

**Figure 300. Mute mode using address mark detection**




## 28.5.11 USART 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.

### 28.5.12 USART 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 174*.

**Table 174. USART 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, the parity bit is equal to 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 equal to 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).

### 28.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 28.4: USART implementation on page 902*.

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 described in *Section 28.5.4* 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 two bits of value ‘1 are sent to enable 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 301: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 928*.

Examples of break frames are given on *Figure 302: Break detection in LIN mode vs. Framing error detection on page 929*.

**Figure 301. Break detection in LIN mode (11-bit break length - LBDL bit is set)**

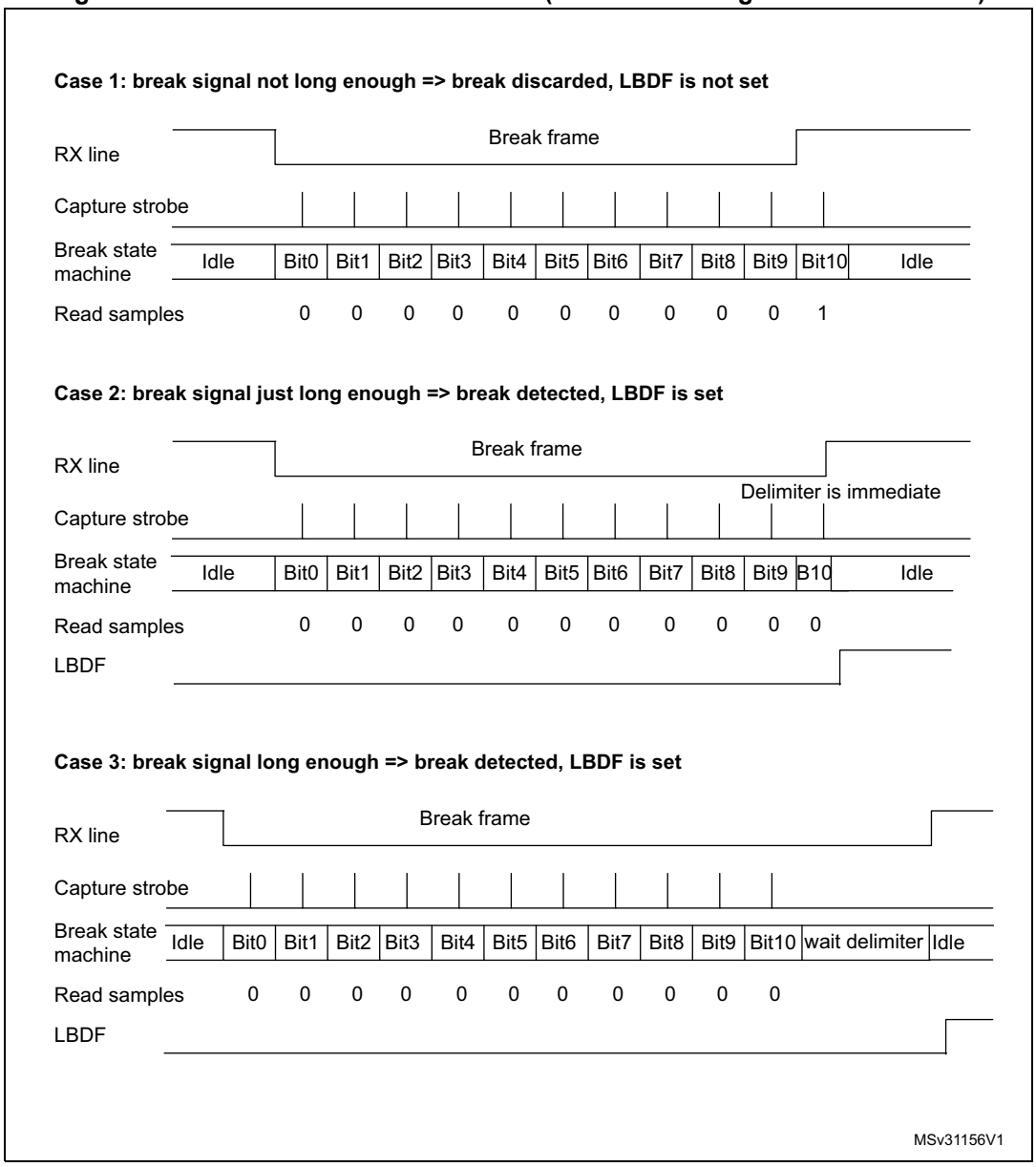
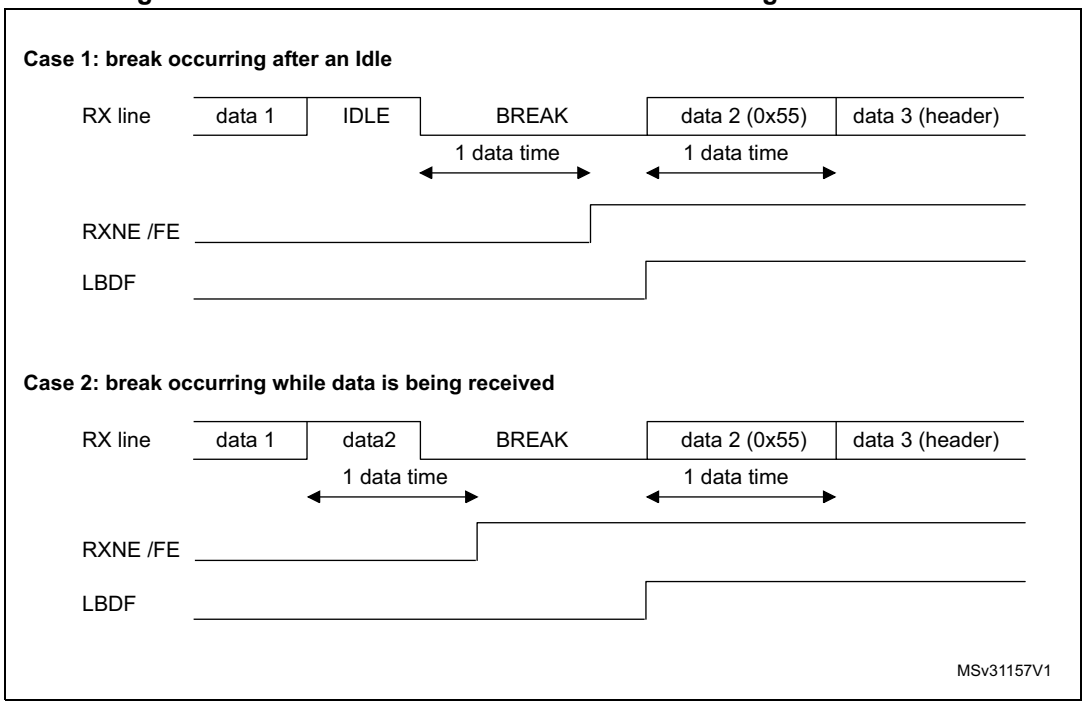


**Figure 302. Break detection in LIN mode vs. Framing error detection**




## 28.5.14 USART synchronous mode

### Master mode

The synchronous master mode is selected by programming 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 CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK 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 303*, *Figure 304* and *Figure 305*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

*Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.*

**Figure 303. USART example of synchronous master transmission**

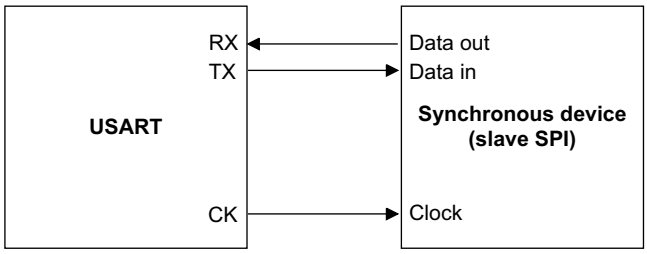



**Figure 304. USART data clock timing diagram in synchronous master mode (M bits = 00)**

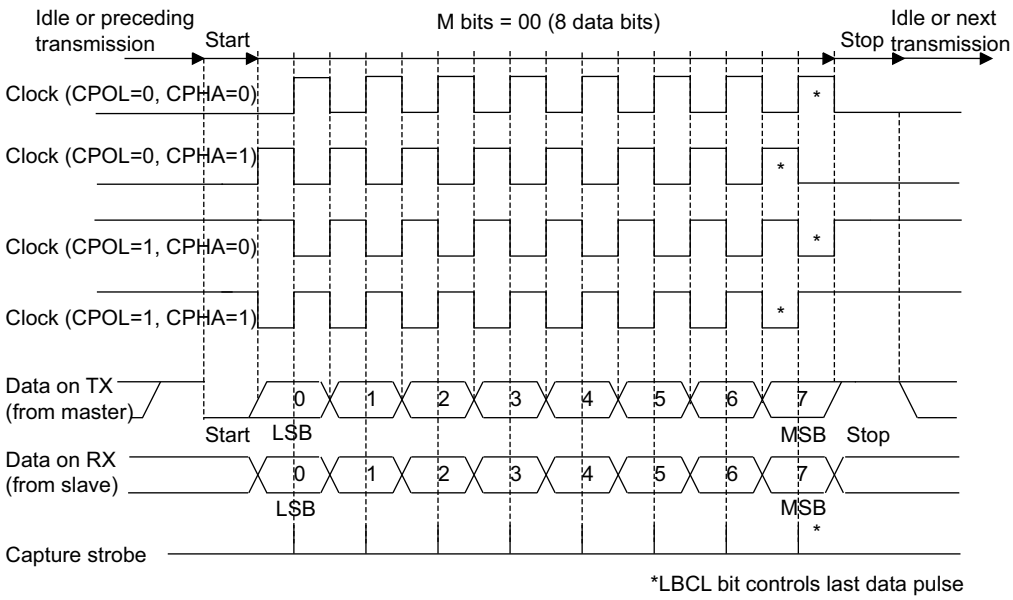
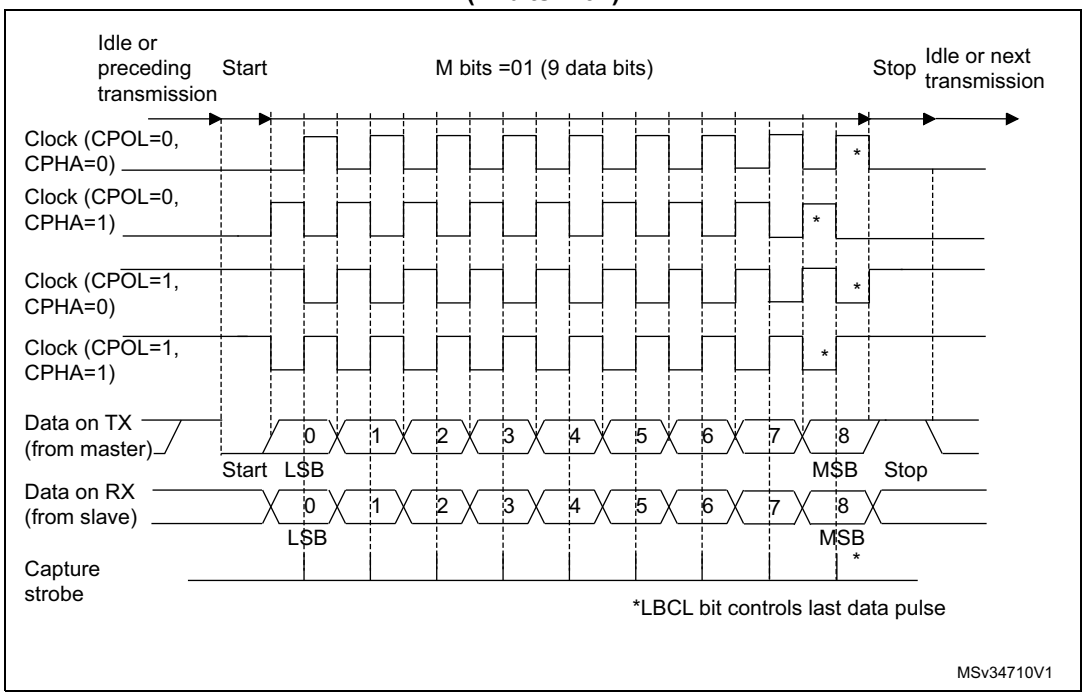


**Figure 305. USART data clock timing diagram in synchronous master mode (M bits = 01)**




## Slave mode

The synchronous slave mode is selected by programming 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 CK pin is the input of the USART in slave mode.

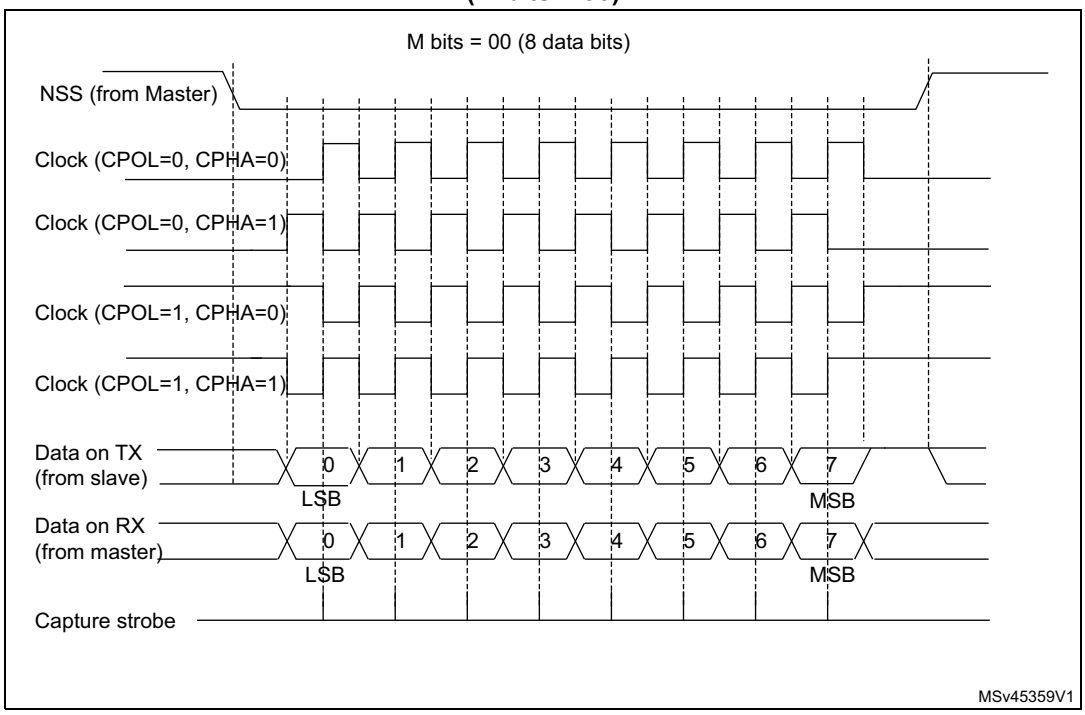
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 306*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

**Figure 306. USART data clock timing diagram in synchronous slave mode (M bits = 00)**




### Slave select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  The 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 is 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.*

*In SPI slave mode, the USART must be enabled before starting the master communications (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 UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending 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 moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR should be written 3* usart_ker_ck *cycles before the first CK edge.*

### 28.5.15 USART 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 conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

### 28.5.16 USART 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 bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = ‘00’ or STOP = ‘11’
- from the end of the second stop bit if STOP = ‘10’.
- from the beginning of the stop bit if STOP = ‘01’.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

### 28.5.17 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 28.4: USART implementation on page 902*.

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 can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

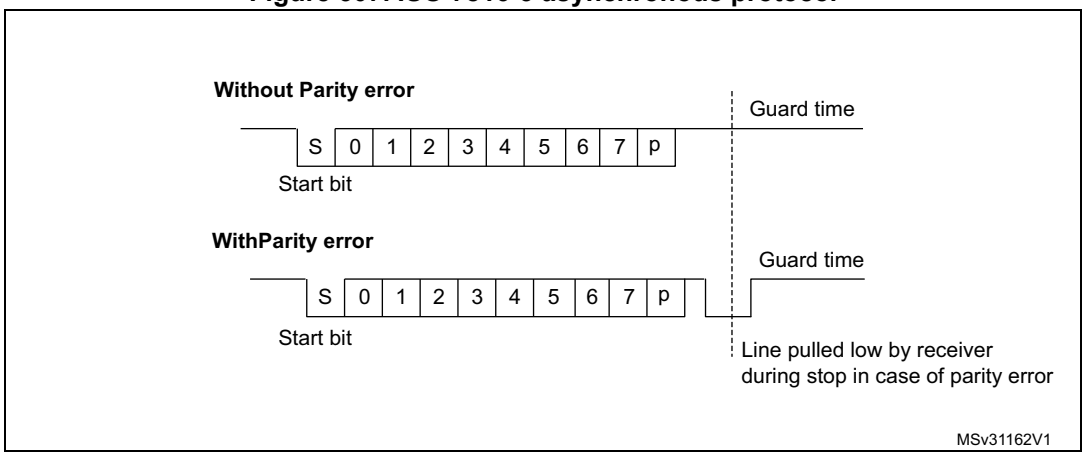
The USART should be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = '11' in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

*Figure 307* shows examples of what can be seen on the data line with and without parity error.

**Figure 307. 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 bitfield. 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.

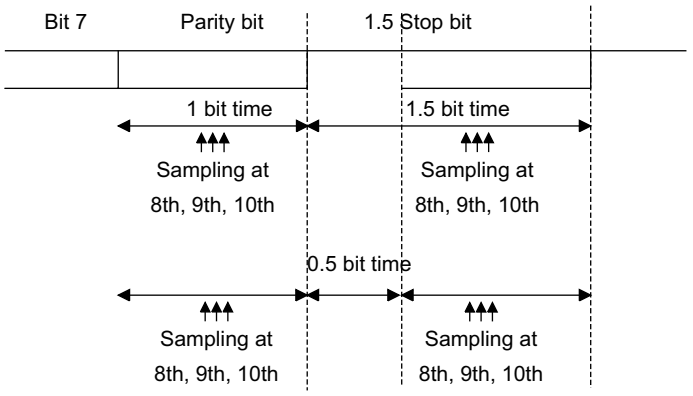
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it 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 bitfield, 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 deassertion 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: Break characters are 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 308* shows 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 308. Parity error detection using the 1.5 stop bits**




The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK 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 USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres 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 USART_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 enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it 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, assuming that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH results in a USART received character of 03 and an odd parity.

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 did not 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.

### 28.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 28.4: USART implementation on page 902*.

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 309*).

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 kbaud 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 310*).
- 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 two 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 309. IrDA SIR ENDEC block diagram**

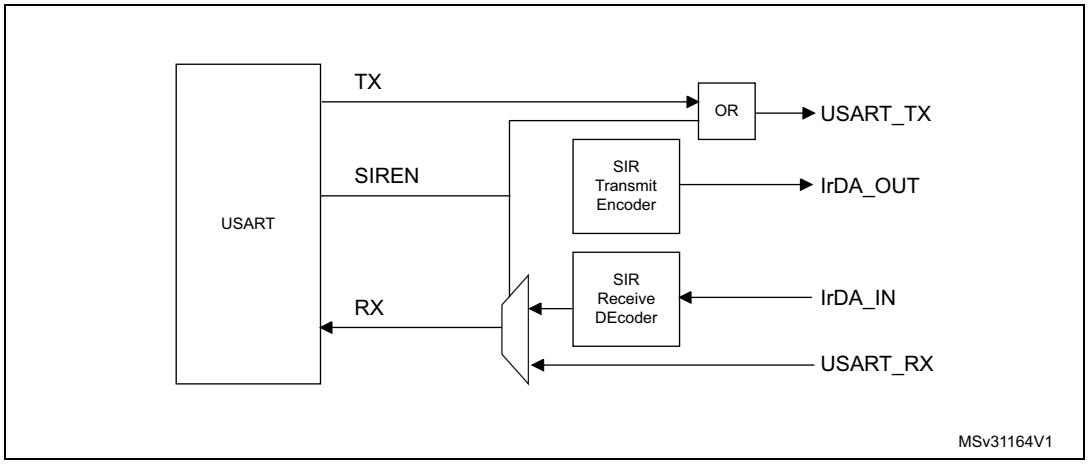
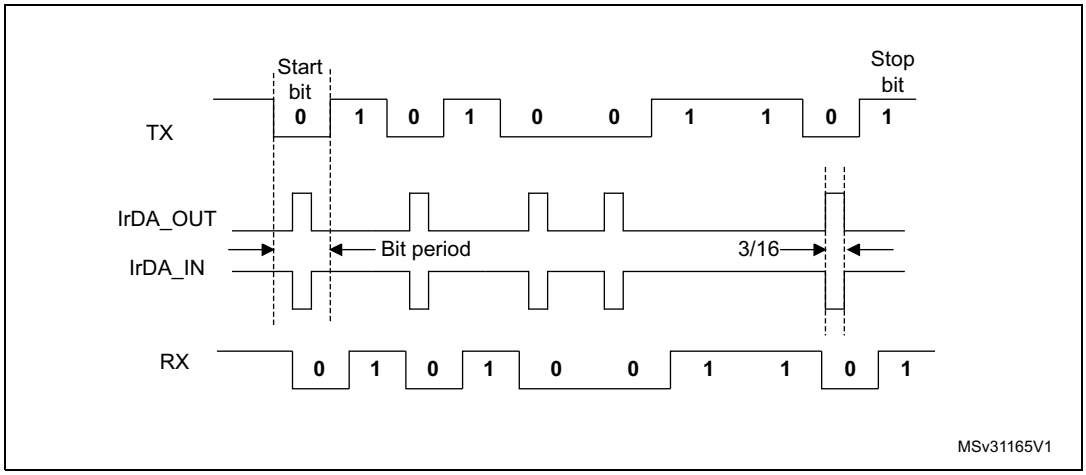



**Figure 310. IrDA data modulation (3/16) - normal mode**



### 28.5.19 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

*Note: Refer to Section 28.4: USART implementation on page 902 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 28.5.6. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.*

#### Transmission using DMA

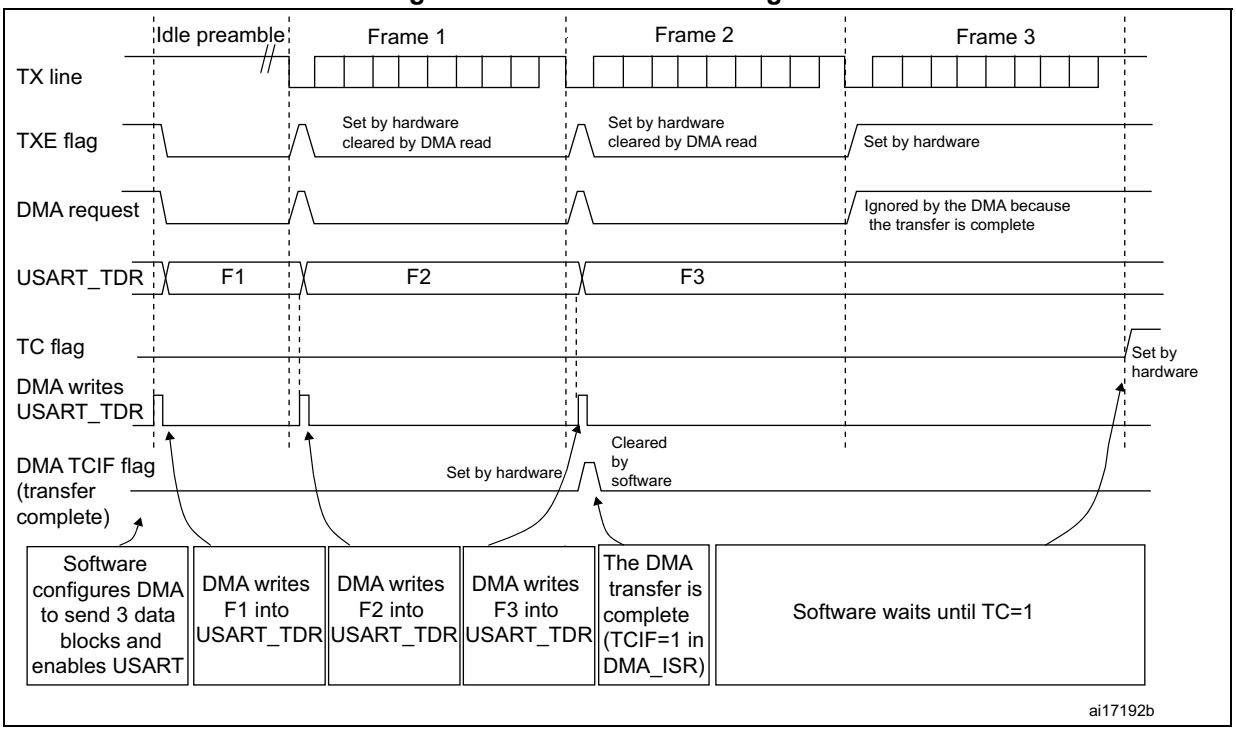
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode 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 before the system enters a low-power mode when the peripheral clock is disabled. 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 311. Transmission using DMA**




*Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).*

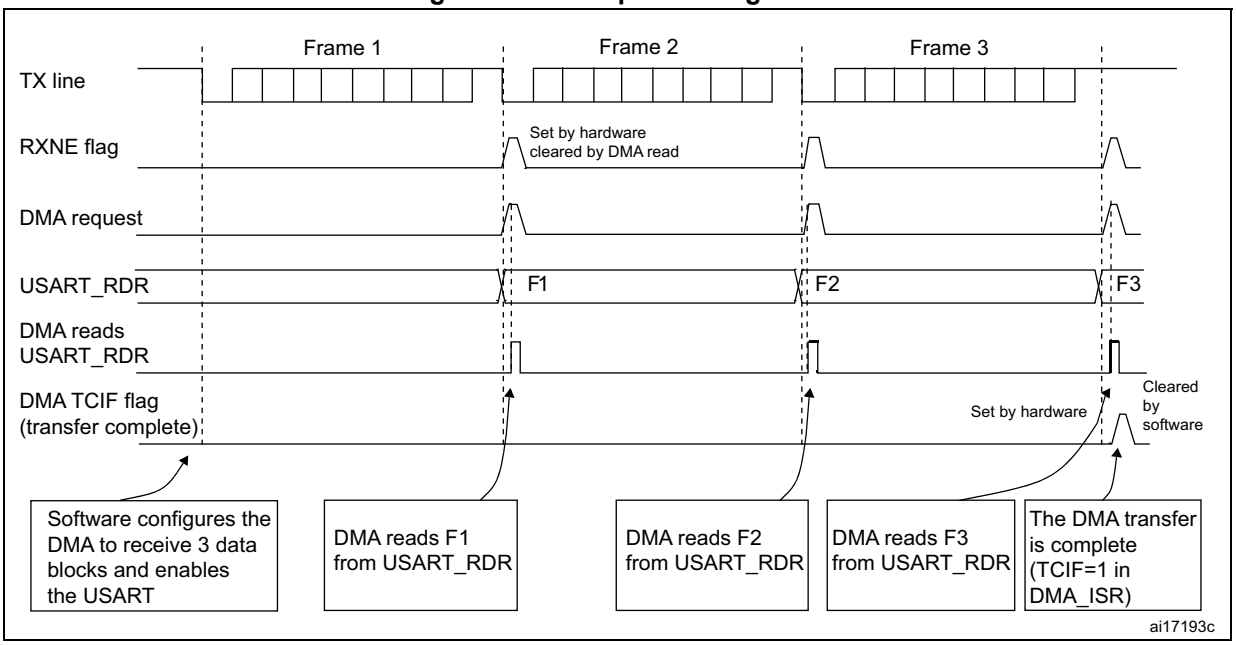
### Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) 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 312. Reception using DMA**




*Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).*

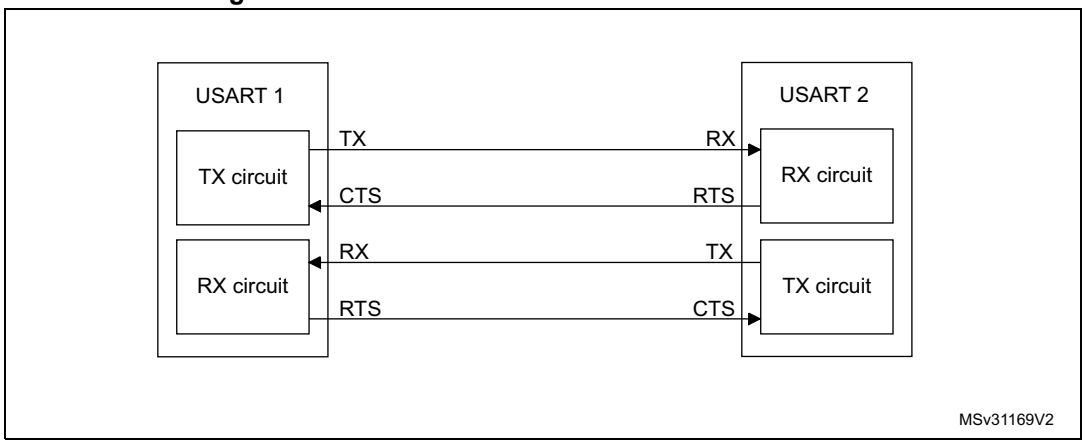
### Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, 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.

## 28.5.20 RS232 hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The *Figure 313* shows how to connect 2 devices in this mode:

**Figure 313. Hardware flow control between 2 USARTs**

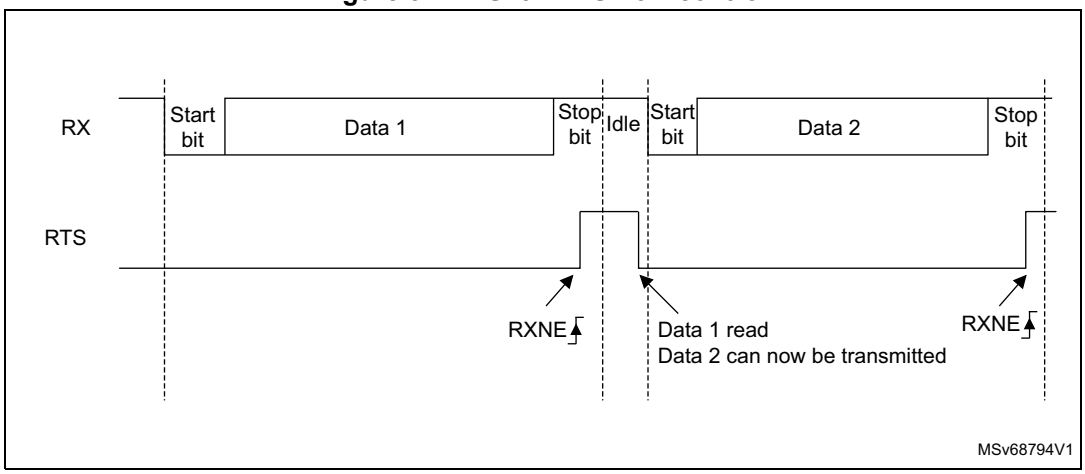


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to '1' in the USART_CR3 register.

### RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 314* shows an example of communication with RTS flow control enabled.

**Figure 314. RS232 RTS flow control**



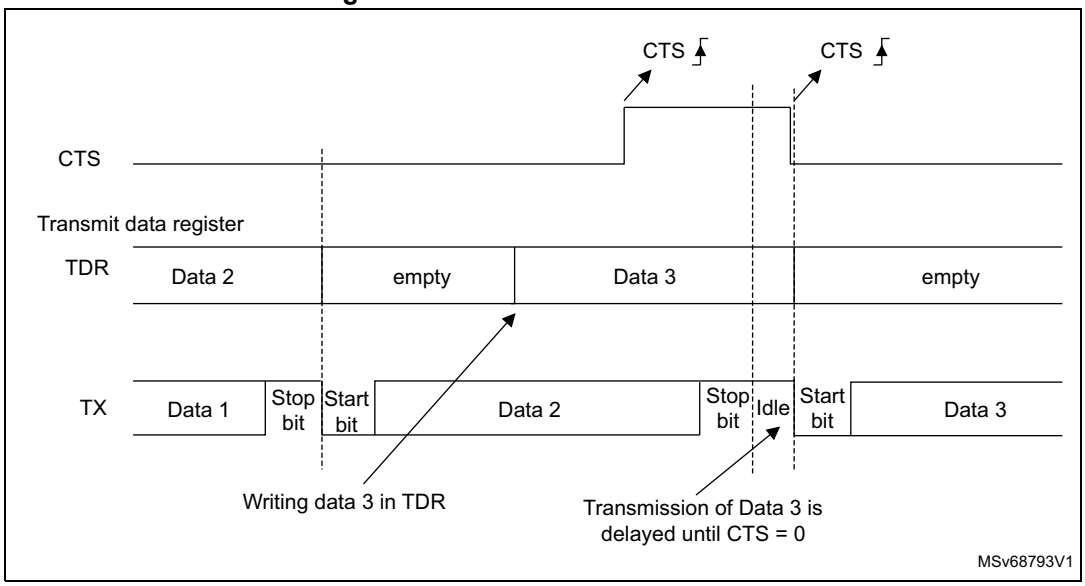

*Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.*

### RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (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 CTS is asserted 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 CTS 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 315* shows an example of communication with CTS flow control enabled.

**Figure 315. RS232 CTS flow control**




*Note: For correct behavior, CTS must be deasserted 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 enables 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] bitfields in the USART_CR1 control register. The deassertion 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] bitfields 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).

### 28.5.21 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wake-up interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to ‘1’. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - to fill the TXFIFO
  - or to empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set.

*Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.*

*The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.*

*When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.*

*When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.*

*When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.*

### Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

*Note: When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (i.e.the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).*

### Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

*Figure 316* shows the USART behavior when the wake-up event is verified.

**Figure 316. Wake-up event verified (wake-up event = address match, FIFO disabled)**

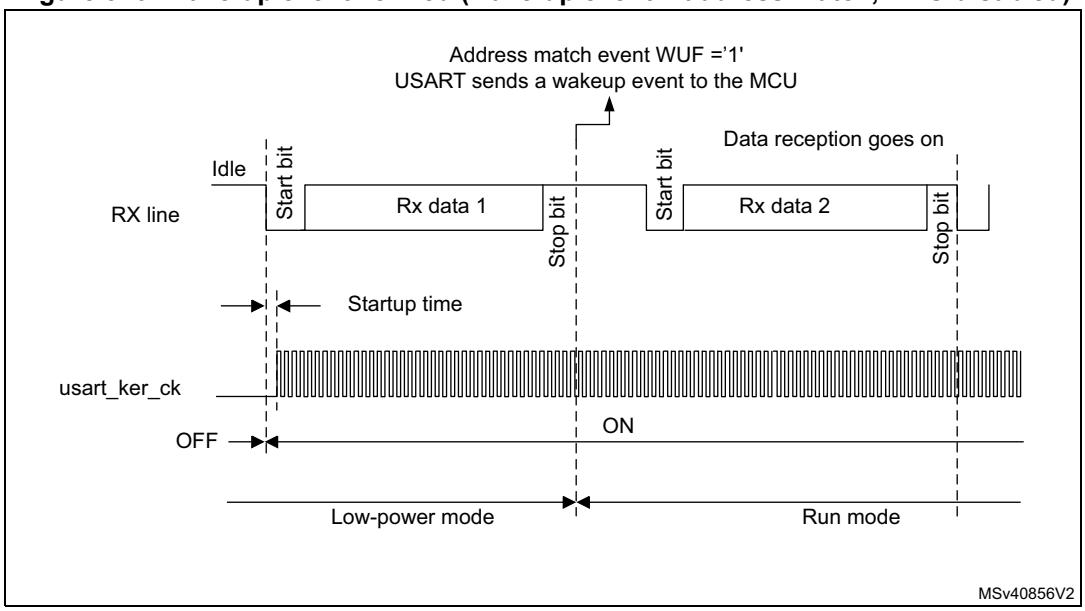
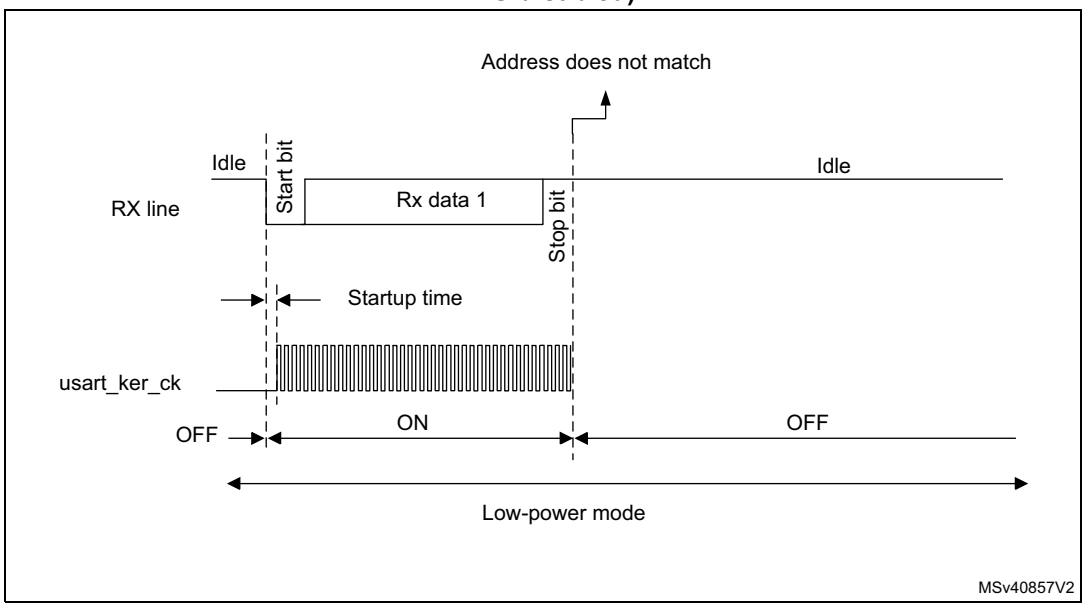



*Figure 317* shows the USART behavior when the wake-up event is not verified.

**Figure 317. Wake-up event not verified (wake-up event = address match, FIFO disabled)**




*Note: The figures above are valid when address match or any received frame is used as wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.*

### Determining the maximum USART baud rate that enables to correctly wake up the device from low-power mode

The maximum baud rate that enables to correctly wake up the device from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 28.5.8: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 172: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the usart_ker_ck inaccuracy.

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

$D_{WUmax}$ = 3.41% - 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 Kbaud.

## 28.6 USART in low-power modes

**Table 175. Effect of low-power modes on the USART**

| Mode | Description |
|---|---|
| Sleep | No effect. USART interrupts cause the device to exit Sleep mode. |
| Stop(1) | The content of the USART registers is kept.<br>The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode. |
| Standby | The USART peripheral is powered down and must be reinitialized after exiting Standby mode. |

1. Refer to *Section 28.4: USART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

## 28.7 USART interrupts

Refer to *Table 176* for a detailed description of all USART interrupt requests.

**Table 176. USART interrupt requests**

| Interrupt vector | Interrupt event | Event flag | Enable Control bit | Interrupt clear method | Exit from Sleep mode | Exit from Stop[1] modes | Exit from Standby mode |
|---|---|---|---|---|---|---|---|
| USART or UART | Transmit data register empty | TXE | TXEIE | Write TDR | Yes | No | No |
| | Transmit FIFO not Full | TXFNF | TXFNFIE | TXFIFO full | | No | |
| | Transmit FIFO Empty | TXFE | TXFEIE | Write TDR or write 1 in TXFRQ | | Yes | |
| | Transmit FIFO threshold reached | TXFT | TXFTIE | Write TDR | | Yes | |
| | CTS interrupt | CTSIF | CTSIE | Write 1 in CTSCF | | No | |
| | Transmission Complete | TC | TCIE | Write TDR or write 1 in TCCF | | No | |
| | Transmission Complete Before Guard Time | TCBGT | TCBGTIE | Write TDR or write 1 in TCBGT | | No | |
| USART or UART | Receive data register not empty (data ready to be read) | RXNE | RXNEIE | Read RDR or write 1 in RXFRQ | Yes | Yes | No |
| | Receive FIFO Not Empty | RXFNE | RXFNEIE | Read RDR until RXFIFO empty or write 1 in RXFRQ | | Yes | |
| | Receive FIFO Full | RXFF[2] | RXFFIE | Read RDR | | Yes | |
| | Receive FIFO threshold reached | RXFT | RXFTIE | Read RDR | | Yes | |
| | Overrun error detected | ORE | RXNEIE/ RXFNEIE | Write 1 in ORECF | | No | |
| | Idle line detected | IDLE | IDLEIE | Write 1 in IDLECF | | No | |
| | Parity error | PE | PEIE | Write 1 in PECF | | No | |
| | LIN break | LBDF | LBDIE | Write 1 in LBDCF | | No | |
| | Noise error in multibuffer communication | NE | EIE | Write 1 in NFCF | | No | |
| | Overrun error in multibuffer communication | ORE[3] | | Write 1 in ORECF | | No | |
| | Framing Error in multibuffer communication | FE | | Write 1 in FECF | | No | |
| | Character match | CMF | CMIE | Write 1 in CMCF | | No | |
| | Receiver timeout | RTOF | RTOFIE | Write 1 in RTOCCF | | No | |
| | End of Block | EOBF | EOBIE | Write 1 in EOBCF | | No | |
| | Wake-up from low-power mode | WUF | WUFIE | Write 1 in WUC | | Yes | |
| | SPI slave underrun error | UDR | EIE | Write 1 in UDRCF | | No | |

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 28.4: USART implementation* for the list of supported Stop modes.

2. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_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).
3. When OVRDIS = 0.

## 28.8 USART registers

Refer to *Section 1.2 on page 51* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

### 28.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

**FIFO mode enabled**

| 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 | TXFNFIE | TCIE | RXFNEIE | IDLEIE | TE | RE | UESM | UE |
| rw | 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 inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

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 bitfield 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 must be used in conjunction with bit 12 (M0) to 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 auto baud rate (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 inhibited

1: USART interrupt 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt 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 must be kept at reset value. Section 28.4: USART implementation on page 902.*

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 bitfield 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 at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 bitfield 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 at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, 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 is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield 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 the 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 bitfield 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 bitfield 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 inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not-full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXFNF =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt 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 low 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'. 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 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset 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.*

*In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.*

### 28.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | FIFO EN | M1 | EOBIE | RTOIE | 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 |
| OVER8 | CMIE | MME | M0 | WAKE | PCE | PS | PEIE | TXEIE | TCIE | RXNEIE | IDLEIE | TE | RE | UESM | UE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:30 Reserved, 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 bitfield 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 must be used in conjunction with bit 12 (M0) to 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 auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bit 27 **EOBIE**: End of Bbock interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt 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 must be kept at reset value. Section 28.4: USART implementation on page 902.*

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 bitfield 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 at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 bitfield 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 at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, 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 is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield 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 the 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 bitfield 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 bitfield 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 inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXE =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt 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 low 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'. 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 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset 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.*

*In smartcard mode, (SCEN = 1), the CK pin is always available when CLKEN = 1, regardless of the UE bit value.*

### 28.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD[7:0] | | | | | | | | 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 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bits 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 and Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 bitfield 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 bitfield 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 enables the use of an external inverter on the TX line.

This bitfield 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 enables the use of an external inverter on the RX line.

This bitfield 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 enables to work in the case of a cross-wired connection to another UART.

This bitfield 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 synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

*In smartcard mode, in order to provide correctly the CK 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 enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 297* and *Figure 298*)

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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**: NSS pin enable

When the DIS_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. Refer to Section 28.4: USART implementation on page 902.*

Bits 2:1 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. Refer to Section 28.4: USART implementation on page 902.*

*Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.*

## 28.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXFTCFG[2:0] | | | RXF TIE | RXFTCFG[2:0] | | | TCBG TIE | TXFTIE | WUFIE | WUS[1:0] | | SCARCNT[2:0] | | | Res. |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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[2:0]:** 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

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]:** 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 inhibited
1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register

*Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF = 1 in the USART_ISR register

*Note: WUFIE must be set before entering in low-power mode.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power 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/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

*If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception 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 bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 14 **DEM**: Driver enable mode

This bit enables 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 at reset value. Section 28.4: USART implementation on page 902.*

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 case 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 USART_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 enables checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 RTS output is deasserted (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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 bitfield 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 bitfield 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

### 28.8.5 USART baud rate register (USART_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 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 15:0 **BRR[15:0]**: USART baud rate

**BRR[15:4]**

BRR[15:4] = USARTDIV[15:4]

**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.

### 28.8.6 USART guard time and prescaler register (USART_GTPR)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GT[7:0] | | | | | | | | 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 **GT[7:0]**: Guard time value

This bitfield 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 bitfield can only be written when the USART is disabled (UE = 0).

*Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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

PSC[7:0] is used to program 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):

**In smartcard mode:**

PSC[4:0] = Prescaler value

PSC[4:0] is used to program 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 1 (IrDA mode) / by 2 (smartcard mode)
00010: Divides the source clock by 2 (IrDA mode) / by 4 (smartcard mode)
00011: Divides the source clock by 3 (IrDA mode) / by 6 (smartcard mode)
...
11111: Divides the source clock by 31 (IrDA mode) / by 62 (smartcard mode)
0010 0000: Divides the source clock by 32 (IrDA mode)
...
1111 1111: Divides the source clock by 255 (IrDA mode)

This bitfield 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 bitfield is reserved and forced by hardware to '0' when the smartcard and IrDA modes are not supported. Refer to Section 28.4: USART implementation on page 902.*

## 28.8.7 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

| 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 bitfield 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 bitfield 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 bitfield gives the Receiver timeout value in terms of number of bits during which there is no activity on the RX line.

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 28.4: USART implementation on page 902.*

### 28.8.8 USART request register (USART_RQR)

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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TXFRQ | RXFRQ | MMRQ | SBKRQ | ABRRQ |
| | | | | | | | | | | | w | w | w | w | w |

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 enables to discard the transmit data. This bit must be used only in smartcard mode, when data have 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 must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (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 are 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 enables 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: When 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 and ABRE flags in the USART_ISR and requests 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

## 28.8.9 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0X80 00C0

X = 2 if FIFO/smartcard mode is enabled

X = 0 if FIFO is enabled and smartcard mode is disabled

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

**FIFO mode enabled**

| 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 | WUF | RWU | SBKF | CMF | BUSY |
| | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ABRF | ABRE | UDR | EOBF | RTOF | CTS | CTSIF | LBDF | TXFNF | TC | RXFNE | IDLE | ORE | NE | 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 threshold programmed in TXFTCFG of USART_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 does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in RXFTCFG in USART_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 does not 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 USART_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 flag

This bit is set 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 the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by 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 unsuccessfully (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 kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is '1'. Refer to Section 28.4: USART implementation on page 902.*

Bit 24 **RXFF**: RXFIFO full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.

0: RXFIFO not full.

1: RXFIFO 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 not empty.

1: TXFIFO 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 low-power mode.

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up 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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART_RQR register

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 (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXFNE 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 kept at reset value.*

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_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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_CR1 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 kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 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 CTS input pin.

0: CTS line set

1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS 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 CTS status line

1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

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 kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to 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.

An interrupt is generated if the TXFNFIE bit =1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

*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).*

*This bit is used during single buffer transmission.*

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR 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 immediately set.*

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if 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 RXFNE 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 USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 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 **NE**: 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 NECF 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 RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

*When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 28.5.8: Tolerance of the USART receiver to clock deviation on page 919).*

*This error is associated with the character in the USART_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.

When transmitting data in smartcard mode, 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_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

*Note: This error is associated with the character in the USART_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: This error is associated with the character in the USART_RDR.*

## 28.8.10 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | TCBGT | Res. | Res. | RE ACK | TE ACK | WUF | RWU | SBKF | CMF | BUSY |
| | | | | | | 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 | TC | RXNE | IDLE | ORE | NE | FE | PE |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set 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 the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by 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 unsuccessfully (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 kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is '1'. Refer to Section 28.4: USART implementation on page 902.*

Bits 24:23 Reserved, must be kept at reset value.

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 low-power mode.

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 19 **RWU**: *Receiver wake-up from mute mode*

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up 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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART_RQR register

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 kept at reset value.*

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_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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_CR1 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 kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 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 CTS input pin.

0: CTS line set

1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS 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 CTS status line

1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

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 kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_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).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register not full

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR 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**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 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 USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 or EIE = 1 in the LPUART_CR1 register, or EIE = 1 in the USART_CR3 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 **NE**: 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 NECF 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 bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

*When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 28.5.8: Tolerance of the USART receiver to clock deviation on page 919).*

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.

When transmitting data in smartcard mode, 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_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

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

## 28.8.11 USART interrupt flag clear register (USART_ICR)

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. | WUCF | Res. | Res. | CMCF | Res. |
| | | | | | | | | | | | w | | | 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 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

Bits 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.

Bits 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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902*

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 must be kept at reset value. Refer to Section 28.4: USART implementation on page 902.*

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 NE 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.

### 28.8.12 USART receive data register (USART_RDR)

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. | 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 291*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 28.8.13 USART transmit data register (USART_TDR)

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. | 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 USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 291*).

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.*

### 28.8.14 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

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. | 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 factor:

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
Remaining 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.*

## 28.8.15 USART register map

The table below gives the USART register map and reset values.

**Table 177. USART 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | USART_CR1 FIFO enabled | RXFFIE | TXFEIE | FIFOEN | M1 | EOBIE | RTOIE | DEAT[4:0] | | | | | DEDT[4:0] | | | | | OVER8 | CMIE | MME | M0 | WAKE | PCE | PS | PEIE | TXFNFIE | TCIE | RXFNEIE | 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 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00 | USART_CR1 FIFO disabled | Res. | Res. | FIFOEN | M1 | EOBIE | RTOIE | DEAT[4:0] | | | | | DEDT[4:0] | | | | | OVER8 | CMIE | MME | M0 | 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 | 0 | 0 | 0 | 0 |
| 0x04 | USART_CR2 | ADD[7:0] | | | | | | | | 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 | TXFTCFG[2:0] | | | RXFTIE | RXFTCFG[2:0] | | | TCBGTIE | TXFTIE | WUFIE | WUS [1:0] | | 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 | 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 FIFO mode enabled | Res. | Res. | Res. | Res. | TXFT | RXFT | TCBGT | RXFF | TXFE | REACK | TEACK | WUF | RWU | SBKF | CMF | BUSY | ABRF | ABRE | UDR | EOBF | RTOF | CTS | CTSIF | LBDF | TXFNF | TC | RXFNE | IDLE | ORE | NE | FE | PE |
| | Reset value | | | | | X | X | X | X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1C | USART_ISR FIFO mode disabled | Res. | Res. | Res. | Res. | Res. | Res. | TCBGT | Res. | Res. | REACK | TEACK | WUF | RWU | SBKF | CMF | BUSY | ABRF | ABRE | UDR | EOBF | RTOF | CTS | CTSIF | LBDF | TXE | TC | RXNE | IDLE | ORE | NE | FE | PE |
| | Reset value | | | | | | | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x20 | USART_ICR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | WUCF | Res. | Res. | CMCF | Res. | Res. | Res. | UDRCF | EOBCF | RTOCF | Res. | CTSCF | LBDCF | TCBGTCF | TCCF | TXFECF | IDLECF | ORECF | NECF | FECF | PECF |
| | Reset value | | | | | | | | | | | | 0 | | | 0 | | | | 0 | 0 | 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 |

**Table 177. USART 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| 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: Memory organization* for the register boundary addresses.

# 29 Serial peripheral interface (SPI)

## 29.1 Introduction

The SPI interface can be used to communicate with external devices using the SPI protocol. SPI 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.

## 29.2 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 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
- Enhanced TI and NSS pulse modes support

## 29.3 SPI implementation

The following table describes all the SPI instances and their features embedded in the devices.

**Table 178. SPI implementation**

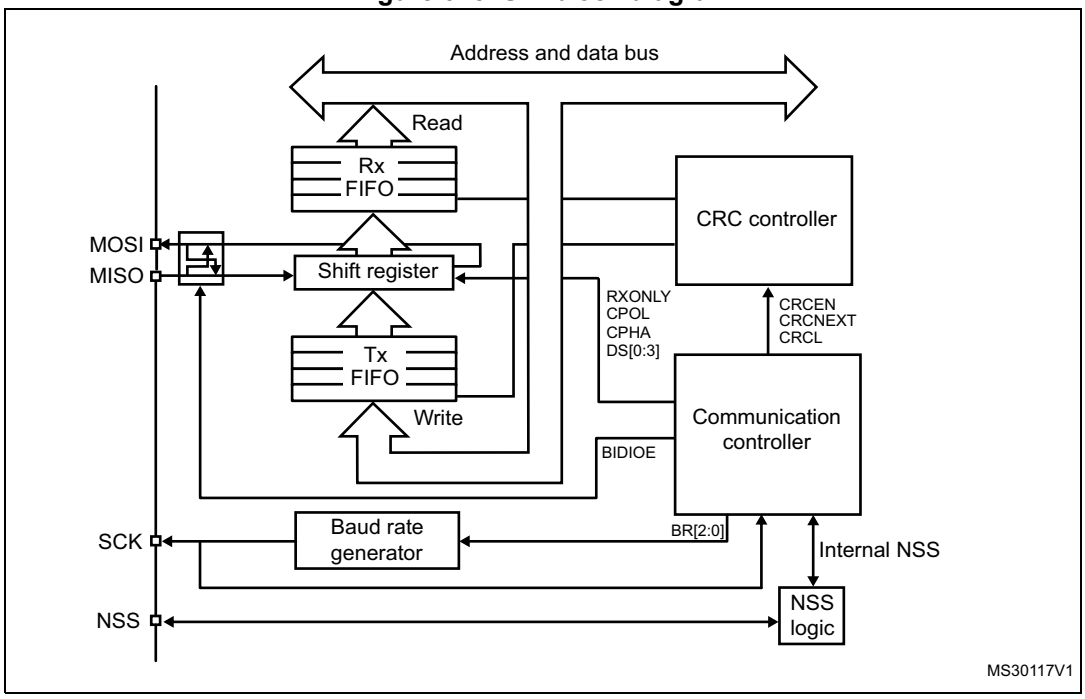
| SPI Features | SPI1 |
|---|---|
| Enhanced NSSP & TI modes | Yes |
| $I^2S$ support | No |
| Hardware CRC calculation | Yes |
| Data size configurable | from 4 to 16 bits |
| Rx/Tx FIFO size | 32 bits |
| Wake-up capability from Low-power Sleep | Yes |

## 29.4 SPI functional description

### 29.4.1 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 318*.

**Figure 318. 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.
- **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 29.4.5: 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.

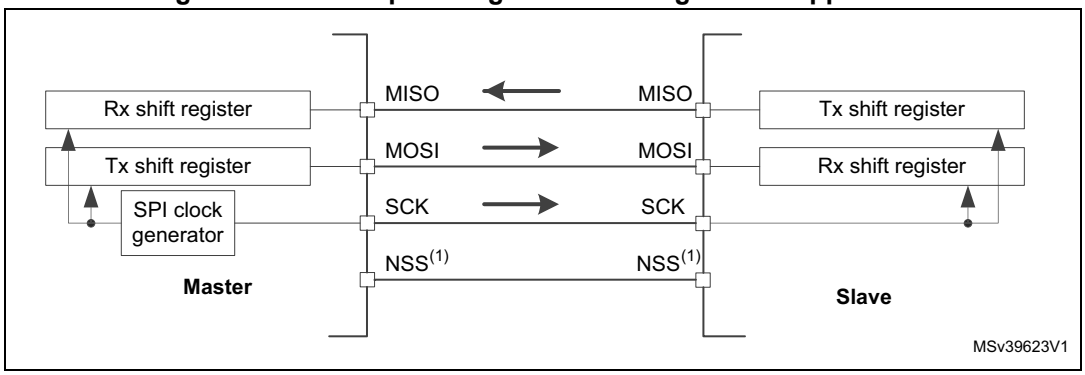
## 29.4.2 Communications between one master and one slave

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 319. Full-duplex single master/ single slave application**

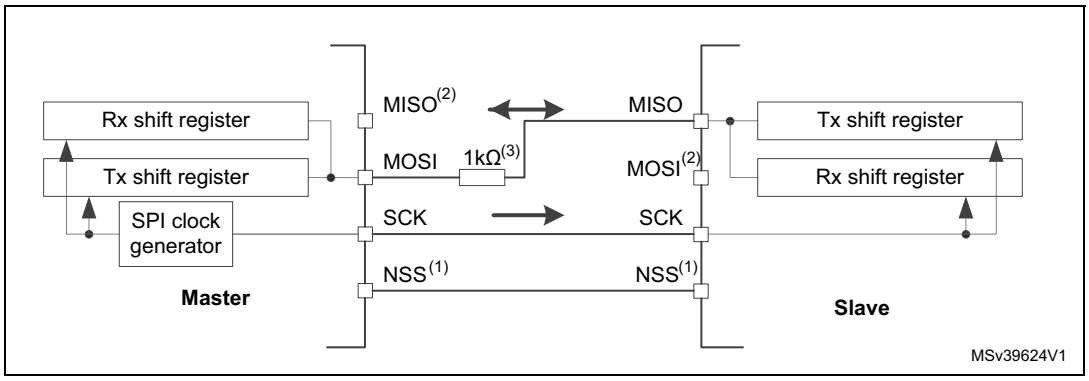



1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 29.4.5: Slave select (NSS) pin management*.

### 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 320. Half-duplex single master/ single slave application**



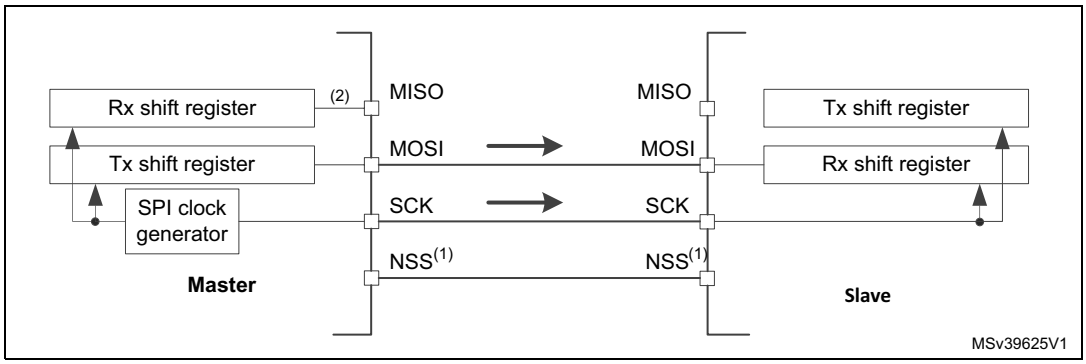

1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 29.4.5: Slave select (NSS) pin management*.
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.
3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectionnal mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communication data). Both nodes then fight while providing opposite output levels on the common line temporary till next node changes its direction settings correspondingly, too. It is suggested to insert a serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation.

### 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_CR1 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 *29.4.5: 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 321. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)**



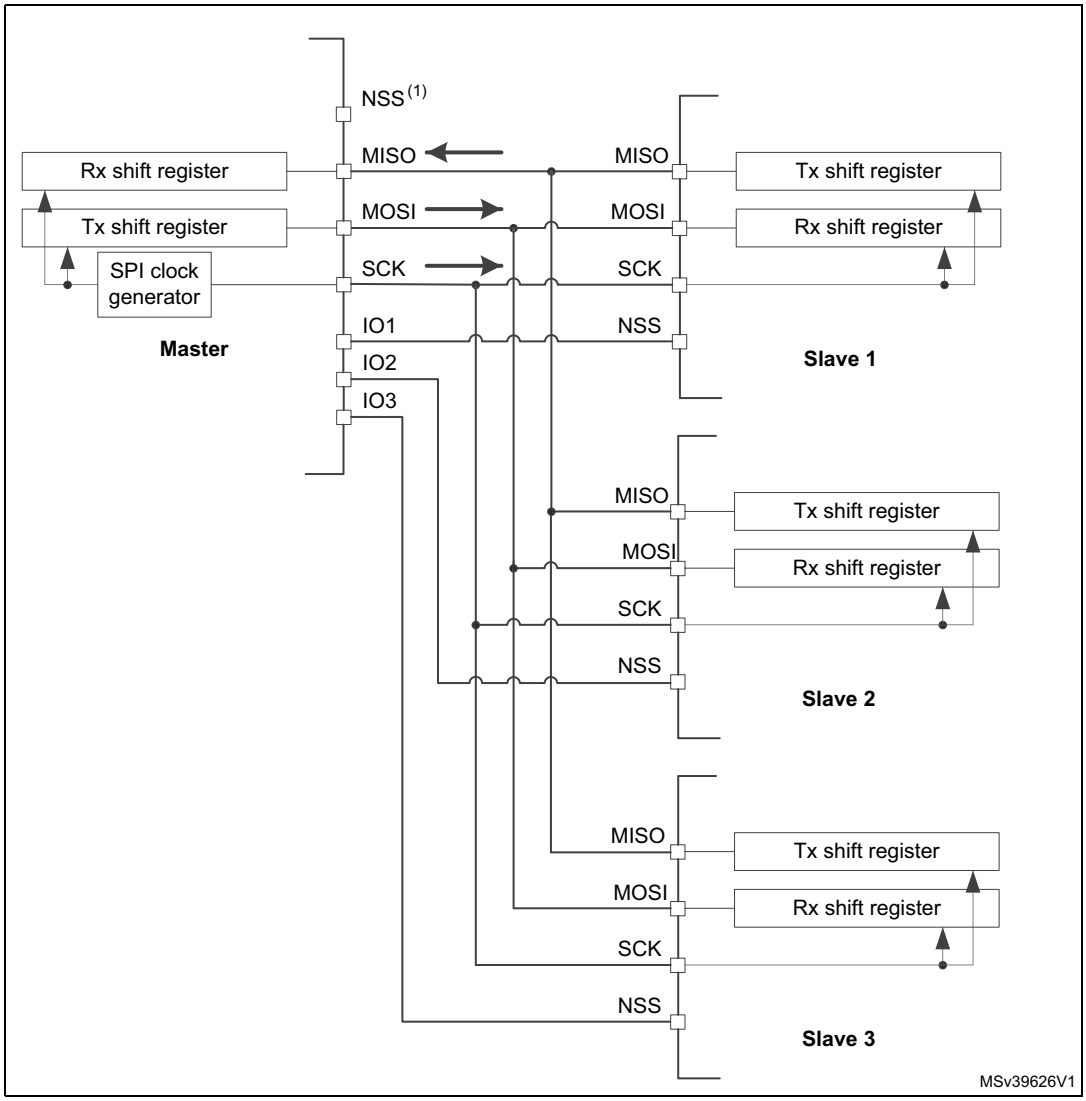

1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 29.4.5: Slave select (NSS) pin management*.
2. An accidental input information is captured at the input of transmitter Rx shift register. All the events associated with the transmitter receive flow must be ignored in standard transmit only mode (e.g. OVR 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).*

### 29.4.3 Standard multislave 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 322*). 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 322. Master and three independent slaves**




1. NSS pin is not used on master side at this configuration. It has to be managed internally (SSM=1, SSI=1) to prevent any MODF error.
2. 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 I/O alternate function input/output section (GPIO)).

## 29.4.4 Multimaster communication

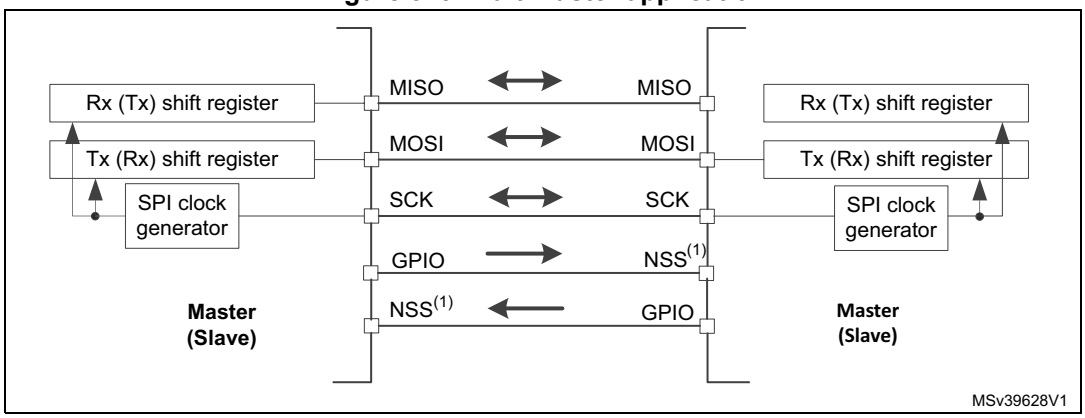
Unless SPI bus is not designed for a multimaster capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

**Figure 323. Multimaster application**




1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.

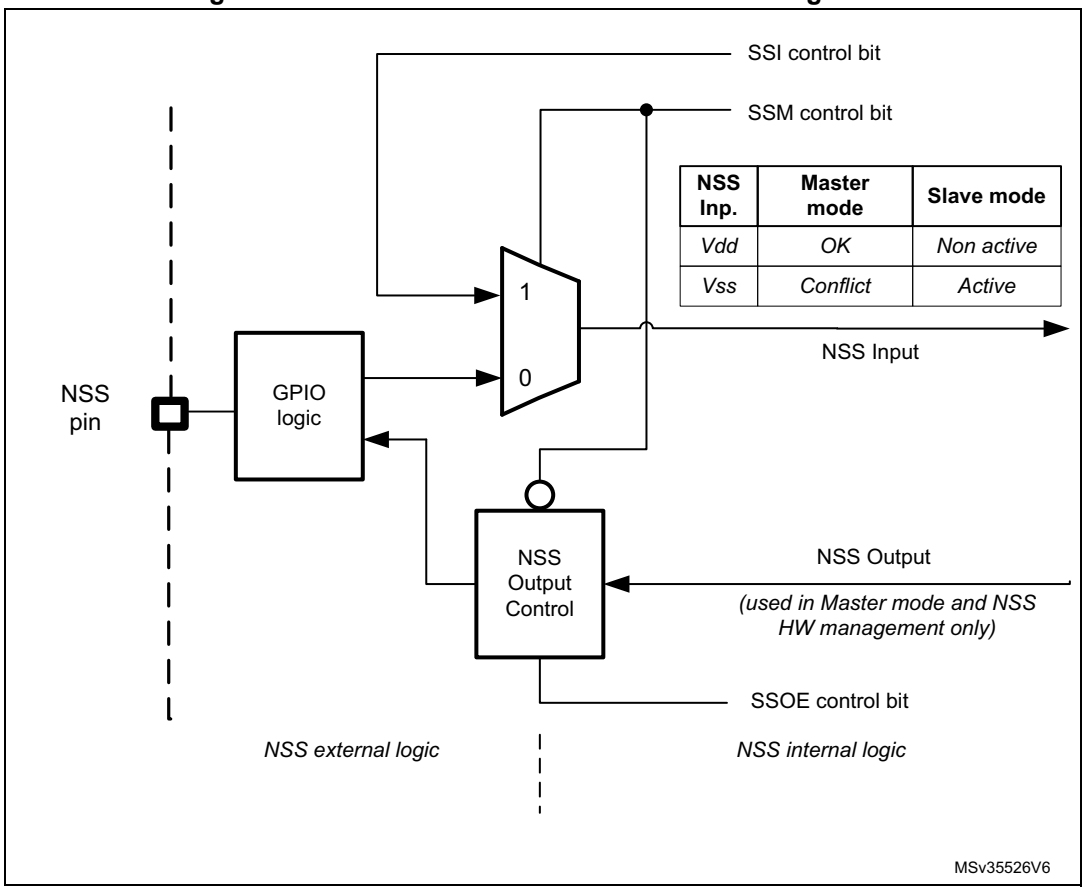
## 29.4.5 Slave select (NSS) pin management

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 324. Hardware/software slave select management**




## 29.4.6 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.

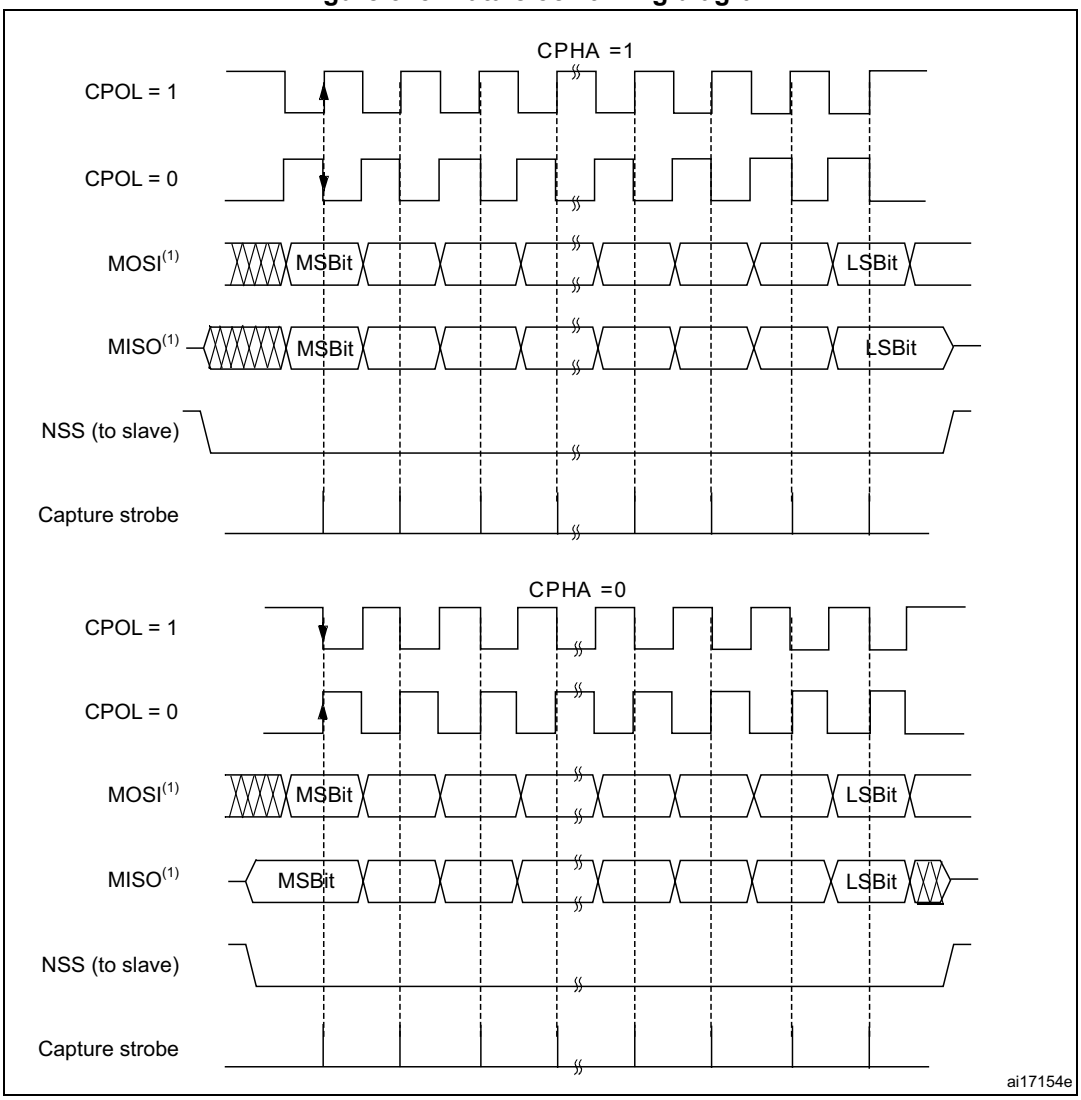
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

*Figure 325*, 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 325. Data clock timing diagram**

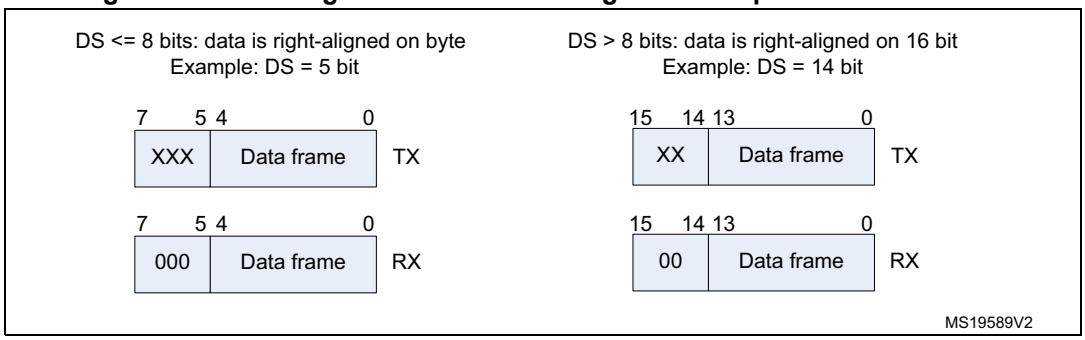



1. The order of data bits depends on LSBFIRST bit setting.

## 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 326*). During communication, only bits within the data frame are clocked and transferred.

**Figure 326. 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.*

### 29.4.7 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated sections. 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 - except the case when CRC is enabled at TI mode).
   c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE cannot 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 (Notes: 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*

### 29.4.8 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 section.

### 29.4.9 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 29.4.14: 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 29.4.13: 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 328* through *Figure 331*.

Another way to manage the data exchange is to use DMA (see *Communication using DMA (direct memory addressing)*).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at *Section 29.4.10: SPI status 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 29.4.5: 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.*

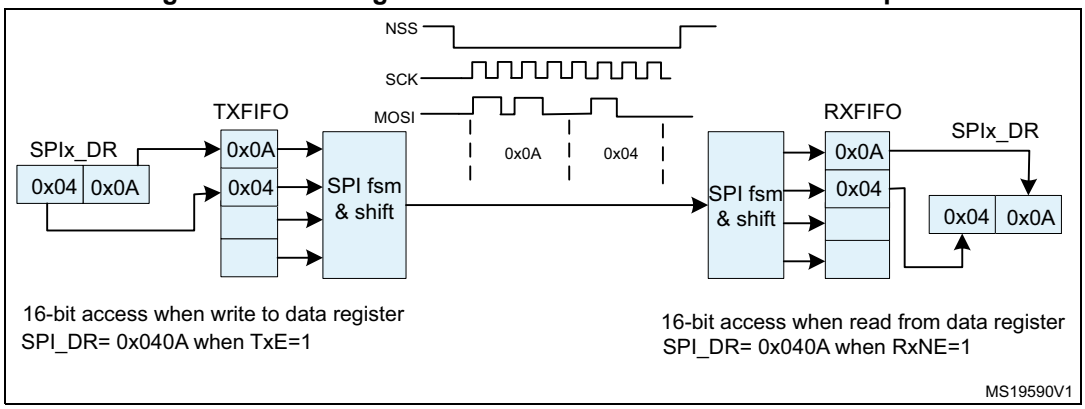
### Data packing

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 327* 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 327. 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 TXDMAEN or RXDMAEN 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 328* through *Figure 331*.

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 on page 999*.)

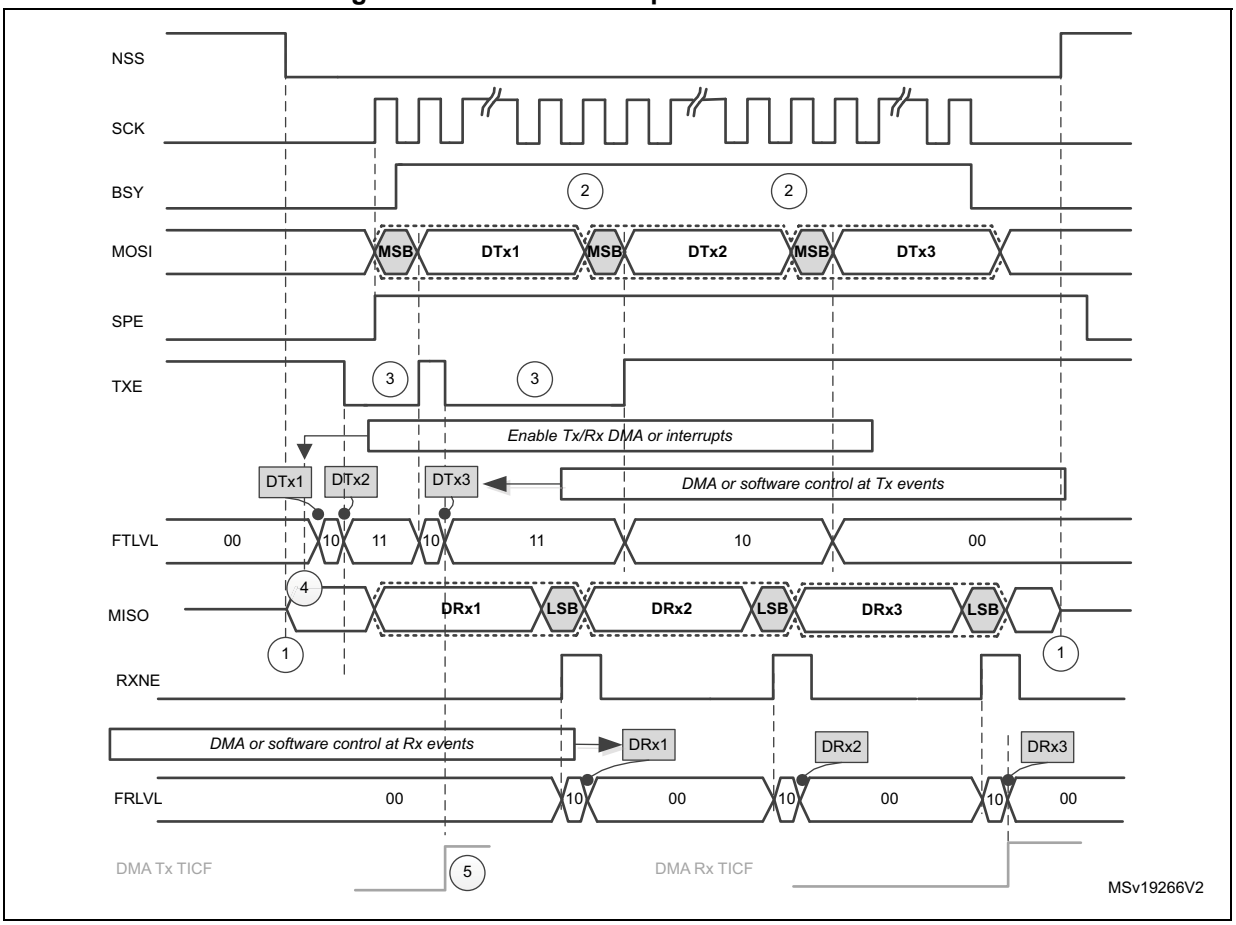
### Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by polling, 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 328 on page 1003* through *Figure 331 on page 1006*:

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 with an 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 328. Master full-duplex communication**




Assumptions for master full-duplex communication example:

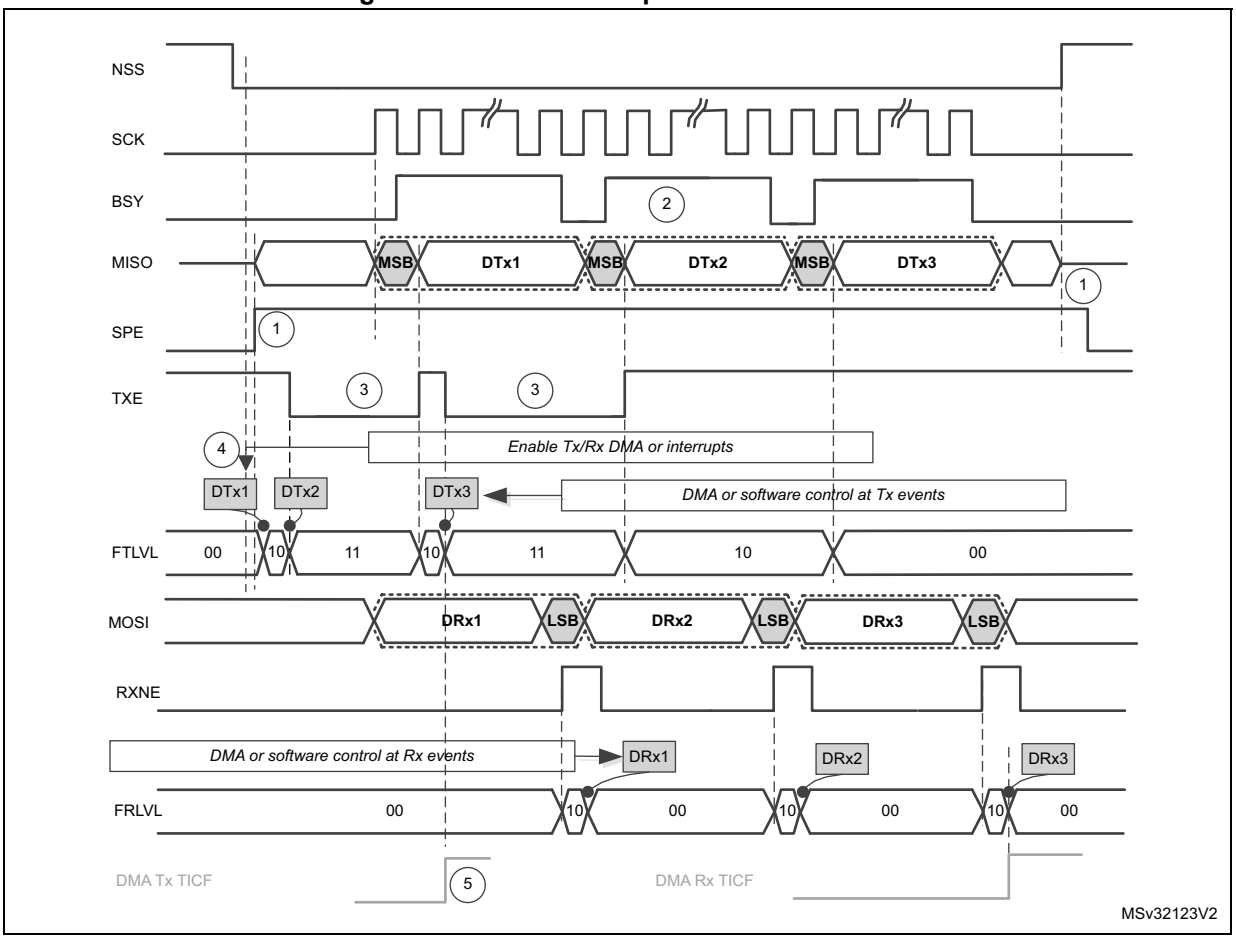
- Data size > 8 bit

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 on page 1002* for details about common assumptions and notes.

**Figure 329. Slave full-duplex communication**




Assumptions for slave full-duplex communication example:

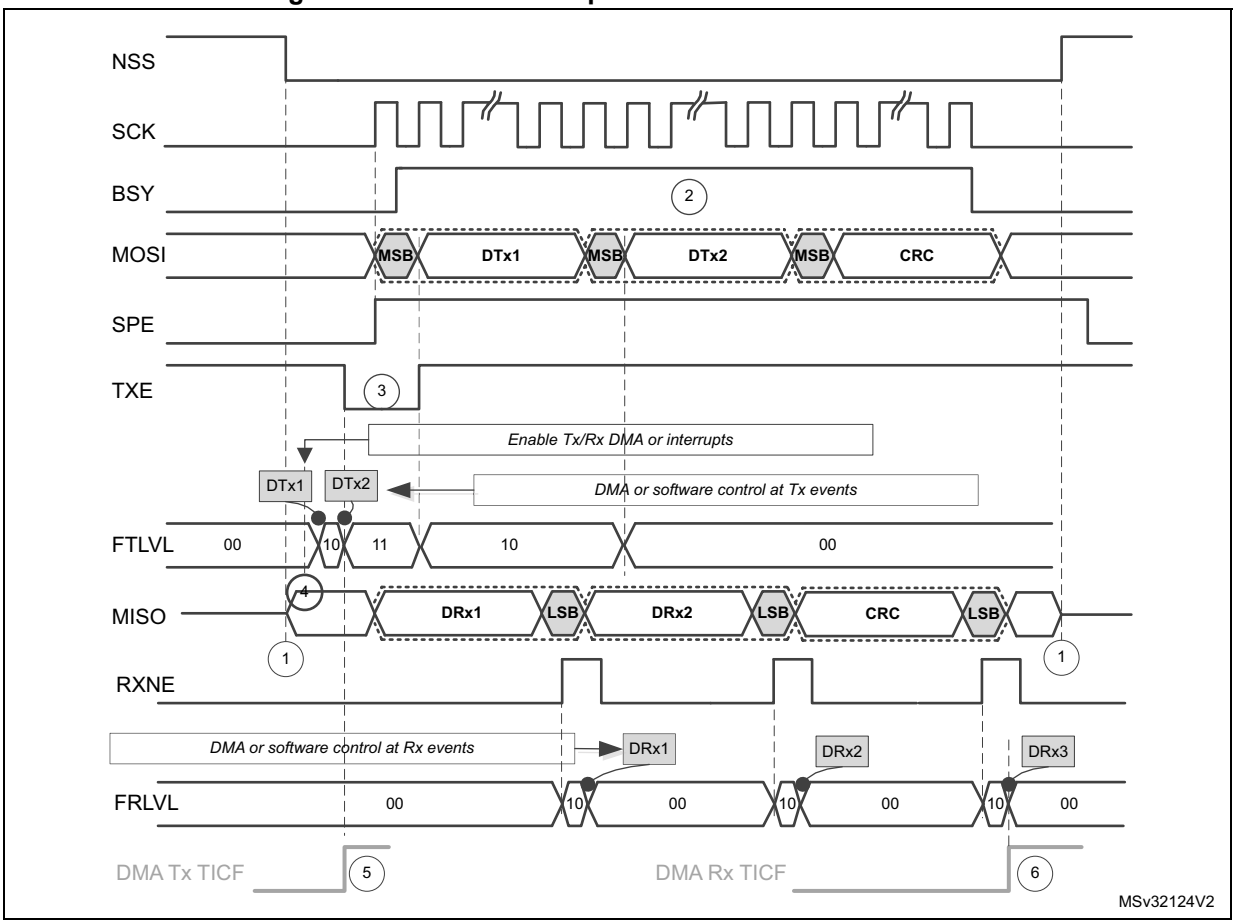
- Data size > 8 bit

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 on page 1002* for details about common assumptions and notes.

**Figure 330. Master full-duplex communication with CRC**




Assumptions for master full-duplex communication with CRC example:

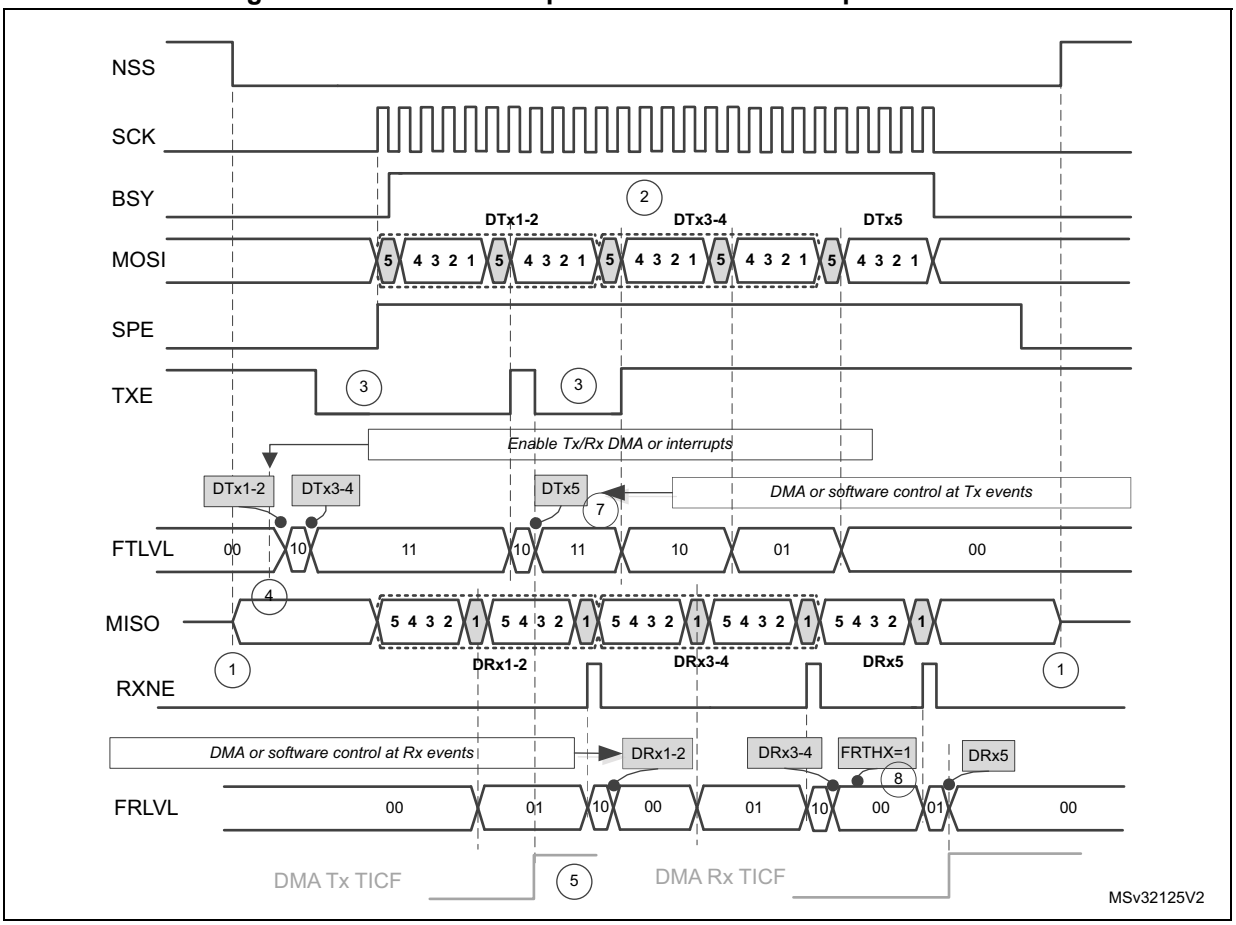
- Data size = 16 bit
- CRC enabled

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 on page 1002* for details about common assumptions and notes.

**Figure 331. 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 on page 1002* for details about common assumptions and notes.

### 29.4.10 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.*

### 29.4.11 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 29.4.14: 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.

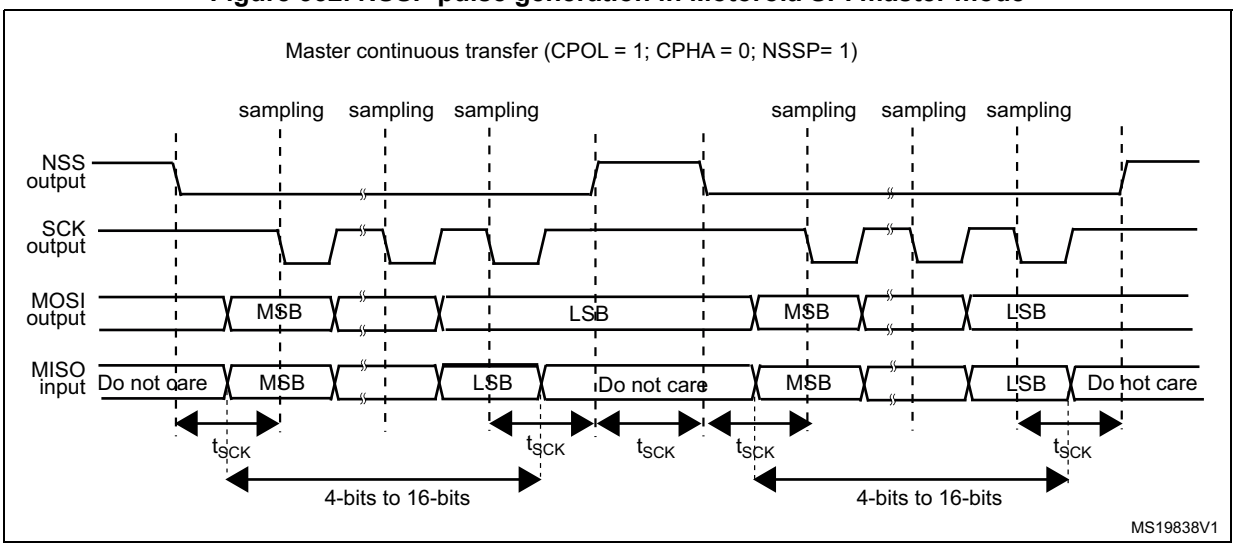
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

### 29.4.12 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 332* illustrates NSS pin management when NSSP pulse mode is enabled.

**Figure 332. 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.

### 29.4.13 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 333*). 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:

$$\frac{t_{baud_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud_rate}}{2} + 6 \times t_{pclk}$$

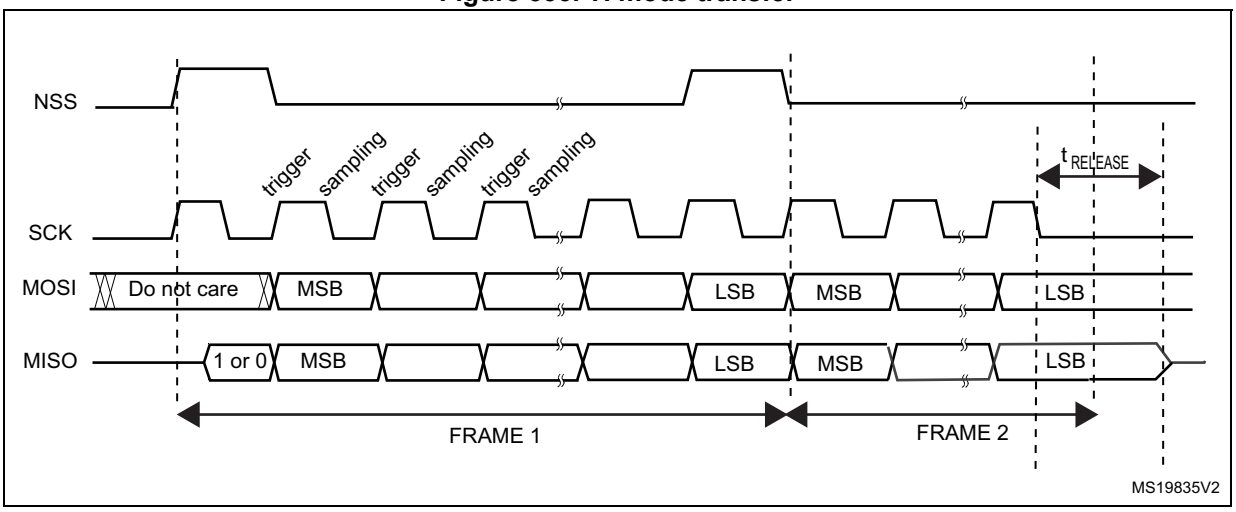
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

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 333: TI mode transfer* shows the SPI communication waveforms when TI mode is selected.

**Figure 333. TI mode transfer**




## 29.4.14 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 interface is configured as a slave, the NSS internal signal needs to be kept low during transaction of the CRC phase once the CRCNEXT signal is released. That is why the CRC calculation cannot be used at NSS Pulse mode when NSS hardware mode should be applied at slave normally.*

*At TI mode, despite the fact that clock phase and clock polarity setting is fixed and independent on SPIx_CR1 register, the corresponding setting CPOL=0 CPHA=1 has to be kept at the SPIx_CR1 register anyway if CRC is applied. In addition, the CRC calculation has to be reset between sessions by SPI disable sequence with re-enable the CRCEN bit described above at both master and slave side, else CRC calculation can be corrupted at this specific mode.*

## 29.5 SPI interrupts

During SPI communication an interrupt 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
- CRC protocol error

Interrupts can be enabled and disabled separately.

**Table 179. 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 | |
| CRC protocol error | CRCERR | |

## 29.6 SPI registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI_DR in addition can be accessed by 8-bit access.

### 29.6.1 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 | BIDIOE | CRC EN | CRCN EXT | 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:*

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.*

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.*

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.*

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.*

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)

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 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 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 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 on page 998.*

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.*

Bit 2 **MSTR:** Master selection

0: Slave configuration

1: Master configuration

*Note: This bit should not be changed when communication is ongoing.*

Bit 1 **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 SPI TI mode except the case when CRC is applied at 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 SPI TI mode except the case when CRC is applied at TI mode.*

## 29.6.2 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 on page 998 if the CRCEN bit is set.*

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 on page 998 if the CRCEN bit is set.*

Bit 12 **FRXTH**: FIFO reception threshold

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)

Bits 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)

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).

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).*

Bit 3 **NSSP**: NSS pulse management

This bit is used in master mode only. it allows 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 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 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

### 29.6.3 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[1:0] | | FRE | BSY | OVR | MODF | CRCERR | Res. | Res. | TXE | RXNE |
| | | | r | r | r | r | r | r | r | r | rc_w0 | | | 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)

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 SPI receive-only mode while CRC calculation is enabled.*

Bit 8 **FRE**: Frame format error

This flag is used for SPI in TI slave mode. Refer to *Section 29.4.11: SPI 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 not busy

1: SPI 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 29.4.10: SPI status flags and Procedure for disabling the SPI on page 998.*

Bit 6 **OVR:** Overrun flag

0: No overrun occurred

1: Overrun occurred

This flag is set by hardware and reset by a 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 *Section : Mode fault (MODF) on page 1008* 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

Note: This flag is set by hardware and cleared by software writing 0.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TXE:** Transmit buffer empty

0: Tx buffer not empty

1: Tx buffer empty

Bit 0 **RXNE:** Receive buffer not empty

0: Rx buffer empty

1: Rx buffer not empty

## 29.6.4 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 29.4.9: 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.*

### 29.6.5 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 (0x0007) 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.*

### 29.6.6 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 CRC frame format is set to be 8-bit length (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 CRC 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.*

### 29.6.7 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 CRC frame format is set to be 8-bit length (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 CRC 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.*

### 29.6.8 SPI register map

*Table 180* shows the SPI register map and reset values.

**Table 180. SPI register map and reset values**

| Offset | Register name reset value | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | SPIx_CR1 | 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. | 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 |
| 0x08 | SPIx_SR | Res. | Res. | Res. | FTLVL[1:0] | | FRLVL[1:0] | | FRE | BSY | OVR | MODF | CRCERR | | | TXE | RXNE |
| | Reset value | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | 1 | 0 |
| 0x0C | SPIx_DR | DR[15:0] | | | | | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10 | SPIx_CRCPR | CRCPOLY[15:0] | | | | | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0x14 | SPIx_RXCRCR | RXCRC[15:0] | | | | | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x18 | SPIx_TXCRCR | TXCRC[15:0] | | | | | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 30 Inter-processor communication controller (IPCC)

## 30.1 Introduction

The inter-processor communication controller (IPCC) is used for communicating data between two processors.

The IPCC block provides a nonblocking signaling mechanism to post and retrieve communication data in an atomic way. It provides the signaling for twelve channels:

- six channels in the direction from processor 1 to processor 2
- six channels in the opposite direction

It is then possible to have two different communication types in each direction.

The IPCC communication data must be located in a common memory, which is not part of the IPCC block.

## 30.2 IPCC main features

- Status signaling for the twelve channels
  - Channel occupied/free flag, also used as lock
- Two interrupt lines per processor
  - One for RX channel occupied (communication data posted by sending processor)
  - One for TX channel free (communication data retrieved by receiving processor)
- Interrupt masking per channel
  - Channel occupied mask
  - Channel free mask
- Two channel operation modes
  - Simplex (each channel has its own communication data memory location)
  - Half duplex (a single channel associated to a bidirectional communication data information memory location)

## 30.3 IPCC functional description

The IPCC communication data is located in a common memory, which is not part of the IPCC block. The address location of the communication data must be known or located in a known common area that, as already stated, is not part of the IPCC block.

For each communication, the IPCC block provides a channel status flag CHnF.

- When 0, the channel status flag CHnF indicates that the associated IPCC channel is free (the receiving processor has retrieved communication data), and can be accessed by the sending processor.
- When 1, the channel status flag CHnF indicates that the associated IPCC channel is occupied (the sending processor has posted communication data) and can be accessed by the receiving processor.

The channel operation mode must be known to both processors. A common parameter can be used to indicate the channel transfer mode and must also be located in a known common area. This parameter is not available from the IPCC.

### 30.3.1 IPCC block diagram

The IPCC (see *Figure 334*) consists of the following subblocks:

- Status block, containing the channel status
- IPCC interface block, providing AHB access to the channel status registers
- Interrupt interface block, providing control for the interrupts

**Figure 334. IPCC block diagram**

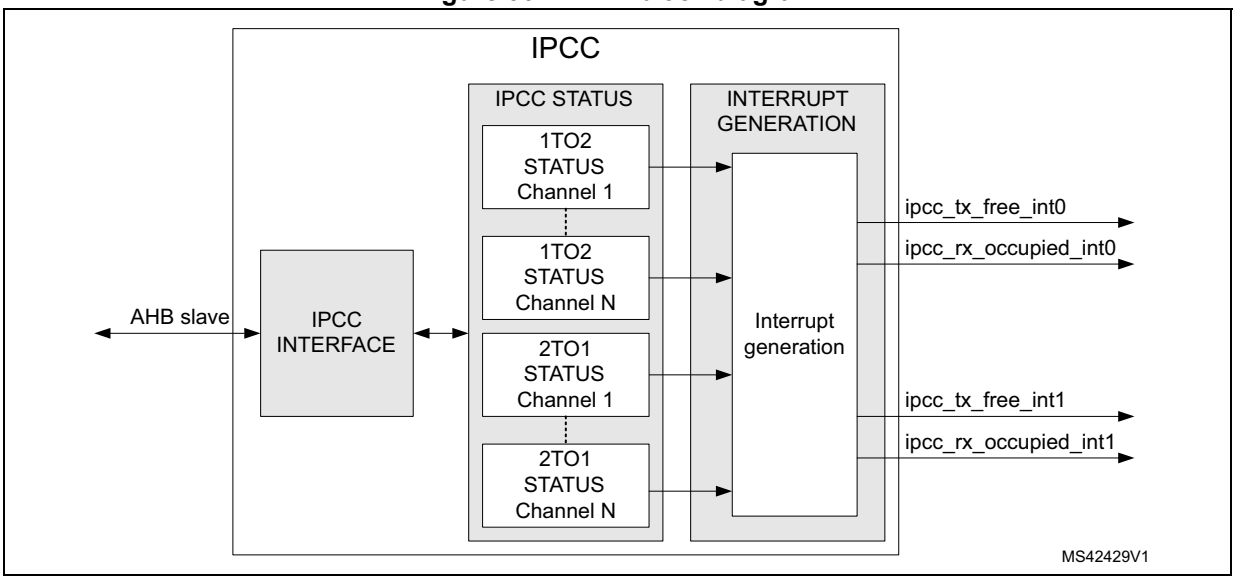



**Table 181. IPCC interface signals**

| Signal | | Description |
|---|---|---|
| Name | Type | |
| AHB slave | I/O | AHB register access bus |
| ipcc_tx_free_int1 | O | TX free interrupt to processor 1 |
| ipcc_rx_occupied_int1 | O | RX occupied interrupt to processor 1 |
| ipcc_tx_free_int2 | O | TX free interrupt to processor 2 |
| ipcc_rx_occupied_int2 | O | RX occupied interrupt to processor 2 |

### 30.3.2 IPCC Simplex channel mode

In Simplex channel mode, a dedicated memory location (used to transfer data in a single direction) is assigned to the communication data. The associated channel N control bits (see *Table 182*) are used to manage the transfer from the sending to the receiving processor.

**Table 182. Bits used for the communication**

| Processor | A | B |
|---|---|---|
| SEND A = 1<br>RECEIVE B = 2 | IPCC_C1CR.TXFIE<br>IPCC_C1MR.CHnFM<br>IPCC_C1SCR.CHnS<br>IPCC_C1TOC2SR.CHnF | IPCC_C2CR.RXOIE<br>IPCC_C2MR.CHnOM<br>IPCC_C2SCR.CHnC |
| SEND A = 2<br>RECEIVE B = 1 | IPCC_C2CR.TXFIE<br>IPCC_C2MR.CHnFM<br>IPCC_C2SCR.CHnS<br>IPCC_C2TOC1SR.CHnF | IPCC_C1CR.RXOIE<br>IPCC_C1MR.CHnOM<br>IPCC_C1SCR.CHnC |

Once the sending processor has posted the communication data in the memory, it sets the channel status flag CHnF to occupied with CHnS.

Once the receiving processor has retrieved the communication data from the memory, it clears the channel status flag CHnF back to free with CHnC.

**Figure 335. IPCC Simplex channel mode transfer timing**

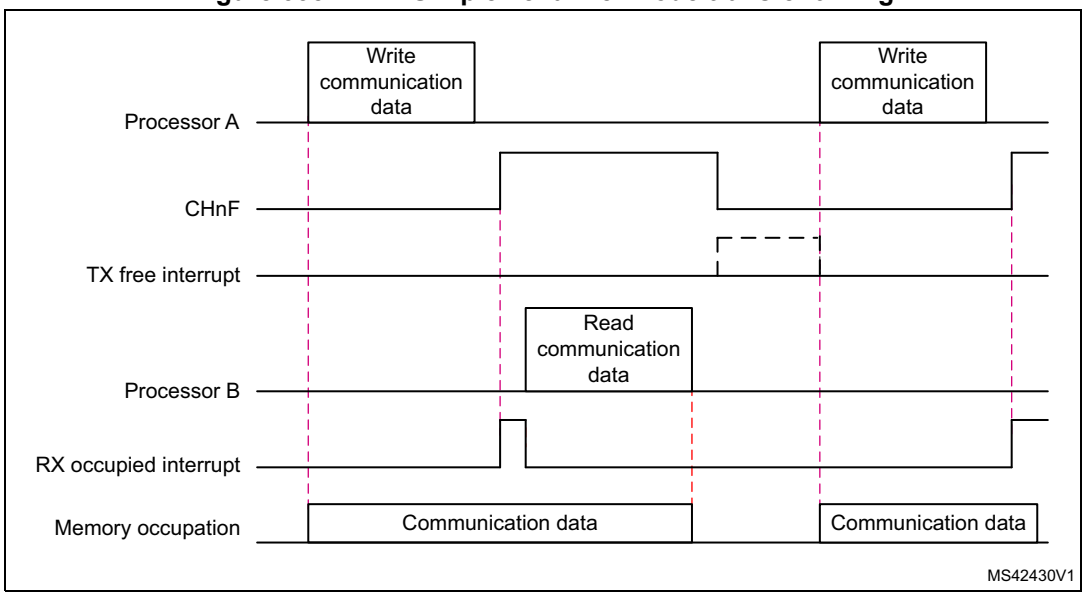
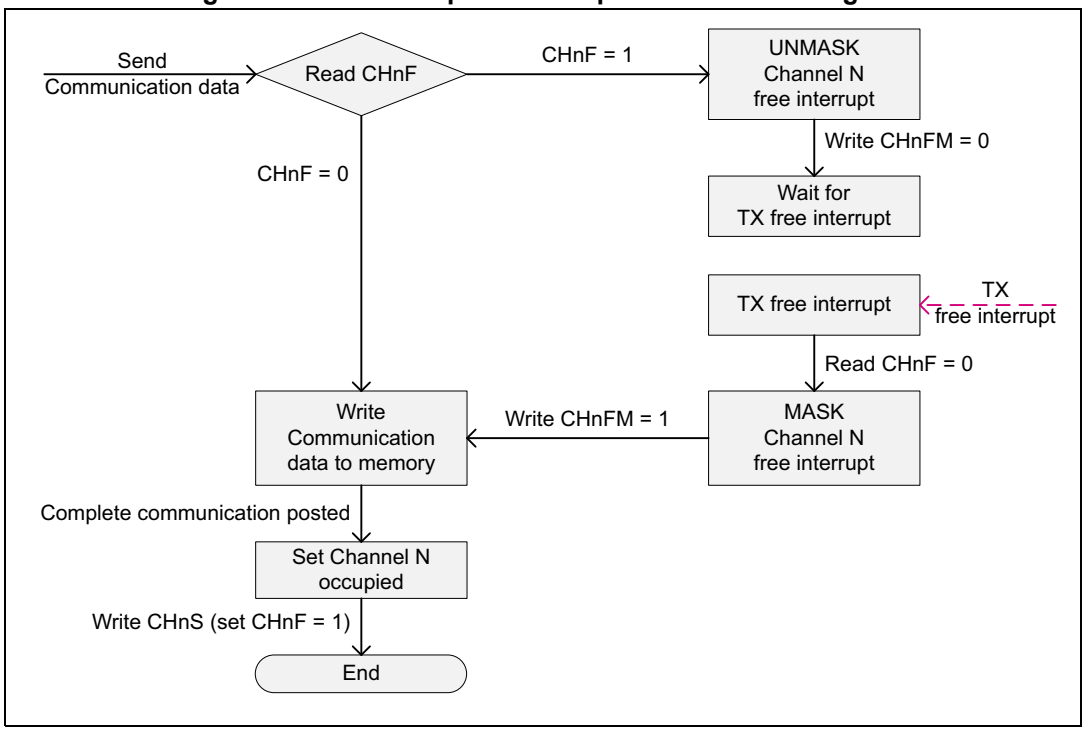


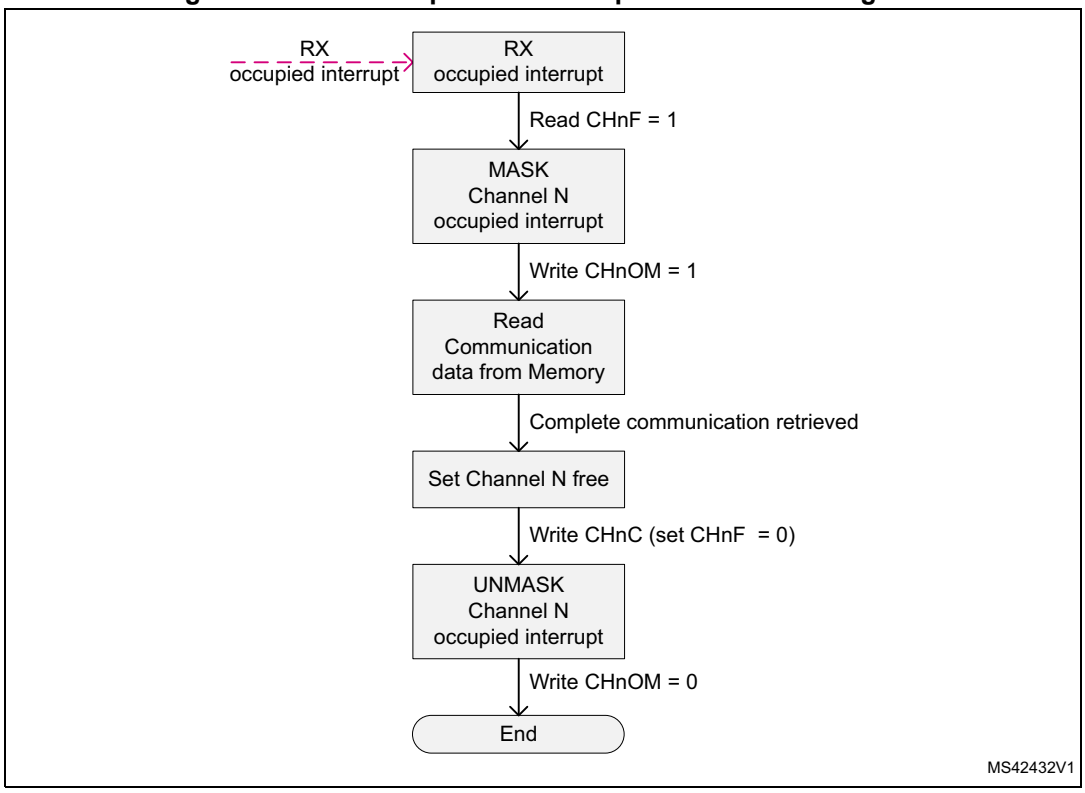
**Figure 336. IPCC Simplex - Send procedure state diagram**




To send communication data:

- The sending processor checks the channel status flag CHnF:
  - When CHnF = 0, the channel is free (last communication data retrieved by the receiving processor), and the new communication data can be written.
  - When CHnF = 1, the channel is occupied (last communication data not retrieved by the receiving processor), and the sending processor unmasks the channel free interrupt (CHnFM = 0).
  - On a TX free interrupt, the sending processor checks which channel became free and masks the channel free interrupt (CHnFM = 1). Then the new communication can take place.
- Once the complete communication data is posted, the channel status is set to occupied with CHnS. This gives memory access to the receiving processor and generates the RX occupied interrupt.

**Figure 337. IPCC Simplex - Receive procedure state diagram**




To receive a communication, the channel occupied interrupt is unmasked (CHnOM = 0):

- On an RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from memory.
- Once the complete communication data is retrieved, the channel status is cleared to free with CHnC. This gives memory access back to the sending processor and may generate the TX free interrupt.
- Once the channel status is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

### 30.3.3 IPCC Half-duplex channel mode

The Half-duplex channel mode is used when one processor sends a communication and the other processor sends a response to each communication (ping-pong).

In Half-duplex channel mode, a single dedicated memory location is assigned to communication data and response, and is used to transfer data in both directions. The sending processor channel status flag CHnF is assigned to the channel and used by both processors (see *Table 182*).

Once the processor A posts communication data into memory, it sets the processor A channel status flag CHnF to occupied with CHnS (giving memory access to processor B).

Once the processor B retrieves communication data from memory, it does not change the channel status flags. The memory access is kept by processor B for the response.

Once the processor B posts the response into memory, it clears the channel status flag CHnF to free with CHnC (giving memory access back to processor A).

Once the processor A retrieves the response from the memory, it does not change the channel status flags. The memory location access is kept by processor A for the next communication data.

**Figure 338. IPCC Half-duplex channel mode transfer timing**

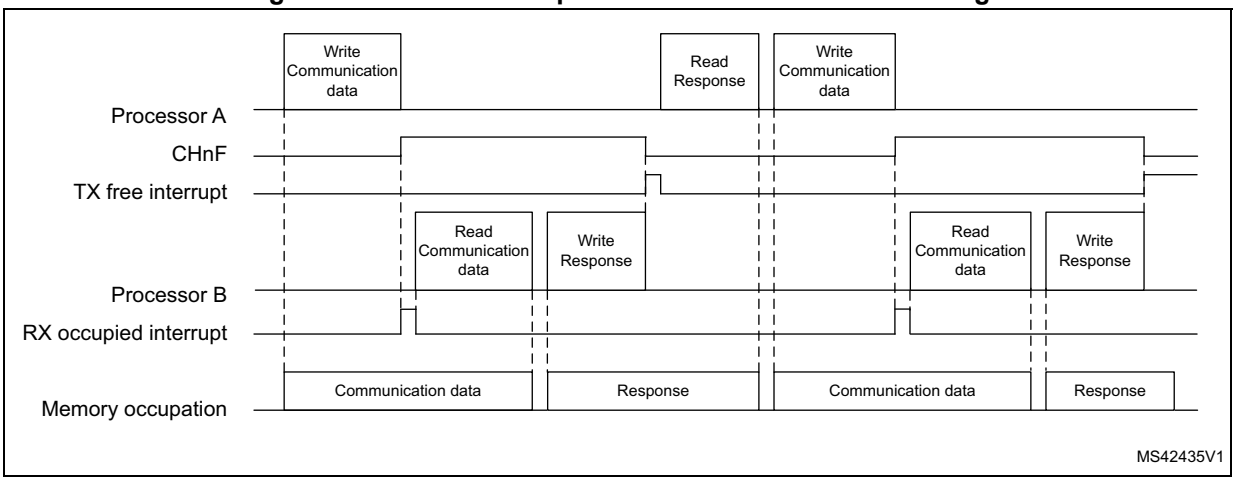
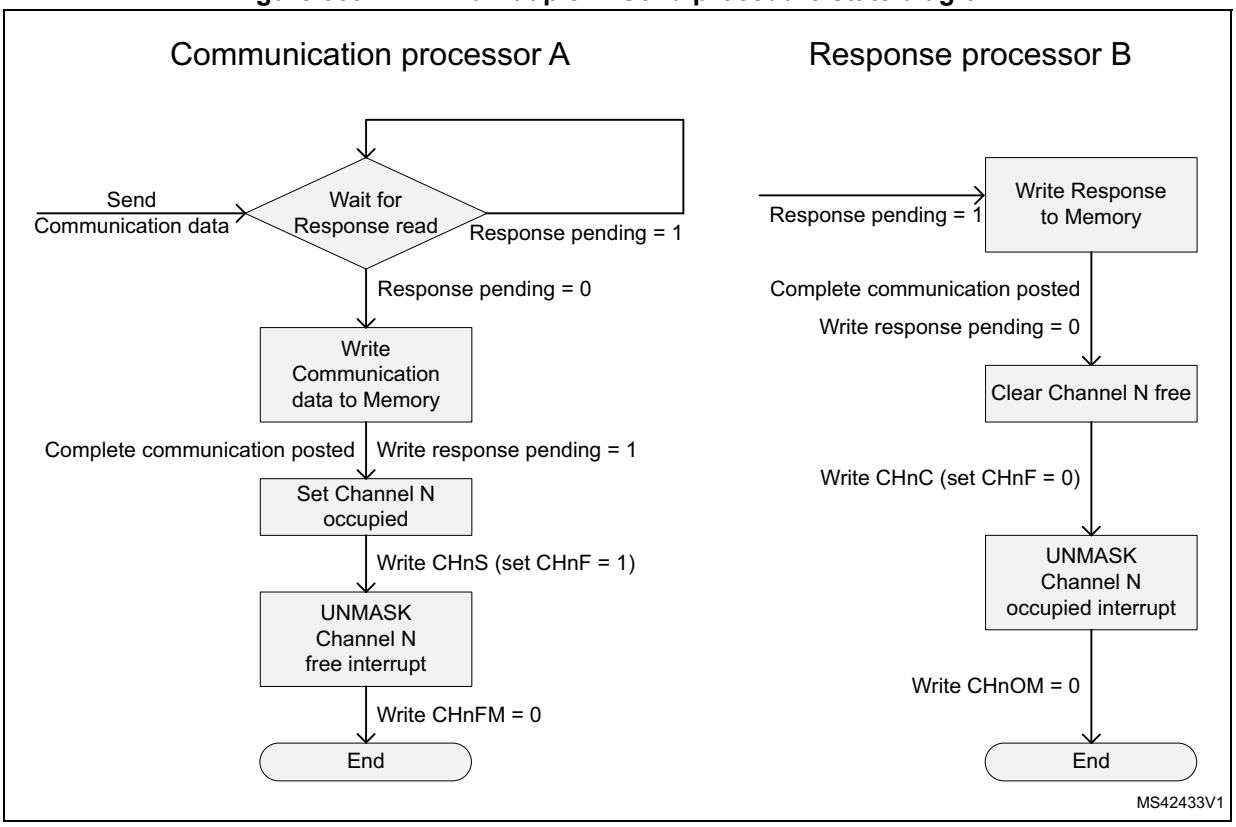



**Figure 339. IPCC Half-duplex - Send procedure state diagram**



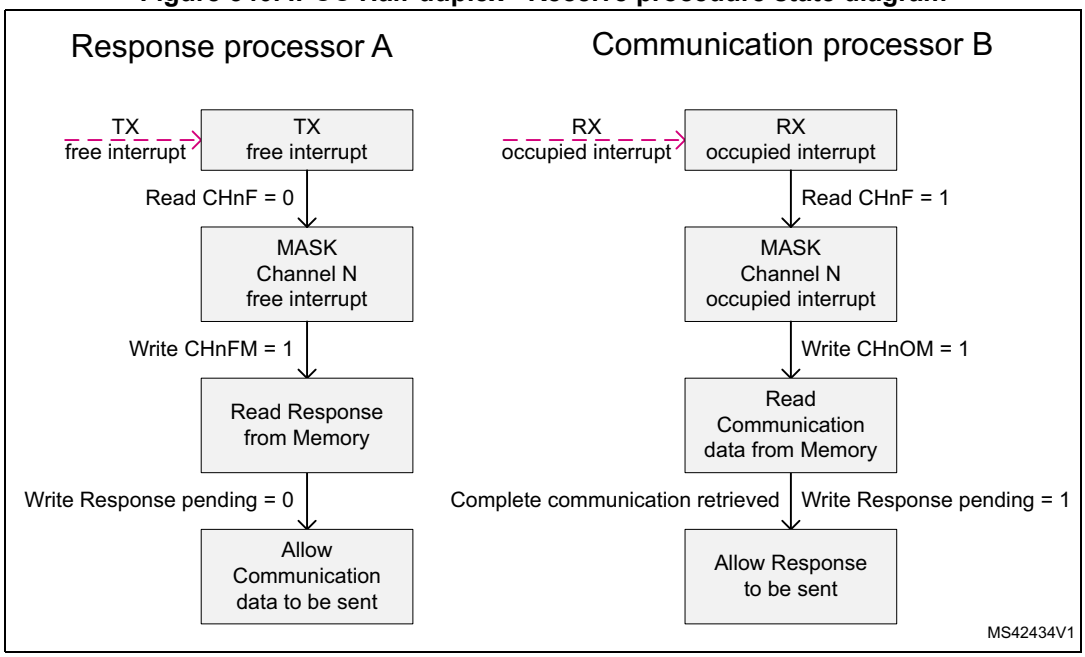
To send communication data:

- The sending processor waits for its response pending software variable to get 0.
  - Once the response pending software variable is 0 the communication data is posted.
- Once the complete communication data has been posted, the channel status flag CHnF is set to occupied with CHnS, and the response pending software variable is set to 1 (this gives memory access and generates the RX occupied interrupt to the receiving processor).
- Once the channel status flag CHnF is set, the channel free interrupt is unmasked (CHnFM = 0).

To send a response:

- The receiving processor waits for its response pending software variable to get 1.
  - Once the response pending software variable is 1, the response is posted.
- Once the complete response is posted, the channel status flag CHnF is cleared to free with CHnC, and the response pending software variable is set to 0 (this gives memory access and generates the TX free interrupt to the sending processor).
- Once the channel status flag CHnF is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

**Figure 340. IPCC Half-duplex - Receive procedure state diagram**




To receive communication data the channel occupied interrupt is unmasked (CHnOM = 0):

- On an RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from the memory.
- Once the complete communication data is retrieved, the response pending software variable is set. The channel status is not changed, access to the memory is kept to post the subsequent response.

To receive the response the channel free interrupt is unmasked (CHnFM = 0):

- On a TX free interrupt, the sending processor checks which channel became free, masks the associated channel free interrupt (CHnFM) and reads the response from the memory.
- Once the complete response is retrieved, the response pending software variable is cleared. The channel status is not changed, access to the memory is kept to post the subsequent communication data.

### 30.3.4 IPCC interrupts

There are four interrupt lines :

- two RX channel occupied interrupts, one for each processor:
  - Interrupt enable RXOIE per processor
  - Individual mask CHnOM per channel
- two TX channel free interrupts, one for each processor
  - Interrupt enable TXFIE per processor
  - Individual mask CHnFM per channel

The RX occupied interrupt is used by the receiving processor and indicates when an unmasked channel status indicates occupied (CHnF = 1).

The TX free interrupt is used by the sending processor, and indicates when an unmasked channel status indicates free (CHnF = 0).

A secure channel only generates a secure interrupt, and only in the case when the channel is secure unmasked and global secure enabled.

A non-secure channel only generates a non-secure interrupt, and only in the case when the channel is non-secure unmasked and global non-secure enabled.

## 30.4 IPCC registers

The peripheral registers must be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses are not permitted and do not generate a bus error.

### 30.4.1 IPCC processor 1 control register (IPCC_C1CR)

Address offset: 0x000

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. | TXFIE |
| | | | | | | | | | | | | | | | 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. | RXOIE |
| | | | | | | | | | | | | | | | rw |

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

Bit 16 **TXFIE**: Processor 1 transmit channel free interrupt enable
Associated with IPCC_C1TOC2SR.
1: Enable an unmasked processor 1 transmit channel free to generate a TX free interrupt.
0: Processor 1 TX free interrupt disabled

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **RXOIE**: Processor 1 receive channel occupied interrupt enable
Associated with IPCC_C2TOC1SR.
1: Enable an unmasked processor 1 receive channel occupied to generate an RX occupied interrupt.
0: Processor 1 RX occupied interrupt disabled

### 30.4.2 IPCC processor 1 mask register (IPCC_C1MR)

Address offset: 0x004

Reset value: 0xFFFF 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. | CH6 FM | CH5 FM | CH4 FM | CH3 FM | CH2 FM | CH1 FM |
| | | | | | | | | | | 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. | Res. | CH6 OM | CH5 OM | CH4 OM | CH3 OM | CH2 OM | CH1 OM |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnFM:** Processor 1 transmit channel n status set, (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Transmit channel n free interrupt masked.
0: Transmit channel n free interrupt not masked.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnOM**: Processor 1 receive channel n status clear (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Receive channel n occupied interrupt masked.
0: Receive channel n occupied interrupt not masked.

### 30.4.3 IPCC processor 1 status set clear register (IPCC_C1SCR)

Address offset: 0x008

Reset value: 0x0000 0000

Reading this register always returns 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. | CH6S | CH5S | CH4S | CH3S | CH2S | CH1S |
| | | | | | | | | | | 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. | Res. | CH6C | CH5C | CH4C | CH3C | CH2C | CH1C |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnS:** Processor 1 transmit channel n status set (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Processor 1 transmit channel n status bit set.
0: No action.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnC**: Processor 1 receive channel n status clear (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Processor 1 receive channel n status bit clear.
0: No action.

### 30.4.4 IPCC processor 1 to processor 2 status register (IPCC_C1TOC2SR)

Address offset: 0x00C

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. | CH6F | CH5F | CH4F | CH3F | CH2F | CH1F |
| | | | | | | | | | | r | r | r | r | r | r |

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnF:** Processor 1 transmit to processor 2 receive channel n status flag before masking (n = 6 to 1).

1: Channel occupied, data can be read by the receiving processor 2.
Generates a channel RX occupied interrupt to processor 2, when unmasked.
0: Channel free, data can be written by the sending processor 1.
Generates a channel TX free interrupt to processor 1, when unmasked.

### 30.4.5 IPCC processor 2 control register (IPCC_C2CR)

Address offset: 0x010

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. | TXFIE |
| | | | | | | | | | | | | | | | 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. | RXOIE |
| | | | | | | | | | | | | | | | rw |

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

Bit 16 **TXFIE:** Processor 2 transmit channel free interrupt enable

Associated with IPCC_C2TOC1SR.
1: Enable an unmasked processor 2 transmit channel free to generate a TX free interrupt.
0: Processor 2 TX free interrupt disabled

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **RXOIE:** Processor 2 receive channel occupied interrupt enable

Associated with IPCC_C1TOC2SR.
1: Enable an unmasked processor 2 receive channel occupied to generate an RX occupied interrupt.
0: Processor 2 RX occupied interrupt disabled

### 30.4.6 IPCC processor 2 mask register (IPCC_C2MR)

Address offset: 0x014

Reset value: 0xFFFF 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. | CH6 FM | CH5 FM | CH4 FM | CH3 FM | CH2 FM | CH1 FM |
| | | | | | | | | | | 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. | Res. | CH6 OM | CH5 OM | CH4 OM | CH3 OM | CH2 OM | CH1 OM |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnFM:** Processor 2 transmit channel n free interrupt mask (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Transmit channel n free interrupt masked.
0: Transmit channel n free interrupt not masked.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnOM**: Processor 2 receive channel n occupied interrupt mask (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Receive channel n occupied interrupt masked.
0: Receive channel n occupied interrupt not masked.

### 30.4.7 IPCC processor 2 status set clear register (IPCC_C2SCR)

Address offset: 0x018

Reset value: 0x0000 0000

Reading this register always returns 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. | CH6S | CH5S | CH4S | CH3S | CH2S | CH1S |
| | | | | | | | | | | 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. | Res. | CH6C | CH5C | CH4C | CH3C | CH2C | CH1C |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnS:** Processor 2 transmit channel n status set (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Processor 2 transmit channel n status bit set.
0: No action.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnC**: Processor 2 receive channel n status clear (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Processor 2 receive channel n status bit clear.
0: No action.

### 30.4.8 IPCC processor 2 to processor 1 status register (IPCC_C2TOC1SR)

Address offset: 0x01C

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. | CH6F | CH5F | CH4F | CH3F | CH2F | CH1F |
| | | | | | | | | | | r | r | r | r | r | r |

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnF:** Processor 2 transmit to processor 1 receive channel n status flag before masking (n = 6 to 1)
1: Channel occupied, data can be read by the receiving processor 1.
Generates a channel RX occupied interrupt to processor 1, when unmasked.
0: Channel free, data can be written by the sending processor 2.
Generates a channel TX free interrupt to processor 2, when unmasked.

### 30.4.9 IPCC register map

**Table 183. IPCC 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | IPCC_C1CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TXFIE | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | RXOIE |
| | Reset value | | | | | | | | | | | | | | | | 0 | | | | | | | | | | | | | | | | 0 |
| 0x0004 | IPCC_C1MR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6FM | CH5FM | CH4FM | CH3FM | CH2FM | CH1FM | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6OM | CH5OM | CH4OM | CH3OM | CH2OM | CH1OM |
| | Reset value | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 |
| 0x0008 | IPCC_C1SCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6S | CH5S | CH4S | CH3S | CH2S | CH1S | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6C | CH5C | CH4C | CH3C | CH2C | CH1C |
| | Reset value | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000C | IPCC_C1TOC2SR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6F | CH5F | CH4F | CH3F | CH2F | CH1F |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0010 | IPCC_C2CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TXFIE | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | RXOIE |
| | Reset value | | | | | | | | | | | | | | | | 0 | | | | | | | | | | | | | | | | 0 |
| 0x0014 | IPCC_C2MR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6FM | CH5FM | CH4FM | CH3FM | CH2FM | CH1FM | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6OM | CH5OM | CH4OM | CH3OM | CH2OM | CH1OM |
| | Reset value | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 |
| 0x0018 | IPCC_C2SCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6S | CH5S | CH4S | CH3S | CH2S | CH1S | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6C | CH5C | CH4C | CH3C | CH2C | CH1C |
| | Reset value | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x001C | IPCC_C2TOC1SR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CH6F | CH5F | CH4F | CH3F | CH2F | CH1F |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 31 Hardware semaphore (HSEM)

## 31.1 Introduction

The hardware semaphore block provides 32 (32-bit) register based semaphores.

The semaphores can be used to ensure synchronization between different processes running between different cores. The HSEM provides a non-blocking mechanism to lock semaphores in an atomic way. The following functions are provided:

- Semaphore lock, in two ways:
  - 2-step lock: by writing COREID and PROCID to the semaphore, followed by a read check
  - 1-step lock: by reading the COREID from the semaphore
- Interrupt generation when a semaphore is unlocked
  - Each semaphore may generate an interrupt on one of the interrupt lines
- Semaphore clear protection
  - A semaphore is only unlocked when COREID and PROCID match
- Global semaphore clear per COREID

## 31.2 Main features

The HSEM includes the following features:

- 32 (32-bit) semaphores
- 8-bit PROCID
- 4-bit COREID
- One interrupt line per processor
- Lock indication

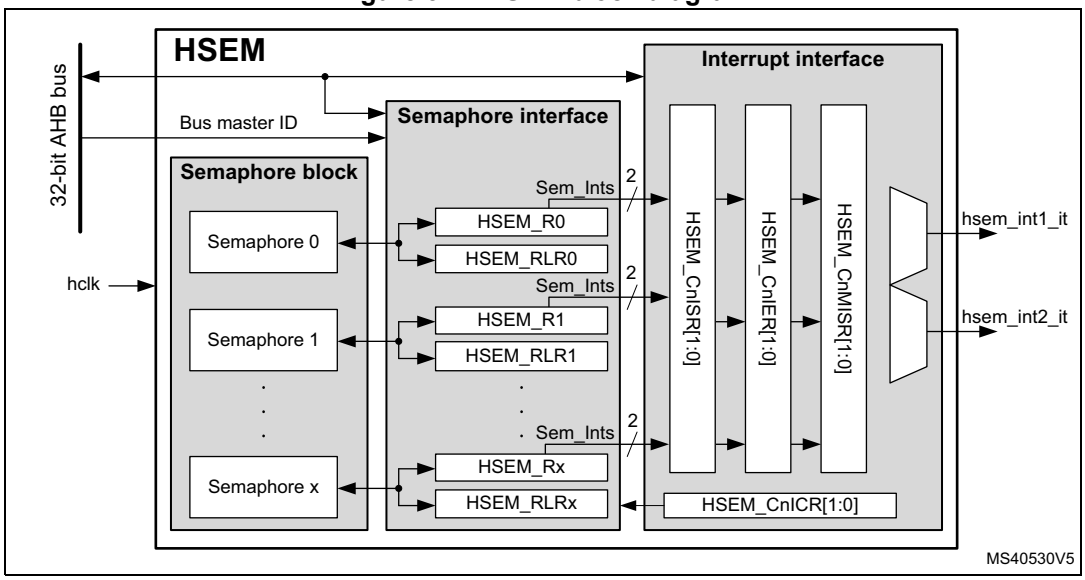
## 31.3 Functional description

### 31.3.1 HSEM block diagram

As shown in *Figure 341*, the HSEM is based on three sub-blocks:

- the semaphore block containing the semaphore status and IDs
- the semaphore interface block providing AHB access to the semaphore via the HSEM_Rx and HSEM_RLRx registers
- the interrupt interface block providing control for the interrupts via HSEM_CnISR, HSEM_CnIER, HSEM_CnMISR, and HSEM_CnICR registers.

**Figure 341. HSEM block diagram**




### 31.3.2 HSEM internal signals

**Table 184. HSEM internal input/output signals**

| Signal name | Signal type | Description |
|---|---|---|
| AHB bus | Digital input/output | AHB register access bus |
| BusMasterID | Digital input | AHB bus master ID |
| hsem_intn_it | Digital output | Interrupt n line (n = 1 to 2) |

### 31.3.3 HSEM lock procedures

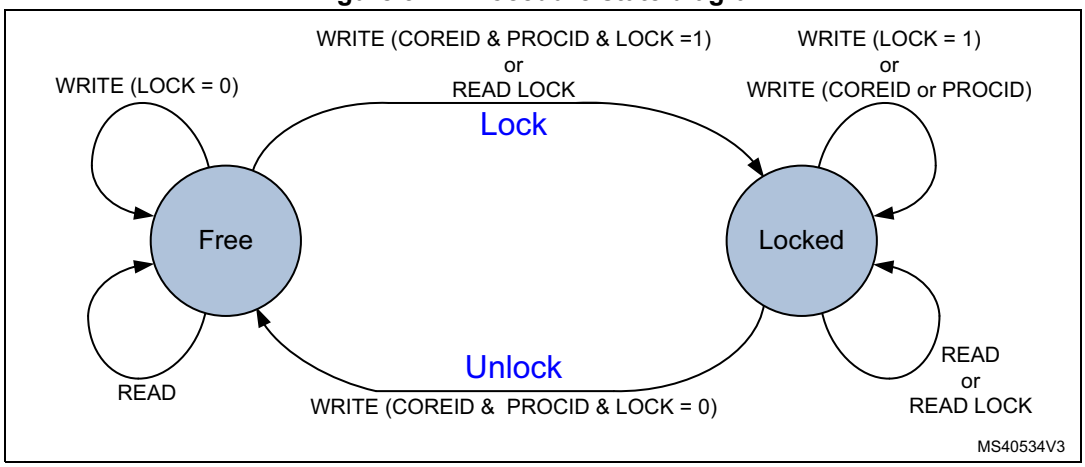
There are two lock procedures, namely 2-step (write) lock and 1-step (read) lock. The two procedures can be used concurrently.

The semaphore is free when its LOCK bit is 0. In this case, the COREID and PROCID are also 0. When the LOCK bit is 1, the semaphore is locked and the COREID indicates which AHB bus master ID has locked it. The PROCID indicates which process of that AHB bus master ID has locked the semaphore.

When write locking a semaphore, the written COREID must match the AHB bus master ID , and the PROCID is written by the AHB bus master software process taking the lock.

When read locking the semaphore, the COREID is taken from the AHB bus master ID, and the PROCID is forced to 0 by hardware. There is no PROCID available with read lock.

**Figure 342. Procedure state diagram**




## 2-step (write) lock procedure

The 2-step lock procedure consists in a write to lock the semaphore, followed by a read to check if the lock has been successful, carried out from the HSEM_Rx register.

- Write semaphore with PROCID and COREID, and LOCK = 1. The COREID data written by software must match the AHB bus master information, that is, a AHB bus master ID = 1writes data COREID = 1.

  Lock is put in place when the semaphore is free at write time.
- Read-back the semaphore

  The software checks the lock status, if PROCID and COREID match the written data, then the lock is confirmed.
- Else retry (the semaphore has been locked by another process, AHB bus master ID).

A semaphore can only be locked when it is free.

A semaphore can be locked when the PROCID = 0.

Consecutive write attempts with LOCK = 1 to a locked semaphore are ignored.

### 1-step (read) lock procedure

The 1-step procedure consists in a read to lock and check the semaphore in a single step, carried out from the HSEM_RLRx register.

- Read lock semaphore with the AHB bus master COREID.
- If read COREID matches and PROCID = 0, then lock is put in place. If COREID matches and PROCID is not 0, this means that another process from the same COREID has locked the semaphore with a 2-step (write) procedure.
- Else retry (the semaphore has been locked by another process, AHB bus master ID).

A semaphore can only be locked when it is free. When read locking a free semaphore, PROCID is 0. Read locking a locked semaphore returns the COREID and PROCID that locked it. All read locks, including the first one that locks the semaphore, return the COREID that locks or locked the semaphore.

*Note: The 1-step procedure must not be used when running multiple processes of the same AHB bus master ID. All processes using the same semaphore read the same status. When only one process locks the semaphore, each process of that AHB bus master ID reads the semaphore as locked by itself with the COREID.*

## 31.3.4 HSEM write/read/read lock register address

For each semaphore, two AHB register addresses are provided, separated in two banks of 32-bit semaphore registers, spaced by a 0x80 address offset.

In the first register address bank the semaphore can be written (locked/unlocked) and read through the HSEM_Rx registers.

In the second register address bank the semaphore can be read (locked) through the HSEM_RLRx registers.

## 31.3.5 HSEM unlock procedures

Unlocking a semaphore is a protected process, to prevent accidental clearing by a AHB bus master ID or by a process not having the semaphore lock right. The procedure consists in writing to the semaphore HSEM_Rx register with the corresponding COREID and PROCID and LOCK = 0. When unlocked the semaphore, the COREID, and the PROCID are all 0.

When unlocked, an interrupt may be generated to signal the event. To this end, the semaphore interrupt must be enabled.

The unlock procedure consists in a write to the semaphore HSEM_Rx register with matching COREID regardless on how the semaphore has been locked (1- or 2-step).

- Write semaphore with PROCID, COREID, and LOCK = 0
- If the written data matches the semaphore PROCID and COREID and the AHB bus master ID , the semaphore is unlocked and an interrupt may be generated when enabled, else write is ignored, semaphore remains locked and no interrupt is generated (the semaphore is locked by another process, AHB bus master ID or the written data does not match the AHB bus master signaling).

*Note: Different processes of the same AHB bus master ID can write any PROCID value. Preventing other processes of the same AHB bus master ID from unlocking a semaphore must be ensured by software, handling the PROCID correctly.*

### 31.3.6 HSEM COREID semaphore clear

All semaphores locked by a COREID can be unlocked at once by using the HSEM_CR register. Write COREID and correct KEY value in HSEM_CR. All locked semaphores with a matching COREID are unlocked, and may generate an interrupt when enabled.

*Note: This procedure may be used in case of an incorrect functioning AHB bus master ID, where another AHB bus master can unlock the locked semaphores by writing the COREID of the incorrect functioning processor into the HSEM_CR register with the correct KEY value. This unlocks all locked semaphores with a matching COREID.*

An interrupt may be generated for the unlocked semaphore(s). To this end, the semaphore interrupt must be enabled in the HSEM_CnIER registers.

### 31.3.7 HSEM interrupts

An interrupt line hsem_intn_it per processor allows each semaphore to generate an interrupt.

An interrupt line provides the following features per semaphore:

- interrupt enable
- interrupt clear
- interrupt status
- masked interrupt status

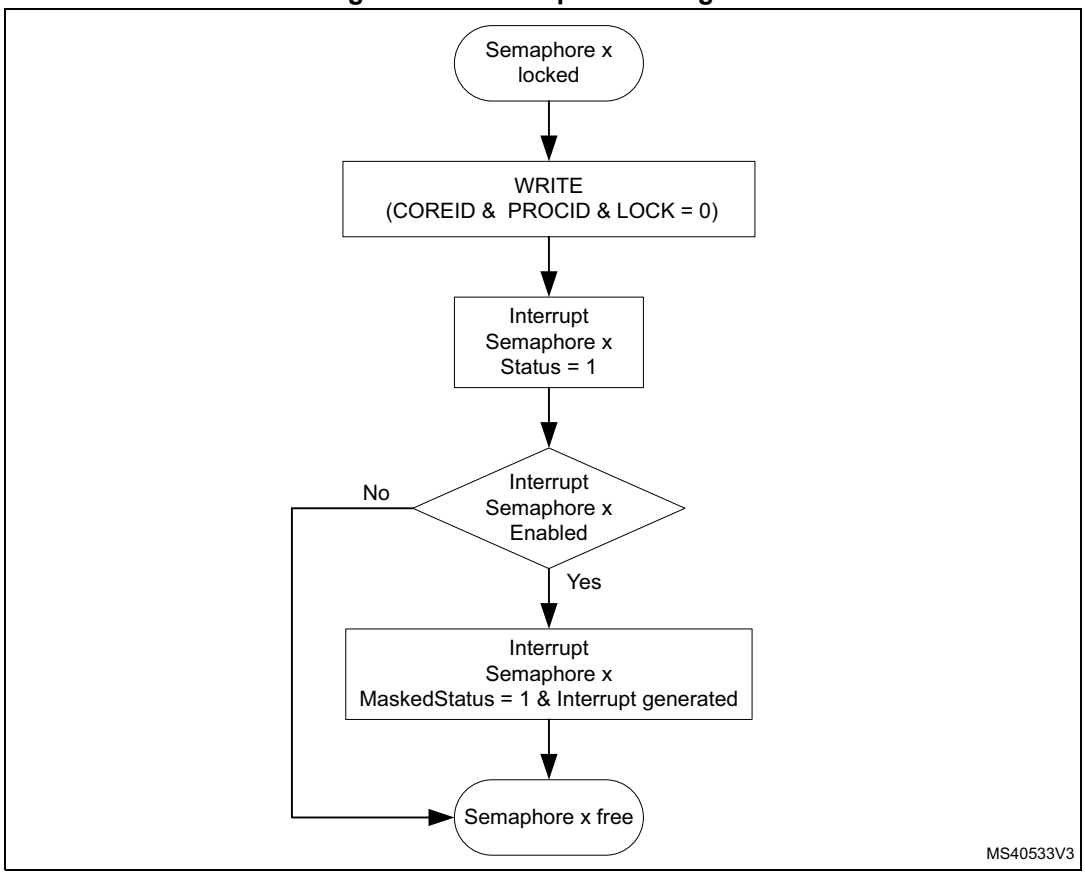
With the interrupt enable (HSEM_CnIER) the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts do not set the masked interrupt status MISF for that semaphore, and do not generate an interrupt on the interrupt line.

The interrupt clear (HSEM_CnICR) clears the interrupt status ISF and masked interrupt status MISF of the associated semaphore for the interrupt line.

The interrupt status (HSEM_CnISR) mirrors the semaphore interrupt status ISF before the enable.

The masked interrupt status (HSEM_CnMISR) only mirrors the semaphore enabled interrupt status MISF on the interrupt line. All masked interrupt status MISF of the enabled semaphores need to be cleared to clear the interrupt line.

**Figure 343. Interrupt state diagram**




The procedure to get an interrupt when a semaphore becomes free is described hereafter.

### Try to lock semaphore x

- If the semaphore lock is obtained, no interrupt is needed.
- If the semaphore lock fails:
- Clear pending semaphore x interrupt status for the interrupt line in HSEM_CnICR.
  Re-try to lock the semaphore x again:
  - If the semaphore lock is obtained, no interrupt is needed (semaphore has been freed between first try to lock and clear semaphore interrupt status).
  - If the semaphore lock fails, enable the semaphore x interrupt in HSEM_CnIER.

### On semaphore x free interrupt, try to lock semaphore x

- If the semaphore lock is obtained:
  Disable the semaphore x interrupt in HSEM_CnIER.
  Clear pending semaphore x interrupt status in HSEM_CnICR.
- If the semaphore x lock fails:
  Clear pending semaphore x interrupt status in HSEM_CnICR.
  Try again to lock the semaphore x:
  - If the semaphore lock is obtained (semaphore has been freed between first try to lock and semaphore interrupt status clear), disable the semaphore interrupt in HSEM_CnIER.

– If the semaphore lock fails, wait for semaphore free interrupt.

*Note: An interrupt does not lock the semaphore. After an interrupt, either the AHB bus master or the process must still perform the lock procedure to lock the semaphore.*

It is possible to have multiple AHB bus masters informed by the semaphore free interrupts. Each AHB bus master gets its interrupt, and the first one to react locks the semaphore.

### 31.3.8 AHB bus master ID verification

The HSEM allows only authorized AHB bus master IDs to lock and unlock semaphores.

- The AHB bus master 2-step lock write access to the semaphore HSEM_Rx register is checked against the valid bus master IDs.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not lock the semaphore.
- The AHB bus master 1-step lock read access from the semaphore HSEM_RLRx register is checked against the valid bus master IDs.
  - An unauthorized AHB bus master ID read from HSEM_RLRx returns all 0.
- The semaphore unlock write access to the HSEM_CR register is checked against the valid bus master IDs. Only the valid bus master IDs can write to the HSEM_CR register and unlock any of the COREID semaphores.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not clear the COREID semaphores.

*Table 185* details the relation between bus master/processor and COREID.

**Table 185. Authorized AHB bus master IDs**

| Bus master 0 (processor1) | Bus master 1 (processor2) |
|---|---|
| COREID = 4 | COREID = 8 |

*Note: Accesses from unauthorized AHB bus master IDs to other registers are granted.*

## 31.4 HSEM registers

Registers must be accessed using word format. Byte and half-word accesses are ignored and have no effect on the semaphores, they generate a bus error.

### 31.4.1 HSEM register semaphore x (HSEM_Rx)

Address offset: 0x000 + 0x4 * x (x = 0 to 31)

Reset value: 0x0000 0000

The HSEM_Rx must be used to perform a 2-step write lock, read back, and for unlocking a semaphore. Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCK | 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. | COREID[3:0] | | | | PROCID[7:0] | | | | | | | |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **LOCK:** Lock indication

This bit can be written and read by software.

0: On write free semaphore (only when COREID and PROCID match), on read semaphore is free.

1: On write try to lock semaphore, on read semaphore is locked.

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]:** Semaphore COREID

Written by software

- When the semaphore is free and the LOCK bit is at the same time written to 1 and the COREID matches the AHB bus master ID.
- When the semaphore is unlocked (LOCK written to 0 and AHB bus master ID matched COREID, the COREID is cleared to 0.
- When the semaphore is unlocked (LOCK bit written to 0 or AHB bus master ID does not match COREID, the COREID is not affected.
- Write when LOCK bit is already 1 (semaphore locked), the COREID is not affected.
- An authorized read returns the stored COREID value.

Bits 7:0 **PROCID[7:0]:** Semaphore PROCID

Written by software

-When the semaphore is free and the LOCK is written to 1, and the COREID matches the AHB bus master ID, PROCID is set to the written data.

- When the semaphore is unlocked, LOCK written to 0 and AHB bus master ID matched COREID, the PROCID is cleared to 0.
- When the semaphore is unlocked, LOCK bit written to 0 and AHB bus master ID does not match COREID, the PROCID is not affected.
- Write when LOCK bit is already 1 (semaphore locked), the PROCID is not affected.
- An authorized read returns the stored PROCID value.

### 31.4.2 HSEM read lock register semaphore x (HSEM_RLRx)

Address offset: 0x080 + 0x4 * x (x = 0 to 31)

Reset value: 0x0000 0000

Accesses the same physical bits as HSEM_Rx. The HSEM_RLRx must be used to perform a 1-step read lock. Only read accesses with authorized AHB bus master IDs are granted. Read accesses with unauthorized AHB bus master IDs are discarded and return 0.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCK | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | COREID[3:0] | | | | PROCID[7:0] | | | | | | | |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31 **LOCK:** Lock indication

This bit is read only by software at this address.

- When the semaphore is free:

A read with a valid AHB bus master ID locks the semaphore and returns 1.

- When the semaphore is locked:

A read with a valid AHB bus master ID returns 1 (the COREID and PROCID reflect the already locked semaphore information).

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]:** Semaphore COREID

This field is read only by software at this address.

On a read, when the semaphore is free, the hardware sets the COREID to the AHB bus master ID reading the semaphore. The COREID of the AHB bus master locking the semaphore is read.

On a read when the semaphore is locked, this field returns the COREID of the AHB bus master that has locked the semaphore.

Bits 7:0 **PROCID[7:0]:** Semaphore processor ID

This field is read only by software at this address.

- On a read when the semaphore is free:

A read with a valid AHB bus master ID locks the semaphore and hardware sets the PROCID to 0.

- When the semaphore is locked:

A read with a valid AHB bus master ID returns the PROCID of the AHB bus master that has locked the semaphore.

### 31.4.3 HSEM interrupt enable register (HSEM_CnIER)

Address offset: 0x100 + 0x010 * (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISE[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 |
| ISE[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **ISE[31:0]:** Interrupt(n) semaphore x enable bit (x = 0 to 31)
This bit is read and written by software.
0: Interrupt(n) generation for semaphore x disabled (masked)
1: Interrupt(n) generation for semaphore x enabled (not masked)

### 31.4.4 HSEM interrupt clear register (HSEM_CnICR)

Address offset: 0x104 + 0x010 * (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISC[31:16] | | | | | | | | | | | | | | | |
| 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 |
| ISC[15:0] | | | | | | | | | | | | | | | |
| 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 31:0 **ISC[31:0]:** Interrupt(n) semaphore x clear bit (x = 0 to 31)
This bit is written by software, and is always read 0.
0: Interrupt(n) semaphore x status ISFx and masked status MISFx not affected.
1: Interrupt(n) semaphore x status ISFx and masked status MISFx cleared.

### 31.4.5 HSEM interrupt status register (HSEM_CnISR)

Address offset: 0x108 + 0x010 * (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISF[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 |
| ISF[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **ISF[31:0]:** Interrupt semaphore x status bit before enable (mask) (x = 0 to 31)
This bit is set by hardware, and reset only by software. This bit is cleared by software writing the corresponding HSEM_CnICR bit.
0: Interrupt semaphore x status, no interrupt pending
1: Interrupt semaphore x status, interrupt pending

### 31.4.6 HSEM interrupt status register (HSEM_CnMISR)

Address offset: 0x10C + 0x010 * (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MISF[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 |
| MISF[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **MISF[31:0]:** Masked interrupt(n) semaphore x status bit after enable (mask) (x = 0 to 31)
This bit is set by hardware and read only by software. This bit is cleared by software writing the corresponding HSEM_CnICR bit. This bit is read as 0 when semaphore x status is masked in HSEM_CnIER bit x.
0: interrupt(n) semaphore x status after masking not pending
1: interrupt(n) semaphore x status after masking pending

### 31.4.7 HSEM clear register (HSEM_CR)

Address offset: 0x140

Reset value: 0x0000 0000

Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| 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 |
| Res. | Res. | Res. | Res. | COREID[3:0] | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | w | w | w | w | | | | | | | | |

Bits 31:16 **KEY[15:0]:** Semaphore clear key
This field can be written by software and is always read 0.
If this key value does not match HSEM_KEYR.KEY, semaphores are not affected.
If this key value matches HSEM_KEYR.KEY, all semaphores matching the COREID are cleared to the free state.

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

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]:** COREID of semaphores to be cleared
This field can be written by software and is always read 0.
This field indicates the COREID for which the semaphores are cleared when writing the HSEM_CR.

Bits 7:0 Reserved, must be kept at reset value.

### 31.4.8 HSEM clear semaphore key register (HSEM_KEYR)

Address offset: 0x144

Reset value: 0x0000 0000

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

Bits 31:16 **KEY[15:0]:** Semaphore clear key
This field can be written and read by software.
Key value to match when clearing semaphores.

Bits 15:0 Reserved, must be kept at reset value.

### 31.4.9 HSEM register map

**Table 186. HSEM register map and reset values**

| Offset | Register name | 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 | **HSEM_R0** | LOCK | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | COREID [3:0] | | | | PROCID[7:0] | | | | | | | |
| | Reset value | 0 | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x004 | **HSEM_R1** | LOCK | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | COREID [3:0] | | | | PROCID[7:0] | | | | | | | |
| | Reset value | 0 | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ⋮ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x07C | **HSEM_R31** | LOCK | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | COREID [3:0] | | | | PROCID[7:0] | | | | | | | |
| | Reset value | 0 | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x080 | **HSEM_RLR0** | LOCK | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | COREID [3:0] | | | | PROCID | | | | | | | |
| | Reset value | 0 | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x084 | **HSEM_RLR1** | LOCK | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | COREID [3:0] | | | | PROCID[7:0] | | | | | | | |
| | Reset value | 0 | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ⋮ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0FC | **HSEM_RLR31** | LOCK | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | COREID [3:0] | | | | PROCID[7:0] | | | | | | | |
| | Reset value | 0 | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100 | **HSEM_C1IER** | ISE[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 |
| 0x104 | **HSEM_C1ICR** | ISC[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 |
| 0x108 | **HSEM_C1ISR** | ISF[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 |
| 0x10C | **HSEM_C1MISR** | MISF[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 |
| 0x110 | **HSEM_C2IER** | ISE[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 |
| 0x114 | **HSEM_C2ICR** | ISC[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 |
| 0x118 | **HSEM_C2ISR** | ISF[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 186. HSEM register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x11C | **HSEM_C2MISR** | MISF[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 |
| 0x140 | **HSEM_CR** | KEY[15:0] | | | | | | | | | | | | | | | | Res. | Res. | Res. | Res. | COREID[3:0] | | | | Res. | Res. | Res. | Res. | 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 | 0 | | | | | | | | |
| 0x144 | **HSEM_KEYR** | KEY[15:0] | | | | | | | | | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | |

Refer to *Section 2.2 on page 56* for the register boundary addresses.

# 32 Debug support (DBG)

## 32.1 Introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Independent breakpoint debugging of each CPU core in the system
- Code execution tracing
- Software instrumentation
- Cross-triggering
- The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm® CoreSight™ components.

- General features:
  - SWJ-DP: JTAG/Serial-wire debug port
  - AHB-AP: AHB access port
- CPU2 debug features:
  - ROM tables
  - System control space (SCS)
  - Breakpoint unit (BPU)
  - Data watchpoint and Trace unit (DWT)
  - Cross trigger interface (CTI)
- CPU1 debug features
  - ROM table
  - System control space (SCS)
  - Breakpoint unit (FPB)
  - Data watchpoint and Trace unit (DWT)
  - Instrumentation trace macrocell (ITM)
  - Cross trigger interface (CTI)
  - Trace port interface unit (TPU)

CPU2 debug access via CPU2 AHB-AP and its associated AHB bus is disabled.

The CPU1 debug features are accessible by the debugger via the CPU1 AHB-AP.

Additional information can be found in the Arm® documents referenced in *Section 32.19*.

## 32.2 Debug use cases

The trace and debug system is designed to support a variety of typical use cases:

- Low cost trace
  Limited trace capability is available over the single-wire debug output. This supports code instrumentation using “printf”, tracing of data and address watchpoints, interrupt

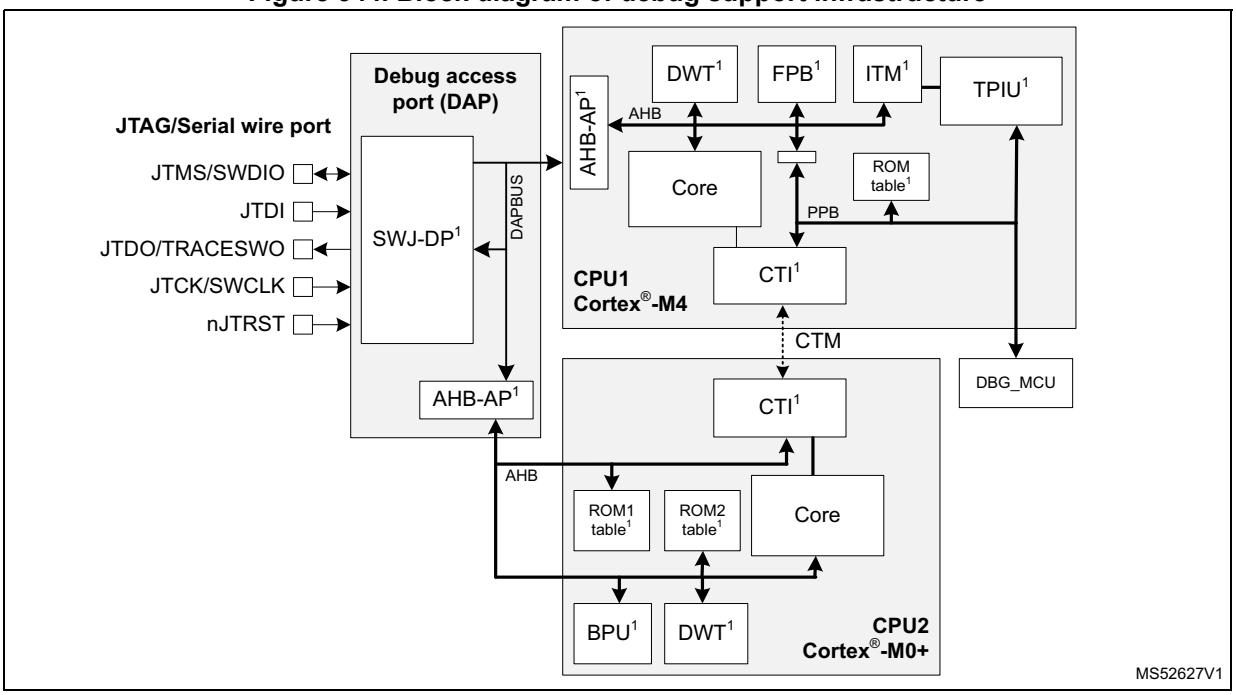
detection and program counter sampling. Single-wire trace can be maintained even when one or both processors are switched off or clock-stopped.

- Breakpoint debugging of each core independently
  Both processor cores can be simultaneously and independently debugged using equipment connected to the JTAG/SWD debug port. This enables, among others, breakpoint and watchpoint setting, code stepping and memory access.
- Synchronous debugging of both cores
  When one core stops due to a breakpoint or a debugger stop command, the other core can be stopped as well. Similarly, the cores can be restarted at the same time. This allows the user to debug loosely coupled applications, which require the processors to remain synchronized.
- Tracing code execution via the trace port
  Trace information from the CPU1 (Cortex®-M4) is combined into a single trace stream and sent to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.

## 32.3 DBG functional description

### 32.3.1 DBG block diagram

**Figure 344. Block diagram of debug support infrastructure**




1. Arm® CoreSight™ component

### 32.3.2 DBG pins and internal signals

**Table 187. JTAG/Serial-wire debug port pins**

| Pin name | JTAG debug port | | SW debug port | | Pin assignment |
|---|---|---|---|---|---|
| | Type | Description | Type | Description | |
| JTMS/SWDIO | I | JTAG test mode select | IO | Serial wire data in/out | PA13 |
| JTCK/SWCLK | I | JTAG test clock | I | Serial wire clock | PA14 |
| JTDI | I | JTAG test data input | - | - | PA15 |
| JTDO/TRACESWO | O | JTAG test data output | - | - | PB3 |
| nJTRST | I | JTAG test reset | - | - | PB4 |

**Table 188. Single Wire Trace port pins**

| Pin name | Type | Description | Pin assignment |
|---|---|---|---|
| TRACESWO | O | Single wire trace asynchronous data out | PB3(1) |

1. TRACESWO is multiplexed with JTDO. This means that single wire trace is only available when using the serial wire debug interface, and not when using JTAG.

### 32.3.3 DBG power domains

The debug components are located in the core power domain. This means that debugger connection is not possible in shutdown or standby low power modes. To avoid losing the connection when the device enters standby mode, it is possible to maintain the power to the core by setting a bit in the DBGMCU. This keeps the processor clocks active, and holds off the reset, so that the debug session is maintained.

### 32.3.4 DBG clocks

The debugger supplies the clock for the debug port via the debug interface pin, JTCK/SWCLK. This clock is used to register the serial input data in both serial wire and JTAG mode, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DAPCLK domain, which covers the rest of the SWJ-DP and the CPU2 access port.

The DAPCLK is a gated version of the system HCLK4.

The DAPCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the debug port CTRL/STAT register. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in the debug port CTRL/STAT register. The DAPCLK is disabled at power up, after OBL, and after wakeup from Standby, and must be disabled when the debugger is disconnected, to reduce power consumption.

The debug and trace components included in the processors (among them ITM, DWG, FPB) are clocked with the corresponding core clock.

### 32.3.5 Debug and low power modes

The devices include power saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off, or the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power saving mode emulation has been implemented. If emulation is enabled for a domain, the domain still enters power saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power saving mode, but the debugger does not lose the connection.

Emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information refer to *Section 32.8*.

### 32.3.6 DBG reset

The debug port (SWJ-DP) is reset by a power-on reset or an OBL reset, and when waking up from Standby mode.

## 32.4 Serial wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a Coresight™ component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- a 5-pin standard JTAG interface (JTAG-DP)
- a 2-pin (clock + data) “serial-wire debug” port (SW-DP)

The two modes are mutually exclusive, since they share the same IO pins.

By default the JTAG-DP is selected after a system or a power-on reset. The five IO pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

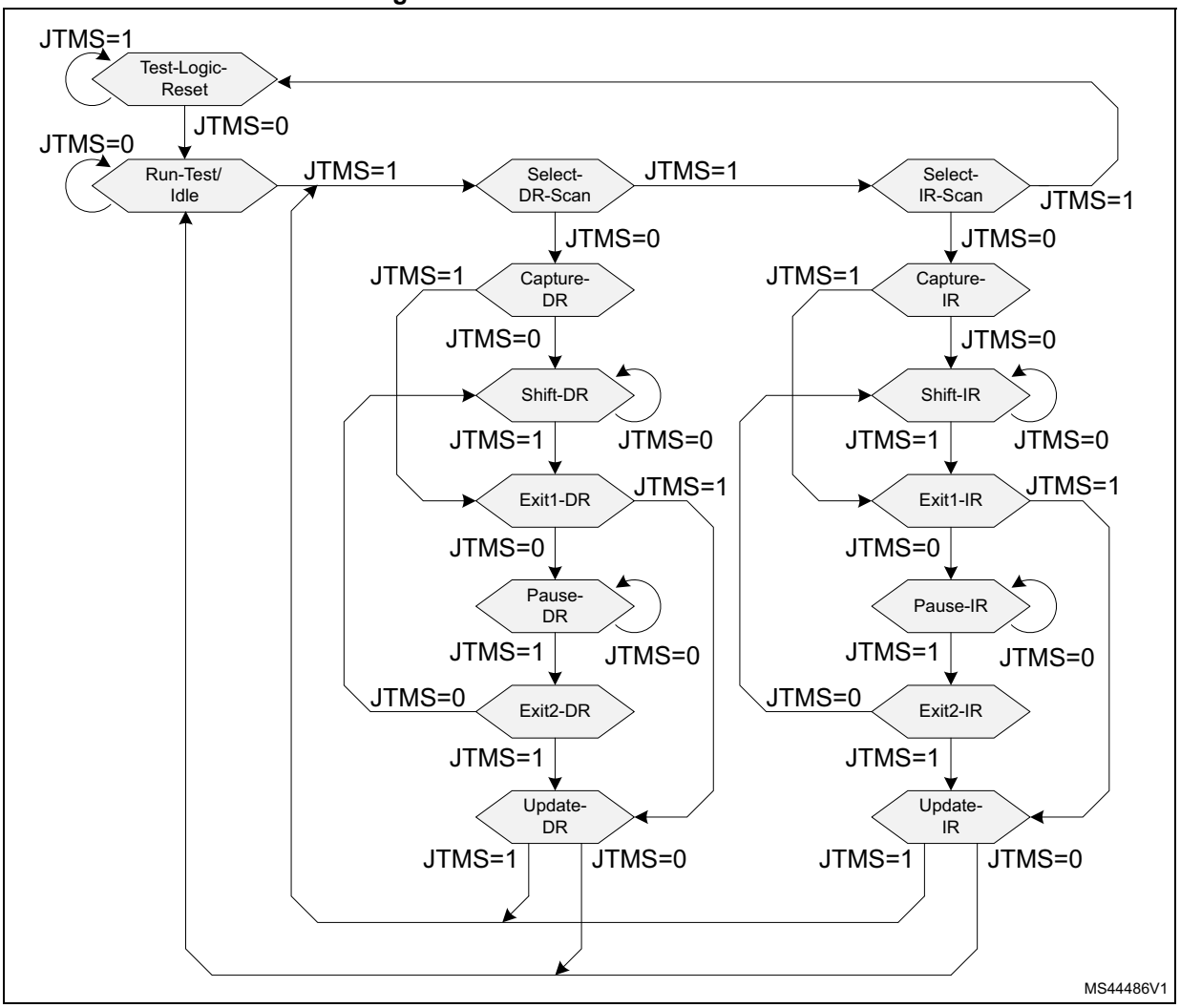
In SW-DP mode, the unused JTAG pins JTDI, JTDO and nJTRST can be used for other functions. It should be noted that all SWJ port IOs can be reconfigured to other functions by software, but debugging is no longer be possible.

### 32.4.1 JTAG debug port

There are two TAPs on the JTAG debug port, the JTAG-DP TAP and the BSC TAP.

The JTAG-DP implements a TAP state machine (TAPSM), shown in *Figure 345*, based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR), and the other one with a number of data registers (DR).

**Figure 345. JTAG TAP state machine**




The operation of the JTAG-DP is as follows:

- When the TAPSM goes through the Capture-IR state, 0b0001 is transferred onto the Instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
- While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
  - The LSB of the IR scan chain is output on JTDO.
  - Bit[n] of the IR scan chain is transferred to bit[n-1].
  - The value on JTDI is transferred to the MSB of the IR scan chain.
- When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the Instruction register.
- When the TAPSM goes through the Capture-DR state, a value is transferred from one

of the Data registers onto one of the DR scan chains, connected between JTDI and JTDO.

- The value held in the Instruction register determines which Data register (and associated DR scan chain) is selected.
- This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shift in the Shift-IR state.
- When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected Data register.
- When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in *Table 189*. The total IR instruction length is 9 bits.

**Table 189. JTAG-DP data registers**

| IR instruction | DR register | Scan chain length | Description |
|---|---|---|---|
| 0000 to 0111 | (BYPASS) | 1 | *Not implemented:* BYPASS selected |
| 1000 | ABORT | 35 | *ABORT register*<br>– Bits 31:1 = Reserved<br>– Bit 0 = APABORT: write 1 to generate an AP abort. |
| 1001 | (BYPASS) | 1 | *Reserved:* BYPASS selected |
| 1010 | DPACC | 35 | *Debug port access register*<br>Initiates the debug port and gives access to a debug port register.<br>– When transferring data IN:<br>Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request<br>Bits 2:1 = A[3:2] = 2-bit address of a debug port register.<br>Bit 0 = RnW = Read request (1) or write request (0).<br>– When transferring data OUT:<br>Bits 34:3 = DATA[31:0] = 32-bit data read following a read request<br>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:<br>010 = OK/FAULT<br>001 = WAIT<br>OTHER = reserved |
| 1011 | APACC | 35 | *Access port access register*<br>Initiates an access port and gives access to an access port register.<br>– When transferring data IN:<br>Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request<br>Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register.<br>Bit 0 = RnW= Read request (1) or write request (0).<br>– When transferring data OUT:<br>Bits 34:3 = DATA[31:0] = 32-bit data read following a read request<br>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:<br>010 = OK/FAULT<br>001 = WAIT<br>OTHER = reserved |
| 1100 | (BYPASS) | 1 | *Reserved:* BYPASS selected |

**Table 189. JTAG-DP data registers (continued)**

| IR instruction | DR register | Scan chain length | Description |
|---|---|---|---|
| 1101 | (BYPASS) | 1 | *Reserved:* BYPASS selected |
| 1110 | IDCODE | 32 | *ID Code*<br>0x6BA0 0477: Arm® JTAG debug port ID code |
| 1111 | BYPASS | 1 | *Bypass*<br>A single JTCK cycle delay is inserted between JTDI and JTDO |

The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification *[1]*.

### 32.4.2 SW debug port

The Serial Wire Debug protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock. A transfer comprises three phases:

1. packet request (8 bits) transmitted by the host, see *Table 190*.
2. acknowledge response (3 bits) transmitted by the target, see *Table 191*.
3. data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read), see *Table 192*.

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

**Table 190. Packet request**

| Bit field | Name | Description |
|---|---|---|
| 0 | Start | Must be “1” |
| 1 | APnDP | – 0: DP register access - see *Table 189* for a list of DP registers<br>– 1: AP register access - see *Section 32.5: Access ports* |
| 2 | RnW | – 0: Write request<br>– 1: Read request |
| 4:3 | A(3:2) | Address field of the DP or AP register (refer to ) |
| 5 | Parity | Single bit parity of preceding bits |
| 6 | Stop | 0 |
| 7 | Park | Not driven by host, must be read as “1” by target |

**Table 191. ACK response**

| Bit field | Name | Description |
|---|---|---|
| 2:0 | ACK | – 000: FAULT<br>– 010: WAIT<br>– 100: OK |

**Table 192. Data transfer**

| Bit field | Name | Description |
|---|---|---|
| 31:0 | WDATA or RDATA | Write or Read data |
| 32 | Parity | Single bit parity of 32 data bits |

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial Wire debug protocol, refer to the Arm® Debug Interface Architecture Specification *[1]*.

*Note: The SWJ-DP implements SWD protocol version 2.*

### 32.4.3 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in *Table 193*.

The debugger can access the DP registers as follows:

- Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a Read or Write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit reset (see *Table 190*). The write data are sent in the data phase.
- To access one of the banked DP registers at address 0x4, the register number must first be written to the DP_SELECT register at address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the content of the DP_SELECT register.

### 32.4.4 DP debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x5BA0 2477

Read only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REVISION[3:0] | | | | PARTNO[7:0] | | | | | | | | Res. | Res. | Res. | MIN |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VERSION[3:0] | | | | DESIGNER[10:0] | | | | | | | | | | | Res. |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | |

Bits 31:28 **REVISION[3:0]**: Revision code
0x5

Bits 27:20 **PARTNO[7:0]**: Part number for the debug port
0xBA

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

Bit 16 **MIN**: Minimal debug port (MINDP) implementation
0x0: MINDP not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]**: DP architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

### 32.4.5 DP abort register (DP_ABORTR)

Address offset: 0x0

Reset value: 0x0000 0000

Write only

| 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. | ORUNERR CLR | WDERR CLR | STKERR CLR | STKCMP CLR | DAPABORT |
| | | | | | | | | | | | w | r | r | r | |

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

Bit 4 **ORUNERRCLR**: Overrrun error clear
0: No effect
1: Clear CTRL/STAT.STICKYORUN bit

Bit 3 **WDERRCLR**: Write data error clear
0: No effect
1: Clear CTRL/STAT.WDATAERR bit

Bit 2 **STKERRCLR**: Sticky error clear
0: No effect
1: Clear CTRL/STAT.STICKYERR bit

Bit 1 **STKCMPCLR**: Sticky compare clear
0: No effect
1: Clear CTRL/STAT.STICKYCMP bit

Bit 0 **DAPABORT**: Aborts current AP transaction if an excessive number of WAIT responses are returned, indicating that the transaction is stalled.
0: No effect
1: Abort transaction

### 32.4.6 DP control and status register (DP_CTRL/STATR)

Address offset: 0x4 and DP_SELECTR.DPBANKSEL = 0

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | CDBGPWRUPACK | CDBGPWRUPREQ | Res. | Res. | Res. | Res. | TRNCNT[11:4] | | | | | | | |
| | | r | r | | | | | r | r | r | r | | | | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRNCNT[3:0] | | | | CMASKLANE[3:0] | | | | WDATAERR | READOK | STICKYERR | STICKYCMP | TRNMODE[1:0] | | STICKYORUN | ORUNDETECT |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | |

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

Bit 29 **CDBGPWRUPACK**: See description in *Section 32.3.4: DBG clocks*.
0 = DAPCLK gated
1 = DAPCLK enabled

Bit 28 **CDBGPWRUPREQ**: Controls the DAPCLK enable request signal.
0 = Requests DAPCLK gating
1 = Requests DAPCLK enable

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:12 **TRNCNT[11:0]**: Transaction counter. To program a sequence of transactions to incremental addresses via an AP, TRNCNT is loaded with the number of transactions to perform. It is decremented at the successful completion of each transaction.

Bits 11:8 **MASKLANE[3:0]**: Indicates the bytes to be masked in pushed-compare and pushed-verify operations (CTRL/STAT.TRNMODE = 1 or 2). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address.

0b1XXX = include byte lane 3 in comparisons
0bX1XX = include byte lane 2 in comparisons
0bXX1X = include byte lane 1 in comparisons
0bXXX1 = include byte lane 0 in comparisons

Bit 7 **WDATAERR**. Write data error (read only) in SW-DP. Indicates that:

– there is a a parity or framing error on the data phase of a write, or
– a write that has been accepted by the DP is then discarded without being submitted to the AP.

This bit is reset by writing 1 to the ABORT.WDERRCLR bit.

0: No error
1: Error has occurred

Reserved in JTAG-DP.

Bit 6 **READOK**. AP read response (read only) in SW-DP. Indicates the response to the last AP read access.

0: Read not OK
1: Read OK

Reserved in JTAG-DP.

Bit 5 **STICKYERR**. Transaction error (read only in SW-DP, R/W in JTAG-DP). Indicates that an error occurred in an AP transaction.

0: No error
1: Error has occurred

In the SW-DP, this bit is reset by writing 1 to the ABORT.STKERRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 4 **STICKYCMP**. Compares match (read only in SW-DP, R/W in JTAG-DP). Indicates that a match occurred in a pushed operation.

0: Match if TRNMODE = 0x1; no match if TRNMODE = 0x2
1: No match if TRNMODE = 0x1; match if TRNMODE = 0x2

In the SW-DP, this bit is reset by writing 1 to the ABORT.STKCMPCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bits 3:2 **TRNMODE[1:0]**: Transfer mode for AP write operations (for read operations, this field must be set to 0x0).

0x0: Normal operation. AP transactions are passed directly to the AP.
0x1: Pushed-verify operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data and if they do not match, the STICKYCMP bit is set.
0x2: Pushed-compare operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data and if they match, the STICKYCMP bit is set.
0x3: reserved

In pushed operation, only the data bytes indicated by the MASKLANE field are included in the compare.

Bit 1 **STICKYORUN**. Overrun (read only in SW-DP, R/W in JTAG-DP). Indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.

0: No overrun

1: Overrun occurred

In the SW-DP, this bit is reset by writing 1 to the ABORT.ORUNERRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 0 **ORUNDETECT**. Overrun detection mode enable.

0: Overrun detection disabled

1: Overrun detection enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions is blocked until the STICKYORUN bit is cleared.

### 32.4.7 DP data link control register (DP_DLCR)

Address offset: 0x4 and DP_SELECTR.DPBANKSEL = 1

Reset value: 0x0000 0040

| 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. | TURNROUND[1:0] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | r | r | | | | | | | | |

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

Bits 9:8 **TURNROUND[1:0]**: Tristate period for SWDIO.

0x0: 1 data bit period

0x1: 2 data bit periods

0x2: 3 data bit periods

0x3: 4 data bit periods

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value (set to 1).

Bits 5:0 Reserved, must be kept at reset value.

### 32.4.8 DP target identification register (DP_TARGETIDR)

Address offset: 0x4 and DP_SELECTR.DPBANKSEL = 2

Reset value: 0x0495 0041

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TREVISION[3:0] | | | | TPARTNO[15:4] | | | | | | | | | | | |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TPARTNO[3:0] | | | | TDESIGNER[10:0] | | | | | | | | | | | Res. |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | |

Bits 31:28 **TREVISION**: Target revision.
0x0: revision 1

Bits 27:12 **TPARTNO**: Target part number.
0x4950: STM32WB50CG/30CE

Bits 11:1 **TDESIGNER**: Target designer JEDEC code.
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value (set to 1)

### 32.4.9 DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4 and DP_SELECTR.DPBANKSEL = 3

Reset value: 0x0000 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TINSTANCE[3:0] | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PROTSVN[3:0] | | | |
| | | | | | | | | | | | | r | r | r | r |

Bits 31:28 **TINSTANCE[3:0]**: Target instance number. Defines the instance number for this device in a multi-drop system.
0x0: Instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial Wire Debug protocol version.
0x1: Version 2

### 32.4.10 DP resend register (DP_RESENDR)

Address offset: 0x8

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESEND[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESEND[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **RESEND**: Returns the value that was returned by the last AP read or DP RDBUFF read. Used in the event of a corrupted read transfer.

### 32.4.11 DP access port select register (DP_SELECTR)

Address offset: 0x8

Reset value: Unknown

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| APSEL[7:0] | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| w | w | w | w | w | w | w | w | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | APBANKSEL[3:0] | | | | DPBANKSEL[3:0] | | | |
| | | | | | | | | w | w | w | w | w | w | w | w |

Bits 31:24 **APSEL[7:0]**: Access port select. Selects the access port for the next transaction.
0x0: AP0 - CPU1 (Cortex®-M4) debug access port (AHB-AP)
0x1: AP1 - CPU2 (Cortex®-M0+) debug access port (AHB-AP)
0x2 to 0xFF: reserved

Bits 27:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select. Selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select. Selects the register at address 0x4 of the debug port.
0x0: CTRL/STAT register
0x1: DLCR register
0x2: TARGETID register
0x3: DLPIDR register
0x4 to 0xF: Reserved

### 32.4.12 DP read buffer register (DP_BUFFR)

Address offset: 0xC

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDBUFF[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDBUFF[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **RDBUFF[31:0]**: Contains the value that was returned by the last AP read access. The value returned by an AP read access can either be obtained using a second read access to the same address, which initiates a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.

### 32.4.13 DP target selection register (DP_TARGETSELR)

Address offset: 0xC

Reset value: Unknown

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TINSTANCE[3:0] | | | | TPARTNO[15:4] | | | | | | | | | | | |
| 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 |
| TPARTNO[3:0] | | | | TDESIGNER[10:0] | | | | | | | | | | | Res. |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | |

Bits 31:28 **TINSTANCE[3:0]**: Target instance number. Defines the instance number for the target device in a multi-drop system. These bits must be written with the same value used for DLPIDR.TINSTANCE to select this device.

Bits 27:12 **TPARTNO[15:0]**: Target part number. Defines the part number for the target device. These bits must be written with the same value used for TARGETID.TPARTNO to select this device.

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code. Defines the JEDEC code for the target device. These bits must be written with the same value used for TARGETID.TDESIGNER to select this device.

Bit 0 Reserved, must be kept at reset value (set to 1).

### 32.4.14 Debug port register map and reset values

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interface.

The debug port address is 2-bit wide, defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

**Table 193. Debug port register map and reset values**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0b00 | DP_DPIDR | REVISION [3:0] | | | | PARTNO[7:0] | | | | | | | | Res. | Res. | Res. | MIN | VERSION [3:0] | | | | DESIGNER[10:0] | | | | | | | | | | | Res. |
| | Reset value | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | | | | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0b00 | DP_ABORTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ORUNERRCLR | WDERRCLR | STKERRCLR | STKCMPCLR | DAPABORT |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| 0b01(1) | DP_CTRL/STATR | Res. | Res. | CDBGPWRUPACK | CDBGPWRUPREQ | Res. | Res. | Res. | Res. | TRNCNT[11:0] | | | | | | | | | | | | APPSET[3:0] | | | | WDATAERR | READOK | STICKYERR | STICKYCMP | TRNMODE[1:0] | | STICKYORUN | ORUNDETECT |
| | 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 |
| 0b01(1) | DP_DLCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TURNROUND [1:0] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | | | | | | | | |
| 0b01(2) | DP_TARGETIDR | TREVISION [3:0] | | | | TPARTNO[15:0] | | | | | | | | | | | | | | | | TDESIGNER[10:0] | | | | | | | | | | | Res. |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 0b01(3) | DP_DLPIDR | TINSTANCE [3:0] | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PROTSVN [3:0] | | | |
| | Reset value | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 1 |
| 0b10 | DP_RESENDR | RESEND[0:31] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 |
| 0b10 | DP_SELECTR | APSEL[7:0] | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | APBANKSEL [3:0] | | | | DPBANKSEL [3:0] | | | |
| | Reset value | x | x | x | x | x | x | x | x | | | | | | | | | | | | | | | | | x | x | x | x | x | x | x | x |
| 0b11 | DP_BUFFR | RDBUFF[0:31] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 |
| 0b11 | DP_TERGETSELR | TINSTANCE [3:0] | | | | TPARTNO[15:0] | | | | | | | | | | | | | | | | TDESIGNER[10:0] | | | | | | | | | | | Res. |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | |

1. DP_SELECTR.DPBANKSEL = 1.
2. DP_SELECTR.DPBANKSEL = 2.
3. DP_SELECTR.DPBANKSEL = 3.

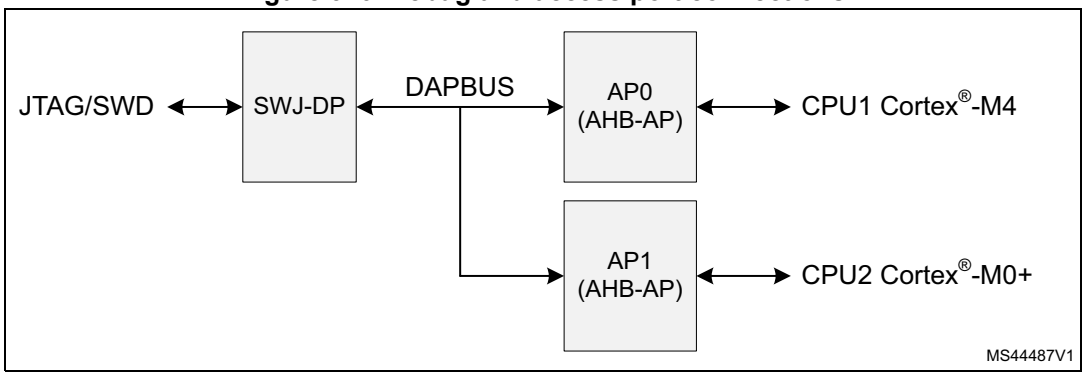
## 32.5 Access ports

As shown in *Figure 346*, there are two access ports (AP) attached to the DP:

1. AP0: CPU1 (Cortex®-M4) access port (AHB-AP): enables access to the debug and trace features integrated in the Cortex®-M4 processor core via its internal AHB bus.
2. AP1: CPU2 (Cortex®-M0+) access port (AHB-AP): enables access to the debug and trace features integrated in the Cortex®-M0+ processor core via its internal AHB bus.

Both access ports are of type MEM-AP, the debug and trace component registers are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 194*.

**Figure 346. Debug and access port connections**




The address of the AP registers is composed of

- Bits [7:4]: content of the SELECT register APBANKSEL field in the DP (see *Section 32.4.11*)
- Bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 193*) or of the SW-DP Packet Request (see *Table 190*), depending on the debug interface used
- Bits [1:0]: Always set to 0

The content of the SELECT register APSEL field in the DP defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program the SELECT register APSEL field in the DP to choose one of the APs, and the APBANKSEL field to select the register bank to be accessed (see *Section 32.4.11*).
2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a Read or Write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit set (see *Table 190*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the MEM-AP registers (i.e. using the above AP register access procedure) as follows:

1. Program the transaction target address in the TAR register.
2. Program the CSW register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the DRW register to initiate a bus transaction at the address held in the TAR register. Alternatively, a read or write to Banked data register BDn triggers an access to address TAR[31:4] + n (this enables up to four consecutive addresses to be accessed without changing the address in the TAR register).

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification *[1]*.

### 32.5.1 AP control/status word register (AP_CSWR)

Address offset: 0x0

Reset value: 0x2300 0040

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | SPROT | Res. | PROT[4:0] | | | | | SPI STATUS | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | 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. | MODE[3:0] | | | | TRIN PROG | DEVICE EN | ADDRIN[1:0] | | Res. | SIZE[2:0] | | |
| | | | | r | r | r | r | r | r | r | r | | r | r | r |

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: Secure transfer request. In the AHB-APs this field sets the protection attribute HPROT[6] of the bus transfer.

0: If SPIDEN is high, secure transfer. If SPIDEN is low, non-secure transfer.
1: Non-secure transfer.

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **PROT[4:0]**: Bus transfer protection. In the AHB-APs this field sets the protection attributes HPROT[4:0] of the bus transfer.

0bXXXX0: Instruction fetch
0bXXXX1: Data access
0bXXX0X: User mode
0bXXX1X: Privileged mode
0bXX0XX: Non-bufferable
0bXX1XX: Bufferable
0bX0XXX: Non-cacheable
0bX1XXX: Cacheable
0b0XXXX: Non-exclusive
0b1XXXX: Exclusive

Bit 23 **SPISTATUS**: Status of SPIDEN option bit (read only). This signal determines whether the debugger can access secure memory.

0: Secure AHB transfers are blocked
1: Secure AHB transfers are allowed

Bits 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: Barrier support enabled. Defines if memory barrier operation is supported.

0x0: Not supported

Bit 7 **TRINPROG**: Transfer in progress (read only). Indicates if a bus transfer is in progress on the AP.

0x0: No transfer in progress.
0x1: Bus transfer in progress.

Bit 6 **DEVICEEN**: Device enabled (read only). Defines whether the AP can be accessed.

0x1: AP access enabled.

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode. Defines whether TAR address is automatically incremented after a transaction.

0x0: No auto-increment

0x1: Address is incremented by the size in bytes of the transaction (SIZE field).

0x2: Packed transfers enabled. A 32-bit AP access gives rise to 1 x 32-bit, 2 x 16-bit, or 4 x 8-bit bus transactions, corresponding to the programmed transaction size. Data are packed or unpacked accordingly.

0x3: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction

0x0: Byte (8-bit)

0x1: Halfword (16-bit)

0x2: Word (32-bit)

0x3-0x7: reserved

### 32.5.2 AP transfer address register (AP_TAR)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TA[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 |
| TA[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **TA[31:0]:** Address of current transfer

### 32.5.3 AP data read/write register (AP_DRWR)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TD[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 |
| TD[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **TD[31:0]:** Data of current transfer

### 32.5.4 AP banked data registers (AP_BD0-3R)

Address offset: 0x10 (DB0R)

Address offset: 0x14 (BD1R)

Address offset: 0x18 (BD2R)

Address offset: 0x1C (BD3R)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TBD[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 |
| TBD[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **TBD[31:0]:** Banked data of current transfer to address TAR. TA+ AP_BDnR address [3:2] + 0b00. Auto address incrementing is not performed on AP_BD0-3R. Banked transfers are only supported for word transfers.

### 32.5.5 AP base address register (AP_BASER)

Address offset: 0xF8

Reset value: 0xE00F F003 (AP0), 0xF000 0003 (AP1)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASEADDR[19:4] | | | | | | | | | | | | | | | |
| 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 |
| BASEADDR[3:0] | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | FORMAT | ENTRY PRESENT |
| r | r | r | r | | | | | | | | | | | r | r |

Bits 31:12 **BASEADDR[19:0]**: Base address (bits 31 to 12) of ROM table for the AP. The twelve LSBs are 0 since the ROM table must be aligned on a 4-Kbyte boundary.
AP0 CPU1 (Cortex®-M4) AHB-AP: 0xE00FF
AP1 CPU2 (Cortex®-M0+) AHB-AP: 0xF0000

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: Base address register format.
1: Arm® debug interface v5

Bit 0 **ENTRYPRESENT**: Indicates that debug components are present on the access port bus.
1: Debug components are present

### 32.5.6 AP identification register (AP_IDR)

Address offset: 0xFC

Reset value: 0x2477 0011 (AP0), 0x6477 0001 (AP1)

Read only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REVISION[3:0] | | | | JEDECBANK[3:0] | | | | JEDECCODE[6:0] | | | | | | | MEMAP |
| 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 |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | IDENTITY[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

Bits 31:28 **REVISION[3:0]**:
0x2: r0p3
0x6: r0p7

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank.
0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code.
0x3B: Arm®

Bit 16 **MEMAP**: Memory access port.
0x1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: Identifies the type of AP.
0x11: CPU1 (Cortex®-M4) AHB-AP (AP0)
0x01: CPU2 (Cortex®-M0+) AHB-AP (AP1)

### 32.5.7 Access port register map and reset values

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interface.

The access port address is 8-bit wide, defined by debug port register DP_SELECTR.APBANKSEL[3:0] field, and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

**Table 194. Access port register map and reset values**

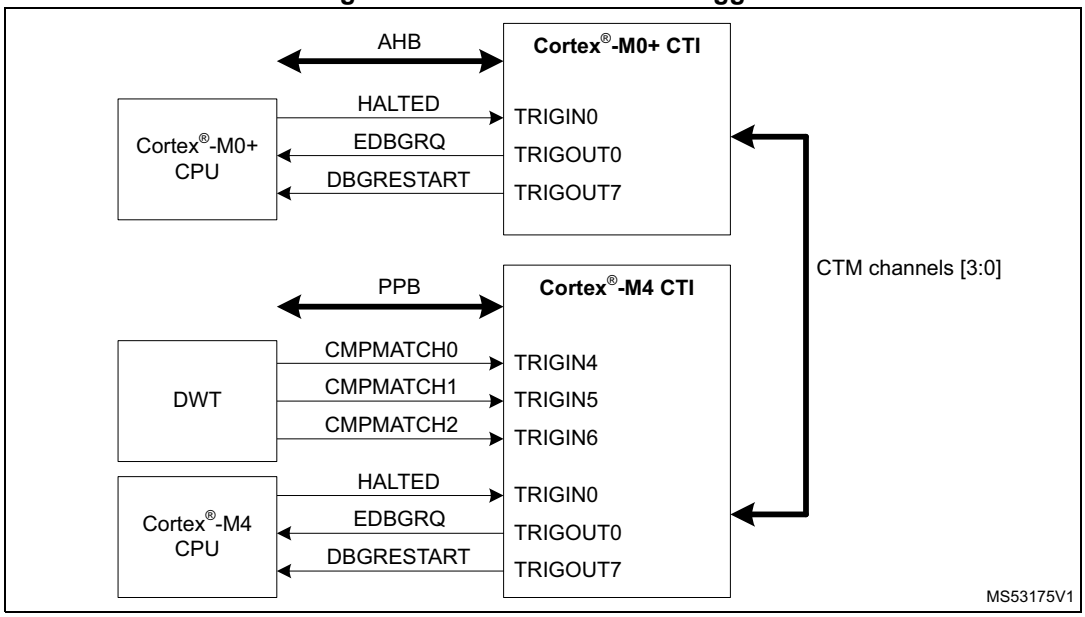
| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | AP_CSWR | Res. | SPROT | Res. | PROT[4:0] | | | | | SPISTATUS | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MODE[3:0] | | | | TRINPROG [1:0] | | ADDRIN [1:0] | | Res. | SIZE [2:0] | | |
| | Reset value | | 0 | | 0 | 0 | 0 | 1 | 1 | 0 | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | | 0 | 0 | 0 |
| 0x04 | AP_TAR | TA[0:31] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | AP_DRWR | TD[0:31] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | AP_BD0R | TBD[0:31] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | AP_BD1R | TBD[0:31] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | AP_BD2R | TBD[0:31] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | AP_BD3R | TBD[0:31] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 |
| 0xF8 | AP_BASER | BASEADDR[0:19] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FORMAT | ENTRYPRESENT |
| | Reset value (AP0) | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | Reset value (AP1) | 1 | 1 | 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 | 0 | 1 | 1 |
| 0xFC | AP_IDR | REVilSION [3:0] | | | | JEDECBANK [3:0] | | | | JEDECCODE[6:0] | | | | | | | MEMAP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | IDENTITY[7:0] | | | | | | | |
| | Reset value (AP0) | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | | | | | | | | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| | Reset value (AP1) | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## 32.6 Cross trigger interface (CTI) and matrix (CTM)

The Cross trigger interfaces (CTIs) and Cross trigger matrix (CTM), taken together, form the CoreSight™ embedded cross trigger (see *Figure 347*).

There are two CTI components, one dedicated to the CPU2 and one dedicated to the CPU1. The CTIs are connected to each other via the CTM. The CTI registers are accessible to the debugger via the corresponding access port and associated AHB.

**Figure 347. Embedded cross trigger**




The CTIs enable events from various sources to trigger debug and/or trace activity. For example, a breakpoint reached in one of the processor cores can stop the other processor, or a transition detected on an external trigger input can start code trace.

Each CTI has up to eight trigger inputs and eight trigger outputs. Any input can be connected to any output, on the same CTI, or on another CTI via the CTM.

The trigger input and output signals for each CTI are listed in tables *195* to *198*.

**Table 195. CPU2 CTI inputs**

| No. | Source signal | Source component | Comments |
|---|---|---|---|
| 0 | HALTED | CPU2 | CPU2 halted - Indicates CPU2 is in debug mode |
| 1 | - | - | Not used |
| 2 | - | - | Not used |
| 3 | - | - | Not used |
| 4 | - | - | Not used |
| 5 | - | - | Not used |
| 6 | - | - | Not used |
| 7 | - | - | Not used |

**Table 196. CPU2 CTI outputs**

| No. | Output signal | Destination component | Comments |
|---|---|---|---|
| 0 | EDBGRQ | CPU2 | CPU2 halt request - Puts CPU2 in debug mode |
| 1 | - | - | Not used |
| 2 | - | - | Not used |
| 3 | - | - | Not used |
| 4 | - | - | Not used |
| 5 | - | - | Not used |
| 6 | - | - | Not used |
| 7 | DBGRESTART | CPU2 | CPU2 restart request - CPU2 exits debug mode |

**Table 197. CPU1 CTI inputs**

| No. | Source signal | Source component | Comments |
|---|---|---|---|
| 0 | HALTED | CPU1 | CPU1 halted - Indicates CPU1 is in debug mode |
| 1 | - | - | Not used |
| 2 | - | - | Not used |
| 3 | - | - | Not used |
| 4 | ETMTRIGGER0 | CPU1 DWT | Trace trigger - Enables CPU1 execution trace |
| 5 | ETMTRIGGER1 | CPU1 DWT | Trace trigger - Enables CPU1 execution trace |
| 6 | ETMTRIGGER2 | CPU1 DWT | Trace trigger - Enables CPU1 execution trace |
| 7 | - | - | Not used |

**Table 198. CPU1 CTI outputs**

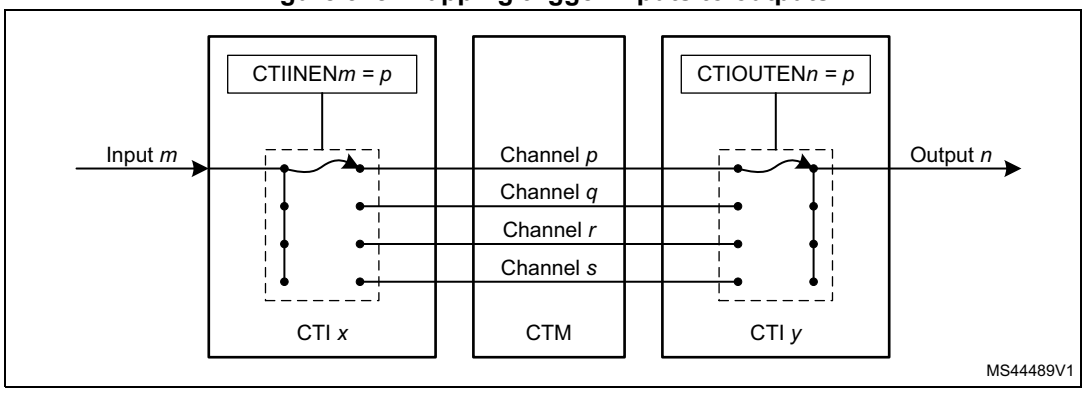
| No. | Source signal | Source component | Comments |
|---|---|---|---|
| 0 | EDBGRQ | CPU1 | CPU1 halt request - Puts CPU1 in debug mode |
| 1 | - | - | Not used |
| 2 | - | - | Not used |
| 3 | - | - | Not used |
| 4 | - | - | Not used |
| 5 | - | - | Not used |
| 6 | - | - | Not used |
| 7 | DBGRESTART | CPU1 | CPU1 restart request - CPU1 exits debug mode |

There are four event channels in the cross trigger matrix, thus enabling up to four, parallel, bidirectional connections between trigger inputs and outputs on different CTIs. To connect input number *m* on CTI *x* to output number *n* on CTI *y*, the input must be connected to an event channel *p* using the CTIINEN*m* register of CTI *x*. The same channel *p* must be connected to the output using the CTIOUTEN*n* register of CTI *y*.

*Note: This applies even if the input and output belong to the same CTI.*

An input can be connected to more than one channel (up to four), so an input can be routed to several outputs. Similarly, an output can be connected to several inputs. It is also possible to connect several inputs/outputs to the same channel.

**Figure 348. Mapping trigger inputs to outputs**




## Example configurations

When either CPU core hits a breakpoint, stop the other core. Restart the two cores synchronously.

To stop both cores when one of them stops the HALTED output of each core must be connected to the EDBGRQ input of the opposite core.

Referring to *Table 195* and *Table 197*, we see that the HALTED signal from the CPU2 is connected to input 0 of the CPU2 CTI, and the same signal from the CPU1 is connected to the same input on the CPU1 CTI. Hence we program the CTIIEN0 register on each CTI to connect these inputs to a CTM channel (eg. channel 0).

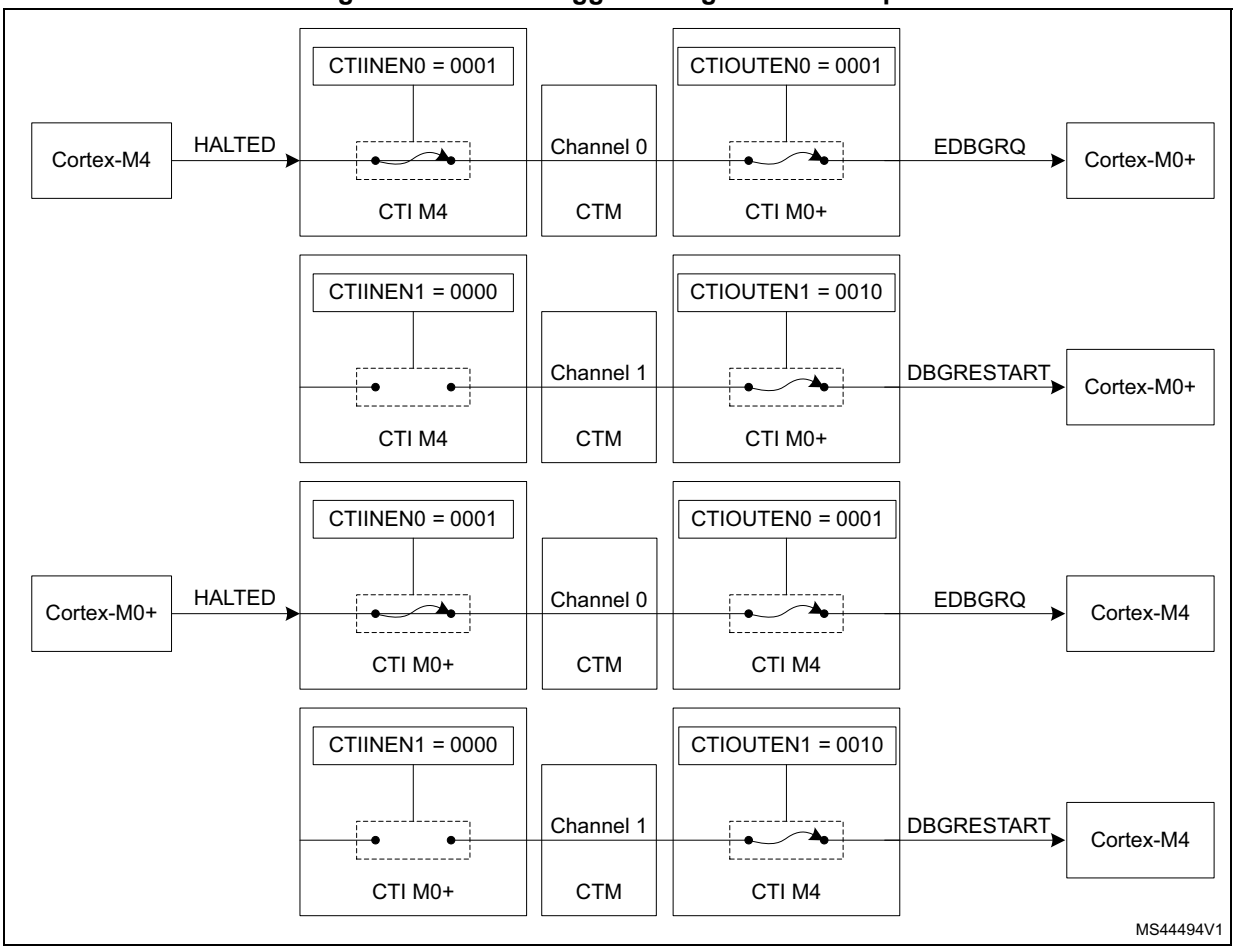
From *Table 196* and *Table 198* we see that the EDBGRQ signals to the CPUs are connected to output 0 of the respective CTIs. So we program the CTIOUTEN0 register on each CTI to connect these outputs to the same CTM channel.

To restart both cores simultaneously the debugger must use the APPPULSE register in one of the CTIs. This allows the debugger to generate a pulse on any of the four CTM channels. The channel must be connected to the DBGRESTART signal of both cores.

From *Table 196* and *Table 198* we see that the DBGRESTART signals to the CPUs are connected to output 1 of the respective CTIs. So we program the CTIOUTEN1 register on each CTI, to connect these outputs to an unused CTM channel (e.g. channel 1).

The above configuration is illustrated in *Figure 349*.

**Figure 349. Cross trigger configuration example**




To force the processors to restart simultaneously, use the following procedure:

1. Clear the debug request by writing 0x01, then 0x00, to the CTIINTACK register in each CTI.
2. Cause a pulse on channel 1 by writing 0x02 to the APPPULSE register in either CTI. This generates a restart request to both processors.

   Note that the debugger can also force both cores to stop simultaneously by writing 0x01 to the APPPULSE register in either CTI, which generates a pulse on channel 0.

For more information on the Cross-Trigger Interface CoreSight™ component refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual *[2]*.

## 32.7 Cross trigger interface registers

The register file base address is 0xE0043000 for CPU1 CTI and 0xF0001000 for CPU2 CTI. CPU1 CTI and CPU2 CTI are accessed via different access ports. The registers are the same for each CTI.

### 32.7.1 CTI control register (CTI_CONTROLR)

Address offset: 0x000

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. | GLBEN |
| | | | | | | | | | | | | | | | rw |

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

Bit 0 **GLBEN**: Global enable.

0: Cross-triggering disabled

1: Cross-triggering enabled

### 32.7.2 CTI trigger acknowledge register (CTI_INTACKR)

Address offset: 0x010

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. | INTACK[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bit 7:0 **INTACK[7:0]**: Trigger acknowledge.

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

### 32.7.3 CTI application trigger set register (CTI_APPSETR)

Address offset: 0x014

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. | APPSET[3:0] | | | |
| | | | | | | | | | | | | rw | | | |

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

Bit 3:0 **APPSET[3:0]**: Set channel event.

Read:

0bXXX0: Channel 0 event inactive
0bXXX1: Channel 0 event active
0bXX0X: Channel 1 event inactive
0bXX1X: Channel 1 event active
0bX0XX: Channel 2 event inactive
0bX1XX: Channel 2 event active
0b0XXX: Channel 3 event inactive
0b1XXX: Channel 3 event active

Write:

0bXXX0: No effect
0bXXX1: Set event on Channel 0
0bXX0X: No effect
0bXX1X: Set event on Channel 1
0bX0XX: No effect
0bX1XX: Set event on Channel 2
0b0XXX: No effect
0b1XXX: Set event on Channel 3

### 32.7.4 CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018

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. | APPCLEAR[3:0] | | | |
| | | | | | | | | | | | | w | | | |

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

Bit 3:0 **APPCLEAR[3:0]**: Clear channel event.
0b0000: No effect
0bXXX1: Clear event on Channel 0
0bXX1X: Clear event on Channel 1
0bX1XX: Clear event on Channel 2
0b1XXX: Clear event on Channel 3

### 32.7.5 CTI application pulse register (CTI_APPPULSER)

Address offset: 0x01C

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. | APPPULSE[3:0] | | | |
| | | | | | | | | | | | | w | | | |

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

Bit 3:0 **APPPULSE[3:0]**: Pulse channel event. This register clears itself immediately.
0b0000: No effect
0bXXX1: Generate pulse on Channel 0
0bXX1X: Generate pulse on Channel 1
0bX1XX: Generate pulse on Channel 2
0b1XXX: Generate pulse on Channel 3

### 32.7.6 CTI trigger In x enable register (CTI_INENRx)

Address offset: 0x020 + 4 * x, where x = 0 to 7

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. | TRIGINEN[3:0] | | | |
| | | | | | | | | | | | | rw | | | |

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

Bit 3:0 **TRIGINEN[3:0]**: Enables or disables a cross trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

0b0000: Trigger does not generate events on Channels
0bXXX1: Trigger n generates events on Channel 0
0bXX1X: Trigger n generates events on Channel 1
0bX1XX: Trigger n generates events on Channel 2
0b1XXX: Trigger n generates events on Channel 3

### 32.7.7 CTI trigger out x enable register (CTI_OUTENRx)

Address offset: 0x0A0 + 4 * x, where x = 0 to 7

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. | TRIGOUTEN[3:0] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bit 3:0 **TRIGOUTEN[3:0]**: For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

0b0000: Channel events do not generate triggers on Trigger outputs
0bXXX1: Channel 0 events generate triggers on Trigger output n
0bXX1X: Channel 1 events generate triggers on Trigger output n
0bX1XX: Channel 2 events generate triggers on Trigger output n
0b1XXX: Channel 3 events generate triggers on Trigger output n

### 32.7.8 CTI trigger in status register (CTI_TRGISTSR)

Address offset: 0x130

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. | TRIGINSTATUS[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit 7:0 **TRIGINSTATUS[7:0]**: Trigger input status.

There is one bit of the register for each CTITRIGIN input. When a bit is set to 1 it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

### 32.7.9 CTI trigger out status register (CTI_TRGOSTSR)

Address offset: 0x134

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. | TRIGOUTSTATUS[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit 7:0 **TRIGOUTSTATUS[7:0]**: Trigger output status.

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1 it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

### 32.7.10 CTI channel in status register (CTI_CHINSTSR)

Address offset: 0x138

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. | CHINSTATUS[3:0] | | | |
| | | | | | | | | | | | | r | r | r | r |

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

Bit 3:0 **CHINSTATUS[3:0]**: Channel input status.

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

### 32.7.11 CTI channel out status register (CTI_CHOUTSTSR)

Address offset: 0x13C

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. | CHOUTSTATUS[3:0] | | | |
| | | | | | | | | | | | | r | r | r | r |

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

Bit 3:0 **CHOUTSTATUS[3:0]**: Channel output status.

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

### 32.7.12 CTI channel gate register (CTI_GATER)

Address offset: 0x140

Reset value: 0x0000 000F

| 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. | GATEEN[3:0] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bit 3:0 **GATEEN[3:0]**: Channel output enable. For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.

0b0000: Channels events do not propagate
0bXXX1: Channel 0 events propagate
0bXX1X: Channel 1 events propagate
0bX1XX: Channel 2 events propagate
0b1XXX: Channel 3 events propagate

### 32.7.13 CTI claim tag set register (CTI_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

| 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. | CLAIMSET[3:0] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits
Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3
Read:
0xF: Indicates there are four bits in claim tag

### 32.7.14 CTI claim tag clear register (CTI_CLAIMCLR)

Address offset: 0xFA4

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. | CLAIMCLR[3:0] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3
Read: Returns current value of claim tag

### 32.7.15 CTI lock access register (CTI_LAR)

Address offset: 0xFB0

Reset value: N/A

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ACCESS_W[31:16] | | | | | | | | | | | | | | | |
| 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 |
| ACCESS_W[15:0] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **ACCESS_W[31:0]**: Enables write access to some CTI registers by processor cores (debuggers do not need to unlock the component)

0xC5AC CE55: Write access enabled

Other values: Write access disabled

### 32.7.16 CTI lock status register (CTI_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

| 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. | LOCK TYPE | LOCK GRANT | LOCK EXIST |
| | | | | | | | | | | | | | r | r | r |

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

Bit 2 **LOCKTYPE**: Indicates the size of the CTI_LAR register

0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock. This bit always reads as zero by an external debugger.

0: Write access is permitted

1: Write access is blocked. Only reads are permitted.

Bit 0 **LOCKEXIST**: Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.

0: No lock control mechanism exists.

1: Lock control mechanism is implemented

### 32.7.17 CTI authentication status register (CTI_AUTHSTATR)

Address offset: 0xFB8

Reset value: 0x0000 000A

| 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. | SNID[1:0] | | SID[1:0] | | NSNID[1:0] | | NSID[1:0] | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug

0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

### 32.7.18 CTI device configuration register (CTI_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0004 0800

| 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. | NUMCH[3:0] | | | |
| | | | | | | | | | | | | r | r | r | r |

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

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

Bits 19:16 **NUMCH[3:0]**: Number of ECT channels available
0x4: 4 channels

Bits 15:8 **NUMTRIG[7:0]**: Number of ECT triggers available
0x8: 8 trigger inputs and 8 trigger outputs

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

Bits 4:0 **EXTMUXNUM[4:0]**: Number of trigger input/output multiplexers
0x0: None

### 32.7.19 CTI device type identifier register (CTI_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0014

| 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. | SUBTYPE[3:0] | | | | MAJORTYPE[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Indicates that this component is a cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight™ SoC-400 system.

### 32.7.20 CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

### 32.7.21 CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0006

| 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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x06: CTI part number

### 32.7.22 CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

| 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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x9: CTI part number

### 32.7.23 CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 004B

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x4: r0p5

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

### 32.7.24 CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.7.25 CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.7.26 CTI CoreSight peripheral identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.7.27 CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 32.7.28 CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

## 32.7.29 CTI register map and reset values

**Table 199. CTI register map and reset values**

| Offset | Register name | 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 | CTI_CONTROLR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | GLBEN |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 0x010 | CTI_INTACKR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | INTACK[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x014 | CTI_APPSETR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | APPSET[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x018 | CTI_APPCLEAR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | APPCLEAR [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x01C | CTI_APPPULSER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | APPPULSE [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x020 | CTI_INENR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGINEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x024 | CTI_INENR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGINEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x028 | CTI_INENR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGINEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x02C | CTI_INENR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGINEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x030 | CTI_INENR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGINEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x034 | CTI_INENR5 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGINEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x038 | CTI_INENR6 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGINEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x03C | CTI_INENR7 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGINEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x0A0 | CTI_OUTENR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGOUTEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x0S4 | CTI_OUTENR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGOUTEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x0S8 | CTI_OUTENR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGOUTEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |

**Table 199. CTI register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0SC | CTI_OUTENR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGOUTEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x0B0 | CTI_OUTENR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGOUTEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x0B4 | CTI_OUTENR5 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGOUTEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x0B8 | CTI_OUTENR6 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGOUTEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x0BC | CTI_OUTENR7 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGOUTEN [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x130 | CTI_TRIGISTSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGINSTATUS[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x134 | CTI_TRIGOSTSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGOUTSTATUS[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x138 | CTI_CHINSTSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CHISTATUS [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x13C | CTI_CHOUTSTSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CHOSTATUS [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0x140 | CTI_GATER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | GATEEN[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 1 |
| 0xFA0 | CTI_CLAIMSETR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLAIMSET [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 1 |
| 0xFA4 | CTI_CLAIMCLR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLAIMCLR [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0xFB0 | CTI_LAR | KEY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFB4 | CTI_LSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | NTT | SLK | SLI |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 1 | 1 |
| 0xFB8 | CTI_AUTHSTATR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SNID[1:0] | | SID[1:0] | | NSNID[1:0] | | NSID[1:0] | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFC8 | CTI_DEVIDR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | NUMCH[3:0] | | | | NUMTRIG[7:0] | | | | | | | | Res. | Res. | Res. | Res. | EXMUXNUM [4:0] | | | |
| | Reset value | | | | | | | | | | | | | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | | | | 0 | 0 | 0 | 0 | 0 |
| 0xFCC | CTI_DEVTYPER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SUB[3:0] | | | | MAJOR[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

**Table 199. CTI register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xFD0 | CTI_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106CON [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0xFE0 | CTI_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0xFE4 | CTI_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0xFE8 | CTI_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0xFEC | CTI_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | CTI_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | CTI_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE[ 11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0xFF8 | CTI_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | CTI_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Refer to *Section 32.9: CPU2 ROM tables* and to *Section 32.13: CPU1 ROM table* for the register boundary addresses.

## 32.8 Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the debug software) to:

- maintain the clock and power the CPU1 processor core when in low power modes (Sleep, Stop or Standby), CPU2 operation is not influenced in low power debug mode
- maintain the clock and power the system debug and trace components when in low power modes
- stop the clock to certain peripherals (watchdogs, timers, RTC) when either processor core is stopped in debug mode

The DBGMCU registers are not reset by a system reset, only by a power on reset. They are accessible to the debugger via the CPU1 AHB access port at base address 0xE0042000.

*Note: The DBGMCU is not a standard CoreSight™ component, consequently it does not appear in the CPU1 ROM table.*

### 32.8.1 DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x000

Reset value: 0x200X 6495

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REV_ID[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 |
| Res. | Res. | Res. | Res. | DEV_ID[11:0] | | | | | | | | | | | |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:16 **REV_ID[15:0]**: Revision
0x2001 = STM32WB50CG revision Y and STM32WB30CE revision A
0x2003 = STM32WB50CG/30CE revision X

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

Bits 11:0 **DEV_ID[11:0]**: Device ID
0x495: STM32WB50CG/30CE

### 32.8.2 DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | TRGOEN | 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. | TRACE _IOEN | Res. | Res. | DBG_ STANDBY | DBG_ STOP | DBG_ SLEEP |
| | | | | | | | | | | rw | | | rw | rw | rw |

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TRGOEN**: External trigger out/put enable. This bit controls the direction of the bi-directional trigger pin, TRIG_INOUT.

0: Input. TRIG_INOUT is connected to TRGIN.

1: Output. TRIG_INOUT is connected to TRGOUT.

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **TRACE_IOEN**: Trace port and clock enable. This bit enables the trace port clock, TRACECLK.

0: Disabled.

1: Enabled.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Allow debug of the CPU1 in STANDBY and SHUTDOWN modes, no influence on CPU2 operation.

0: Normal operation. All clocks are disabled and the device powered down automatically in STANDBY and SHUTDOWN modes.

1: Automatic clock stop/power down disabled. All active CPU1 clocks and oscillators continue to run during STANDBY and SHUTDOWN modes, and the device supply is maintained, allowing full CPU1 debug capability. On exit from STANDBY and SHUTDOWN modes, a device reset is performed.

*Note: On exit from STANDBY no power reset is performed, a system reset is generated, and the wakeup clock is not HSI16, but MSI when DBG_STANDBY is set.*

Bit 1 **DBG_STOP**: Allow debug of the CPU1 in STOP mode, no influence on CPU2 operation.

0: Normal operation. All CPU1 clocks are disabled automatically in STOP mode

1: Automatic clock stop disabled. All active clocks and oscillators continue to run during STOP mode, allowing full CPU1 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 CPU1 in SLEEP mode, no influence on CPU2 operation.

0: Normal operation. Processor clock is stopped automatically in SLEEP mode

1: Automatic clock stop disabled. CPU1 processor clock continue to run, resulting in full CPU1 debug capability

### 32.8.3 DBGMCU CPU1 APB1 peripheral freeze register 1 (DBGMCU_APB1FZR1)

Address offset: 0x03C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DBG_ LPTIM1_ STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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. | DBG_ IWDG_ STOP | DBG_ WWDG _STOP | DBG_ RTC_ STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_ TIM2_ STOP |
| | | | rw | rw | rw | | | | | | | | | | rw |

Bit 31 **DBG_LPTIM1_STOP**: LPTIM1 stop in CPU1 debug

0: Normal operation. LPTIM1 continues to operate while CPU1 is in debug mode
1: Stop in debug. LPTIM1 is frozen while CPU1 is in debug mode.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in CPU1 debug

0: Normal operation. I2C1 SMBUS timeout continues to operate while CPU1 is in debug mode
1: Stop in debug. I2C1 SMBUS timeout is frozen while CPU1 is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP**: IWDG stop in CPU1 debug

0: Normal operation. IWDG continues to operate while CPU1 is in debug mode
1: Stop in debug. IWDG is frozen while CPU1 is in debug mode.

Bit 11 **DBG_WWDG_STOP**: WWDG stop in CPU1 debug

0: Normal operation. WWDG continues to operate while CPU1 is in debug mode
1: Stop in debug. WWDG is frozen while CPU1 is in debug mode.

Bit 10 **DBG_RTC_STOP**: RTC stop in CPU1 debug

0: Normal operation. RTC continues to operate while CPU1 is in debug mode
1: Stop in debug. RTC is frozen while CPU1 is in debug mode.

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in CPU1 debug

0: Normal operation. TIM2 continues to operate while CPU1 is in debug mode
1: Stop in debug. TIM2 is frozen while CPU1 is in debug mode.

### 32.8.4 DBGMCU CPU2 APB1 peripheral freeze register 1 (DBGMCU_C2APB1FZR1)

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGRQ.

Address offset: 0x040

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DBG_ LPTIM1_ STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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. | DBG_ IWDG_ STOP | Res. | DBG _RTC_ STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_ TIM2_ STOP |
| | | | rw | | rw | | | | | | | | | | rw |

Bit 31 **DBG_LPTIM1_STOP**: LPTIM1 stop in CPU2 debug
0: Normal operation. LPTIM1 continues to operate while CPU2 is in debug mode
1: Stop in debug. LPTIM1 is frozen while CPU2 is in debug mode.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in CPU2 debug
0: Normal operation. I2C1 SMBUS timeout continues to operate while CPU2 is in debug mode
1: Stop in debug. I2C1 SMBUS timeout is frozen while CPU2 is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP**: IWDG stop in CPU2 debug
0: Normal operation. IWDG continues to operate while CPU2 is in debug mode
1: Stop in debug. IWDG is frozen while CPU2 is in debug mode.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **DBG_RTC_STOP**: RTC stop in CPU2 debug
0: Normal operation. RTC continues to operate while CPU2 is in debug mode
1: Stop in debug. RTC is frozen while CPU2 is in debug mode.

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in CPU2 debug
0: Normal operation. TIM2 continues to operate while CPU2 is in debug mode
1: Stop in debug. TIM2 is frozen while CPU2 is in debug mode.

### 32.8.5 DBGMCU CPU1 APB1 peripheral freeze register 2 (DBGMCU_APB1FZR2)

Address offset: 0x044

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. | DBG_ LPTIM2_ STOP | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | rw | | | | | |

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

Bit 5 **DBG_LPTIM2_STOP**: LPTIM2 stop in CPU1 debug

0: Normal operation. LPTIM2 continues to operate while CPU1 is in debug mode

1: Stop in debug. LPTIM2 is frozen while CPU1 is in debug mode.

Bits 7:0 Reserved, must be kept at reset value.

### 32.8.6 DBGMCU CPU2 APB1 peripheral freeze register 2 (DBGMCU_C2APB1FZR2)

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGRQ.

Address offset: 0x048

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. | DBG_ LPTIM2_ STOP | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | rw | | | | | |

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

Bit 5 **DBG_LPTIM2_STOP**: LPTIM2 stop in CPU2 debug

0: Normal operation. LPTIM2 continues to operate while CPU2 is in debug mode

1: Stop in debug. LPTIM2 is frozen while CPU2 is in debug mode.

Bits 7:0 Reserved, must be kept at reset value.

### 32.8.7 DBGMCU CPU1 APB2 peripheral freeze register (DBGMCU_APB2FZR)

Address offset: 0x04C

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. | DBG_ TIM17_ STOP | DBG_ TIM16_ STOP | Res. |
| | | | | | | | | | | | | | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | DBG_ TIM1_ STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | rw | | | | | | | | | | | |

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

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in CPU1 debug

0: Normal operation. TIM17 continues to operate while CPU1 is in debug mode
1: Stop in debug. TIM17 is frozen while CPU1 is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in CPU1 debug

0: Normal operation. TIM16 continues to operate while CPU1 is in debug mode
1: Stop in debug. TIM16 is frozen while CPU1 is in debug mode.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in CPU1 debug

0: Normal operation. TIM1 continues to operate while CPU1 is in debug mode
1: Stop in debug. TIM1 is frozen while CPU1 is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

### 32.8.8 DBGMCU CPU2 APB2 peripheral freeze register (DBGMCU_C2APB2FZR)

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGRQ.

Address offset: 0x050

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. | DBG_ TIM17_ STOP | DBG_ TIM16_ STOP | Res. |
| | | | | | | | | | | | | | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | DBG_ TIM1_ STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | rw | | | | | | | | | | | |

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

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in CPU2 debug

0: Normal operation. TIM17 continues to operate while CPU2 is in debug mode
1: Stop in debug. TIM17 is frozen while CPU2 is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in CPU2 debug

0: Normal operation. TIM16 continues to operate while CPU2 is in debug mode
1: Stop in debug. TIM16 is frozen while CPU2 is in debug mode.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in CPU2 debug

0: Normal operation. TIM1 continues to operate while CPU2 is in debug mode
1: Stop in debug. TIM1 is frozen while CPU2 is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

## 32.8.9 DBGMCU register map and reset values

**Table 200. DBGMCU register map and reset values**

| Offset | Register name | 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 | DBGMCU_IDCODE | REV_ID[15:0] | | | | | | | | | | | | | | | | Res. | Res. | Res. | Res. | DEV_ID[11:0] | | | | | | | | | | | |
| | Reset value | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0x004 | DBGMCU_CR | Res. | Res. | Res. | TRGOEN | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRACE_IOEN | Res. | Res. | DBG_STANDBY | DBG_STOP | DBG_SLEEP |
| | Reset value | | | | 0 | | | | | | | | | | | | | | | | | | | | | | | 0 | | | 0 | 0 | 0 |
| 0x008-0x038 | 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. |
| 0x03C | DBGMCU_ APB1FZR1 | DBG_LPTIM1_STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_I2C1_STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_IWDG_STOP | DBG_WWDG_STOP | DBG_RTC_STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_TIM2_STOP |
| | Reset value | 0 | | | | | | | | | | 0 | | | | | | | | | 0 | 0 | 0 | | | | | | | | | | 0 |
| 0x040 | DBGMCU_ C2APB1FZR1 | DBG_LPTIM1_STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_I2C1_STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_IWDG_STOP | Res. | DBG_RTC_STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_TIM2_STOP |
| | Reset value | 0 | | | | | | | | | | 0 | | | | | | | | | 0 | | 0 | | | | | | | | | | 0 |
| 0x044 | DBGMCU_ APB1FZR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_LPTIM2_STOP | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | |
| 0x048 | DBGMCU_ C2APB1FZR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_LPTIM2_STOP | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | |
| 0x04C | DBGMCU_ APB2FZR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_TIM17_STOP | DBG_TIM16_STOP | Res. | Res. | Res. | Res. | Res. | DBG_TIM1_STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | 0 | 0 | | | | | | 0 | | | | | | | | | | | |

**Table 200. DBGMCU register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x050 | DBGMCU_ C2APB2FZR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBG_TIM17_STOP | DBG_TIM16_STOP | Res. | Res. | Res. | Res. | Res. | DBG_TIM1_STOP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | | | | | | | | | | | | 0 | 0 | | | | | | 0 | | | | | | | | | | | |

Refer to *Section 32.8: Microcontroller debug unit (DBGMCU)* for the register boundary addresses.

## 32.9 CPU2 ROM tables

The ROM tables are CoreSight™ components that contain the base addresses of all the Coresight™ debug components accessible via the AHBD. These tables allow a debugger to discover the topology of the CoreSight™ system automatically.

There are two ROM tables in the CPU2 sub-system:

1. ROM1: CPU2 processor ROM table, pointed to by the BASE register in the CPU2 AHB-AP. It contains the base address pointers for the CTI, as well as for the CPU2 ROM table.
2. ROM2: CPU2 ROM table, containing pointers to the CPU2 System control space registers, which allow the debugger to identify the CPU core, as well as to the remaining CoreSight™ components in the CPU2 subsystem (PBU, DWT).

The CPU2 processor ROM table occupies a 4-Kbyte, 32-bit wide chunk of AHB address space, from 0xF0000000 to 0xF0000FFC.

**Table 201. CPU2 processor ROM table**

| Address in ROM table | Component name | Component base address | Component address offset | Size | Entry |
|---|---|---|---|---|---|
| 0xF0000000 | CPU2 ROM table | 0xE00FF000 | 0xF00FF000 | 4 KB | 0xF00FF003 |
| 0xF0000004 | CTI | 0xF0001000 | 0x00001000 | 4 KB | 0x00001003 |
| 0xF0000008 | Not used | - | - | - | 0x00002002 |
| 0xF000000C | Not used | - | - | - | 0x10000002 |
| 0xF0000010 | Top of table | - | - | - | 0x00000000 |
| 0xF000000C to 0xF0000FC8 | Reserved | - | - | - | 0x00000000 |
| 0xF0000FCC to 0xF0000FFC | ROM table registers | - | - | - | See *Table 203* |

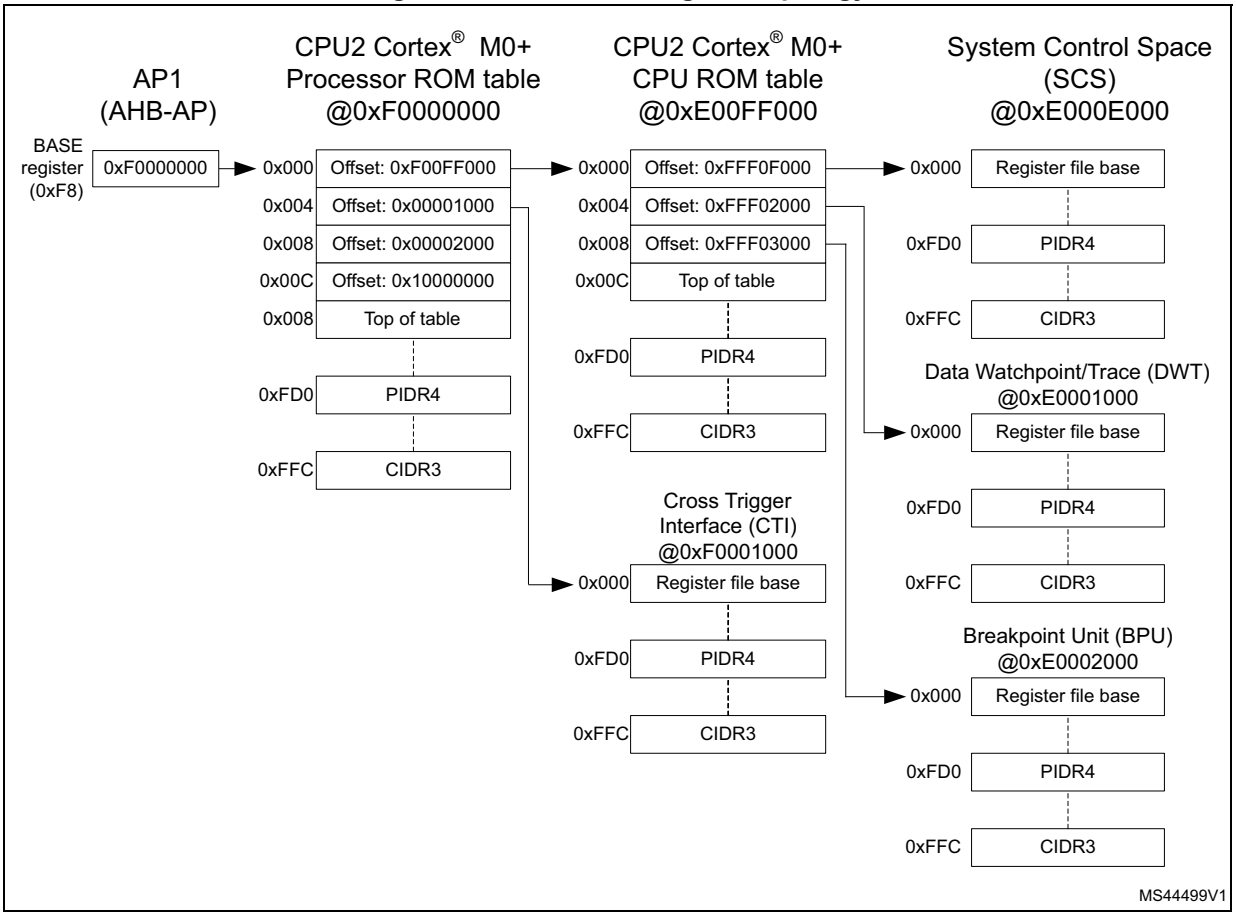
The CPU2 ROM table occupies a 4 KB, 32-bit wide chunk of APB-D address space, from 0xE00FF000 to 0xE00FFFFC.

**Table 202. CPU2 ROM table**

| Address in ROM table | Component name | Component base address | Component address offset | Size | Entry |
|---|---|---|---|---|---|
| 0xE00FF000 | SCS | 0xE000E000 | 0xFFF0F000 | 4 KB | 0xFFF0F003 |
| 0xE00FF004 | DWT | 0xE0001000 | 0xFFF02000 | 4 KB | 0xFFF02003 |
| 0xE00FF008 | BPU | 0xE0002000 | 0xFFF03000 | 4 KB | 0xFFF03003 |
| 0xE00FF00C | Top of table | - | - | - | 0x00000000 |
| 0xE00FF010 to 0xE00FFFC8 | Reserved | - | - | - | 0x00000000 |
| 0xE00FFFCC to 0xE00FFFFC | ROM table registers | - | - | - | See *Table 204* |

The topology for the CoreSight™ components in the CPU2 subsystem is shown in *Figure 350*.

**Figure 350. CPU2 CoreSight™ topology**



### 32.9.1 CPU2 ROM1 memory type register (C2ROM1_MEMTYPER)

Address offset: 0xFCC

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. | Res. | Res. | Res. | Res. | SYSMEM |
| | | | | | | | | | | | | | | | rw |

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

Bit 0 **SYSMEM**: System memory
0x1: System memory is present on this bus

### 32.9.2 CPU2 ROM1 CoreSight peripheral identity register 4 (C2ROM1_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC continuation code

### 32.9.3 CPU2 ROM1 CoreSight peripheral identity register 0 (C2ROM1_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0C0

| 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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0xC0: Cortex®-M0+ processor ROM table

### 32.9.4 CPU2 ROM1 CoreSight peripheral identity register 1 (C2ROM1_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

| 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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: Cortex®-M0+processor ROM table

### 32.9.5 CPU2 ROM1 CoreSight peripheral identity register 2 (C2ROM1_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

### 32.9.6 CPU2 ROM1 CoreSight peripheral identity register 3 (C2ROM1_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.9.7 CPU2 ROM1 CoreSight component identity register 0 (C2ROM1_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.9.8 CPU2 ROM1 CoreSight peripheral identity register 1 (C2ROM1_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.9.9 CPU2 ROM1 CoreSight component identity register 2 (C2ROM1_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 32.9.10 CPU2 ROM1 CoreSight component identity register 3 (C2ROM1_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

### 32.9.11 CPU2 processor ROM table registers and reset values

**Table 203. CPU2 processor ROM table register map and reset values**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xFCC | C2ROM1_ MEMTYPER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SYSMEM |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 |
| 0xFD0 | C2ROM1_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 4KCOUNT [3:0] | | | | JEP106CON [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0xFE0 | C2ROM1_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFE4 | C2ROM1_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0xFE8 | C2ROM1_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0xFEC | C2ROM1_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | C2ROM1_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | C2ROM1_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0xFF8 | C2ROM1_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | C2ROM1_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Refer to *Section 32.9: CPU2 ROM tables* for the register boundary addresses.

### 32.9.12 CPU2 ROM2 memory type register (C2ROM2_MEMTYPER)

Address offset: 0xFCC

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. | Res. | Res. | Res. | Res. | SYSMEM |
| | | | | | | | | | | | | | | | rw |

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

Bit 0 **SYSMEM**: System memory
0x1: System memory is present on this bus

### 32.9.13 CPU2 ROM2 CoreSight peripheral identity register 4 (C2ROM2_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC continuation code

### 32.9.14 CPU2 ROM2 CoreSight peripheral identity register 0 (C2ROM2_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0C0

| 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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0xC0: CPU2 ROM table

### 32.9.15 CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

| 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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: CPU2 ROM table

### 32.9.16 CPU2 ROM2 CoreSight peripheral identity register 2 (C2ROM2_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

### 32.9.17 CPU2 ROM2 CoreSight peripheral identity register 3 (C2ROM2_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.9.18 CPU2 ROM2 CoreSight component identity register 0 (C2ROM2_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.9.19 CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.9.20 CPU2 ROM2 CoreSight component identity register 2 (C2ROM2_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 32.9.21 CPU2 ROM2 CoreSight component identity register 3 (C2ROM2_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | 3r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

## 32.9.22 CPU2 ROM table register map and reset values

**Table 204. CPU2 ROM table register map and reset values**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xFCC | C2ROM2_ MEMTYPER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SYSMEM |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 |
| 0xFD0 | C2ROM2_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 4KCOUNT [3:0] | | | | JEP106CON[ 3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0xFE0 | C2ROM2_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFE4 | C2ROM2_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0xFE8 | C2ROM2_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0xFEC | C2ROM2_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | C2ROM2_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | C2ROM2_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0xFF8 | C2ROM2_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | C2ROM2_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Refer to *Section 32.9: CPU2 ROM tables* for the register boundary addresses.

## 32.10 CPU2 data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:

- Watchpoint
- PC sampling trigger
- Data address sampling trigger
- Data comparator (for comparator 1 only)
- Clock cycle counter comparator (for comparator 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares the value held in its DWT_COMPR register with one of the following items:

- a data address
- an instruction address
- a data value
- the cycle count value (comparator 0 only)

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT Data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value.
- A watchpoint debug event, on either the PC value or the accessed data address.
- A CMPMATCH[N] event, that signals the match outside the DWT unit.

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual *[5]*.

### 32.10.1 DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NUMCOMP[3:0]. | | | | NOTRCPKT | NOEXTTRIG | NOCYCCNT | NOPRFCNT | Res. | CYCEVTENA | FOLDEVTENA | LSUEVTENA | SLEEPEVTENA | EXCEVTENA | CPIEVTENA | EXCTRCENA |
| r | r | r | r | r | r | r | r | | 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. | PCSAMPLENA | SYNCTAP[1:0] | | CYCTAP | POSTINIT[3:0] | | | | POSTRESET[3:0] | | | | CYCCNTENA |
| | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:28 **NUMCOMP**: Number of comparators implemented (read only)
0x4: Four comparators

Bit 27 **NOTRCPKT**: Trace sampling and exception tracing support (read only)
0x0: Supported

Bit 26 **NOEXTTRIG**: External match signal, CMPMATCH support (read only)
0x0: Supported

Bit 25 **NOCYCCNT**: Cycle counter support (read only)
0x0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read only)
0x0: Supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CYCEVTENA**: Enable for POSTCNT underflow event counter packet generation
0x0: Disabled
0x1: Enabled

Bit 21 **FOLDEVTENA**: Enable for folded instruction counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 20 **LSUEVTENA**: Enable for LSU counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 19 **SLEEPEVTENA**: Enable for sleep counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 18 **EXCEVTENA**: Enable for exception overhead counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 17 **CPIEVTENA**: Enable for CPI counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 16 **EXCTRCENA**: Enable for exception trace generation
0x0: Disabled
0x1: Enabled

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

Bit 12 **PCSAMPLENA**: Enables use of POSTCNT counter as a timer for Periodic PC sample packet generation.

0x0: Disabled
0x1: Enabled

Bits 11:10 **SYNCTAP[1:0]**: Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the synchronization packet rate.

0x0: Disabled. No synchronization packets
0x1: Tap at CYCCNT[24]
0x2: Tap at CYCCNT[26]
0x3: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Selects the position of the POSTCNT tap on the CYCCNT counter.

0x0: Tap at CYCCNT[6]
0x1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter. Writes to this field are ignored if POSTCNT counter is enabled (ie. CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTPRESET[3:0]**: Reload value of the POSTCNT counter.

Bit 0 **CYCCNTENA**: Enables CYCCNT counter.

0x0: Disabled
0x1: Enabled

### 32.10.2 DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CYCCNT[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 |
| CYCCNT[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **CYCCNT[31:0]**: Processor clock cycle counter

### 32.10.3 DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

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. | CPICNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **CPICNT[7:0]**: CPI counter. Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNTR, and counts any instruction fetch stalls.

### 32.10.4 DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C

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. | EXCCNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter. Counts the number of cycles spent in exception processing.

### 32.10.5 DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

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. | SLEEPCNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **SLEEPCNT[7:0]**: Sleep cycle counter. Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

### 32.10.6 DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

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. | LSUCNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **LSUCNT[7:0]**: Load store counter. Counts additional cycles required to execute load and store instructions.

### 32.10.7 DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

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. | FOLDCNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **FOLDCNT[7:0]**: Folded instruction counter. Increments on each instruction that takes 0 cycles.

### 32.10.8 DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EIASAMPLE[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EIASAMPLE[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **EIASAMPLE[31:0]**: Executed Instruction Address sample value. Samples the current value of the program counter.

### 32.10.9 DWT comparator register x (DWT_COMPxR)

Address offset: 0x020 + x * 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COMP[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 |
| COMP[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **COMP[31:0]**: Reference value for comparison.

### 32.10.10 DWT mask register x (DWT_MASKxR)

Address offset: 0x024 + x * 0x10 (for x = 0 to 3)

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. | MASK[4:0] | | | | |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

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

Bits 4:0 **MASK[4:0]**: Comparator mask size. Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

### 32.10.11 DWT function register x (DWT_FUNCTxR)

Address offset: 0x028 + x * 0x10 (for x = 0 to 3)

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. | MATCHED | Res. | Res. | Res. | Res. | DATAVADDR1[3:0] | | | |
| | | | | | | | r | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATAVADDR0[3:0] | | | | DATAVSIZE[1:0] | | LINK1 ENA | DATAV MATCH | CYC MATCH | Res. | EMIT RANGE | Res. | FUNCTION[3:0] | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | | rw | rw | rw | rw |

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

Bit 24 **MATCHED**: Comparator match (read only). Indicates if a comparator match has occurred since the register was last read.

0: No match
1: Match occurred

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **DATAVADDR1[3:0]**: When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12 **DATAVADDR0[3:0]**: When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10 **DATAVSIZE[1:0]**: For data value matching, specifies the size of the required data comparison.

0x0: Byte
0x1: Half word
0x2: Word
0x3: reserved

Bit 9 **LNK1ENA**: Indicates whether use of a second linked comparator is supported (read only).

0x1: Supported

Bit 8 **DATAVMATCH**: Enables cycle comparison.

0x0: Perform address comparison
0x1: Perform data value comparison

Bit 7 **CYCMATCH**: Enables cycle count comparison on comparator 0. This field is reserved for other comparators.

0x0: No cycle count comparison
0x1: Compare DWT_COMP0R with the cycle counter, DWT_CYCCNTR

Bit 6 Reserved, must be kept at reset value.

Bit 5 **EMITRANGE**: Enables generation of data trace address offset packets (containing data address bits 0 to 15)

0x0: Disabled
0x1: Enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]**: Selects action to take on comparator match. The meaning of this bit field depends on the setting of the DATAVMATCH and CYCMATCH fields. See *[5]*.

### 32.10.12 DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

### 32.10.13 DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 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. | 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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x02: DWT part number

### 32.10.14 DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

| 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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x9: DWT part number

### 32.10.15 DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

### 32.10.16 DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.10.17 DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.10.18 DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.10.19 DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

## 32.10.20 DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

### 32.10.21 CPU2 DWT registers

The CPU2 DWT registers are located at address range 0xE0001000 to 0xE0001FFC, on the AHBD.

**Table 205. CPU2 DWT register map and reset values**

| Offset | Register name | 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 | DWT_CTRLR | NUMCOMP[3:0] | | | | NOTRCPKT | NOEXTTRIG | NOCYCCNT | NOPRFCNT | Res. | CYCEVTENA | FOLDEVTENA | LSUEVTENA | SLEEPEVTENA | EXCEVTENA | CPIEVTENA | EXCTRCENA | Res. | Res. | Res. | PCSAMPLENA | SYNCTAP[1:0] | | CYCTAP | POSINIT[3:0] | | | | POSTPRESET[3:0] | | | CYCCNTENA |
| | Reset value | 0 | 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 | 0 |
| 0x004 | DWT_CYCCNTR | CYCCNT[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 |
| 0x008 | DWT_CPICNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CPICNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00C | DWT_EXCCNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EXCCNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x010 | DWT_SLPCNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SLEEPCNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x014 | DWT_LSUCNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LSUCNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x018 | DWT_FOLDCNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | FOLDCNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x01C | DWT_PCSR | EIASAMPLE[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 | DWT_COMP0R | COMP[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 | DWT_MASK0R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MASK[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| 0x028 | DWT_FUNCT0R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MATCHED | Res. | Res. | Res. | Res. | DATAVADDR1[3:0] | | | | DATAVADDR0[3:0] | | | | DATAVSIZE[1:0] | | LNK1ENA | DATAVMATCH | CYCMATCH | Res. | EMITRANGE | Res. | FUNCTION[3:0] | | | |
| | Reset value | | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 |
| 0x030 | DWT_COMP1R | COMP[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 |
| 0x034 | DWT_MASK1R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MASK[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |

**Table 205. CPU2 DWT register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x038 | DWT_FUNCT1R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MATCHED | Res. | Res. | Res. | q | DATAVADDR1[3:0] | | | | DATAVADDR0[3:0] | | | | DATAVSIZE[1:0] | | LNK1ENA | DATAVMATCH | CYCMATCH | Res. | EMITRANGE | Res. | FUNCTION[3:0] | | | |
| | Reset value | | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 |
| 0x040 | DWT_COMP2R | COMP[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 |
| 0x044 | DWT_MASK2R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MASK[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| 0x048 | DWT_FUNCT2R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MATCHED | Res. | Res. | Res. | Res. | DATAVADDR1[3:0] | | | | DATAVADDR0[3:0] | | | | DATAVSIZE[1:0] | | LNK1ENA | DATAVMATCH | CYCMATCH | Res. | EMITRANGE | Res. | FUNCTION[3:0] | | | |
| | Reset value | | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 |
| 0x050 | DWT_COMP3R | COMP[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 | DWT_MASK3R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MASK[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| 0x058 | DWT_FUNCT3R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MATCHED | Res. | Res. | Res. | Res. | DATAVADDR1[3:0] | | | | DATAVADDR0[3:0] | | | | DATAVSIZE[1:0] | | LNK1ENA | DATAVMATCH | CYCMATCH | Res. | EMITRANGE | Res. | FUNCTION[3:0] | | | |
| | Reset value | | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 |
| 0xFD0 | DWT_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 4KCOUNT [3:0] | | | | JEP106CON [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0xFE0 | DWT_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0xFE4 | DWT_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0xFE8 | DWT_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0xFEC | DWT_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | DWT_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | DWT_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE[ 11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

**Table 205. CPU2 DWT register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xFF8 | DWT_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | DWT_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Refer to *Section 32.9: CPU2 ROM tables* for the register boundary addresses.

## 32.11 CPU2 breakpoint unit (PBU)

The BPU allows the user to set hardware breakpoints. It contains eight comparators that monitor the instruction fetch address and return a breakpoint instruction when a match is detected.The CPU2 PBU does not support Flash memory patch functionality.

### 32.11.1 BPU control register (BPU_CTRLR)

Address offset: 0x000

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. | NUM_CODE[6:4] | | | NUM_LIT[3:0] | | | | NUM_CODE[3:0] | | | | Res. | Res. | KEY | ENABLE |
| | r | r | r | r | r | r | r | r | r | r | r | | | rw | rw |

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

Bits 14:12 **NUM_CODE[6:4]**: Number of instruction address comparators supported - least significant bits (read only).

0x0: 8 instruction comparators supported.

Bits 11:8 **NUM_LIT[3:0]**: Number of literal address comparators supported (read only).

0x0: No literal comparators supported.

Bits 7:4 **NUM_CODE[3:0]**: Number of instruction address comparators supported - least significant bits (read only).

0x8: 8 instruction comparators supported

Bit 1 **KEY**: Write protect key. A write to BPU_CTRLR register is ignored if this bit is not set to 1.

Bits 0 **ENABLE**: BPU enable

0x0: Disable

0x1: Enable

### 32.11.2 BPU remap register (BPU_REMAPR)

Address offset: 0x004

Reset value: 0x0000 0000

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

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

Bit 29 **RMPSPT**: Indicates whether Flash memory patch remap is supported (read only).
0x0: Remapping not supported.

Bits 28:0 Reserved, must be kept at reset value.

### 32.11.3 BPU comparator registers (BPU_COMPxR)

Address offset: 0x008 + x * 0x4 (for x = 0 to 7)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REPLACE[1:0] | | Res. | COMP[26:14] | | | | | | | | | | | | |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COMP[13:0] | | | | | | | | | | | | | | Res. | ENABLE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw |

Bits 31:30 **REPLACE[1:0]**: Defines the behavior when a match occurs between the COMP field and the instruction fetch address.
0x0: Reserved
0x1: Breakpoint on lower half-word, upper half-word is unaffected.
0x2: Breakpoint on upper half-word, lower half-word is unaffected.
0x3: Breakpoint on both upper and lower half-words.

Bit 29 Reserved, must be kept at reset value.

Bits 28:2 **COMP[26:0]**: Value to compare with address bits 28:2 of accesses to instruction code memory (0x00000000 to 0x1FFFFFFF). If a match occurs, the action to be taken is defined by the REPLACE field.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Comparator enable. The comparator is only enabled if both this bit and the BPU ENABLE bit in the BPU_CTRLR register are set.
0: Disabled
1: Enabled

### 32.11.4 BPU CoreSight peripheral identity register 4 (BPU_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

### 32.11.5 BPU CoreSight peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0

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. | 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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x0C: BPU part number

### 32.11.6 BPU CoreSight peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

| 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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x0: BPU part number

### 32.11.7 BPU CoreSight peripheral identity register 2 (BPU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x2: r0p3

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

### 32.11.8 BPU CoreSight peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.11.9 BPU CoreSight component identity register 0 (BPU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.11.10 BPU CoreSight peripheral identity register 1 (BPU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.11.11 BPU CoreSight component identity register 2 (BPU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 32.11.12 BPU CoreSight component identity register 3 (BPU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

### 32.11.13 CPU2 BPU register map and reset values

The CPU2 BPU registers are located at address range 0xE0002000 to 0xE0002FFC.

**Table 206. CPU2 BPU register map and reset values**

| Offset | Register name | 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 | BPU_CTRLR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | NUM_CODE[6:4] | | | NUM_LIT[3:0] | | | | NUM_CODE[3:0] | | | | Res. | Res. | KEY | ENABLE |
| | Reset value | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | | | 0 | 0 |
| 0x004 | BPU_REMAPR | Res. | Res. | RMPSP | Res. | Res. | Res. | Res. | Res. | 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 | | | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x008 to 0x024 | BPU_COMP0-7R | REPLACE[1:0] | | Res. | COMP[26:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | Res. | ENABLE |
| | 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 |
| 0xFD0 | BPU_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 4KCOUNT [3:0] | | | | JEP106CON [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0xFE0 | BPU_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0xFE4 | BPU_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0xFE8 | BPU_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0xFEC | BPU_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | BPU_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | BPU_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0xFF8 | BPU_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | BPU_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Refer to *Section 32.9: CPU2 ROM tables* for the register boundary addresses.

## 32.12 CPU2 cross trigger interface (CTI)

See *Section 32.6*.

## 32.13 CPU1 ROM table

The ROM table is a CoreSight™ component that contains the base addresses of all the Coresight™ debug components accessible via the AHB-AP. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There is one ROM table in the CPU1 sub-system. This table is pointed to by the BASE register in the CPU1 AHB-AP. It contains the base address pointer for the System control space registers, which allows the debugger to identify the CPU core, as well as for the FPB, DWT, ITM and CTI.

The CPU1 ROM table (see *Table 207*) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00FF000 to 0xE00FFFFC.

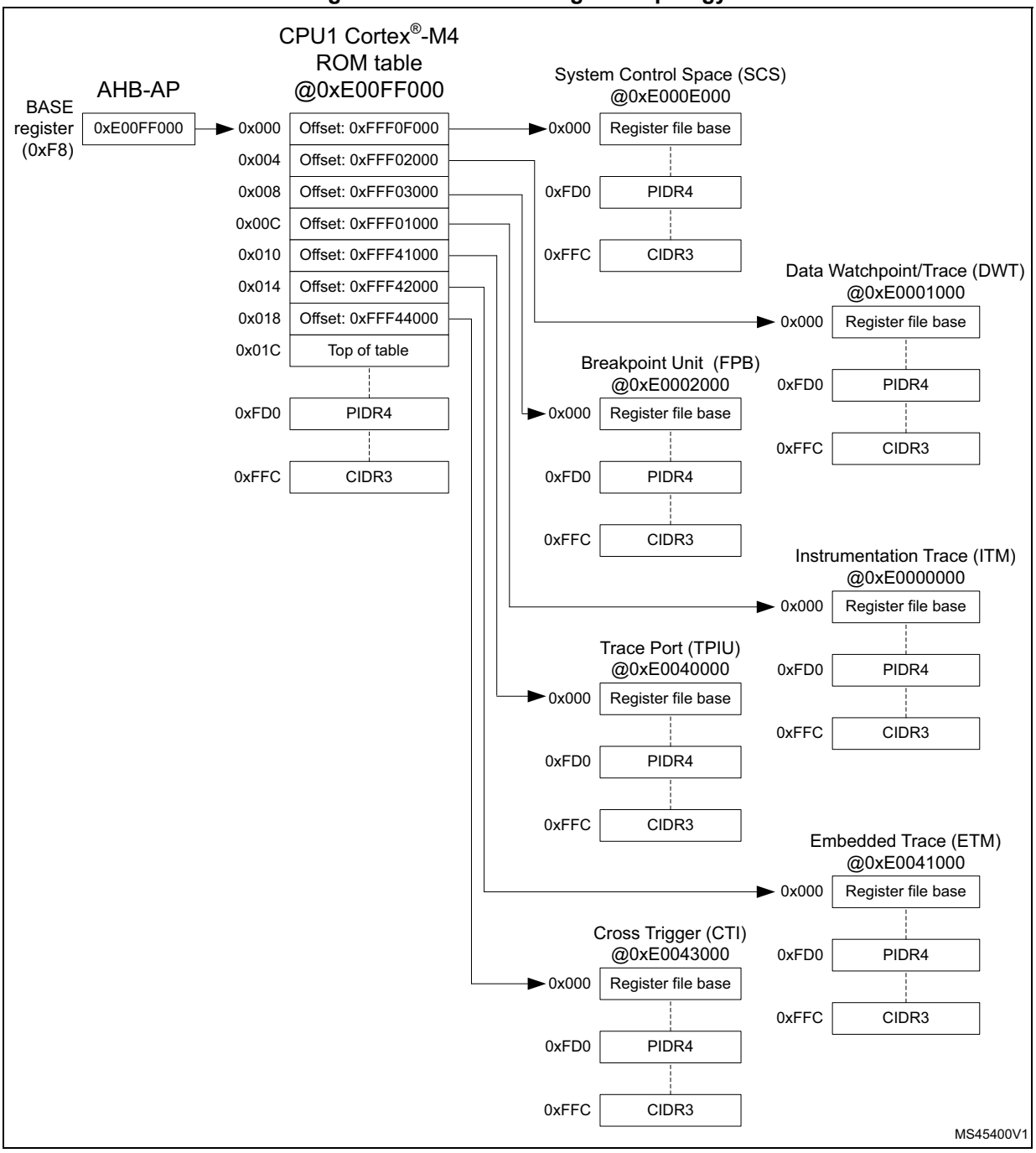
**Table 207. CPU1 ROM table**

| Address in ROM table | Component name | Component base address | Component address offset | Size | Entry |
|---|---|---|---|---|---|
| 0xE00FF000 | SCS | 0xE000E000 | 0xFFF0F000 | 4 KB | 0xFFF0F003 |
| 0xE00FF004 | DWT | 0xE0001000 | 0xFFF02000 | 4 KB | 0xFFF02003 |
| 0xE00FF008 | FPB | 0xE0002000 | 0xFFF03000 | 4 KB | 0xFFF03003 |
| 0xE00FF00C | ITM | 0xE0000000 | 0xFFF01000 | 4 KB | 0xFFF01003 |
| 0xE00FF010 | TPIU | 0xE0040000 | 0xFFF41000 | 4 KB | 0xFFF41003 |
| 0xE00FF014 | ETM(1) | 0xE0041000 | 0xFFF42000 | 4 KB | 0xFFF42003 |
| 0xE00FF018 | CTI | 0xE0043000 | 0xFFF44000 | 4 KB | 0xFFF44003 |
| 0xE00FF01C | Top of table | - | - | - | 0x00000000 |
| 0xE00FF020 to 0xE00FFFC8 | Reserved | - | - | - | 0x00000000 |
| 0xE00FFFCC to 0xE00FFFFC | ROM table registers | - | - | - | See *Table 203* |

1. Visible in ROM table but not at product level.

The topology for the CoreSight™ components in the CPU1 subsystem is shown in *Figure 351*.

**Figure 351. CPU1 CoreSight™ topology**




1. Visible in ROM table but not at product level.

## 32.13.1 CPU1 ROM memory type register (C1ROM_MEMTYPER)

Address offset: 0xFCC

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. | Res. | Res. | Res. | Res. | SYSMEM |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **SYSMEM**: System memory
0x1: System memory is present on this bus

### 32.13.2 CPU1 ROM CoreSight peripheral identity register 4 (C1ROM_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

### 32.13.3 CPU1 ROM CoreSight peripheral identity register 0 (C1ROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0095

| 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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x95: STM32WB50CG/30CE

### 32.13.4 CPU1 ROM CoreSight peripheral identity register 1 (C1ROM_PIDR1)

Address offset: 0xFE4

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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: STM32WB50CG/30CE

### 32.13.5 CPU1 ROM CoreSight peripheral identity register 2 (C1ROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

### 32.13.6 CPU1 ROM CoreSight peripheral identity register 3 (C1ROM_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.13.7 CPU1 ROM CoreSight component identity register 0 (C1ROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.13.8 CPU1 ROM CoreSight peripheral identity register 1 (C1ROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.13.9 CPU1 ROM CoreSight component identity register 2 (C1ROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 32.13.10 CPU1 ROM CoreSight component identity register 3 (C1ROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

### 32.13.11 CPU1 ROM table register map and reset values

**Table 208. CPU1 ROM table register map and reset values**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xFCC | C1ROM_ MEMTYPER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SYSMEM |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 |
| 0xFD0 | C1ROM_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 4KCOUNT [3:0] | | | | JEP106CON [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFE0 | C1ROM_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0xFE4 | C1ROM_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0xFE8 | C1ROM_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0xFEC | C1ROM_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | C1ROM_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | C1ROM_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0xFF8 | C1ROM_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | C1ROM_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Refer to *Section 32.13: CPU1 ROM table* for the register boundary addresses.

## 32.14 CPU1 data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:

- Watchpoint
- PC sampling trigger
- Data address sampling trigger
- Data comparator (comparator 1 only)
- Clock cycle counter comparator (comparator 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares the value held in its DWT_COMPR register with one of the following:

- a data address
- an instruction address
- a data value
- the cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT Data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value.
- A watchpoint debug event, on either the PC value or the accessed data address.
- A CMPMATCH[N] event, that signals the match outside the DWT unit.

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual *[5]*.

### 32.14.1 DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NUMCOMP[3:0]. | | | | NOTRCPKT | NOEXTTRIG | NOCYCCNT | NOPRFCNT | Res. | CYCEVTENA | FOLDEVTENA | LSUEVTENA | SLEEPEVTENA | EXCEVTENA | CPIEVTENA | EXCTRCENA |
| r | r | r | r | r | r | r | r | | 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. | PCSAMPLENA | SYNCTAP[1:0] | | CYCTAP | POSTINIT[3:0] | | | | POSTRESET[3:0] | | | | CYCCNTENA |
| | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:28 **NUMCOMP[3:0]**: Number of comparators implemented (read only)
0x4: Four comparators

Bit 27 **NOTRCPKT**: Trace sampling and exception tracing support (read only)
0x0: Supported

Bit 26 **NOEXTTRIG**: External match signal, CMPMATCH support (read only)
0x0: Supported

Bit 25 **NOCYCCNT**: Cycle counter support (read only)
0x0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read only)
0x0: Supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CYCEVTENA**: Enable for POSTCNT underflow event counter packet generation
0x0: Disabled
0x1: Enabled

Bit 21 **FOLDEVTENA**: Enable for folded instruction counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 20 **LSUEVTENA**: Enable for LSU counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 19 **SLEEPEVTENA**: Enable for sleep counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 18 **EXCEVTENA**: Enable for exception overhead counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 17 **CPIEVTENA**: Enable for CPI counter overflow event generation
0x0: Disabled
0x1: Enabled

Bit 16 **EXCTRCENA**: Enable for exception trace generation
0x0: Disabled
0x1: Enabled

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

Bit 12 **PCSAMPLENA**: Enables use of POSTCNT counter as a timer for Periodic PC sample packet generation.

0x0: Disabled
0x1: Enabled

Bits 11:10 **SYNCTAP[1:0]**: Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the synchronization packet rate.

0x0: Disabled. No synchronization packets
0x1: Tap at CYCCNT[24]
0x2: Tap at CYCCNT[26]
0x3: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Selects the position of the POSTCNT tap on the CYCCNT counter.

0x0: Tap at CYCCNT[6]
0x1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter. Writes to this field are ignored if POSTCNT counter is enabled (ie. CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTPRESET[3:0]**: Reload value of the POSTCNT counter.

Bit 0 **CYCCNTENA**: Enables CYCCNT counter.

0x0: Disabled
0x1: Enabled

### 32.14.2 DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CYCCNT[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 |
| CYCCNT[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **CYCCNT[31:0]**: Processor clock cycle counter

### 32.14.3 DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

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. | CPICNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **CPICNT[7:0]**: CPI counter. Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNTR, and counts any instruction fetch stalls.

### 32.14.4 DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C

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. | EXCCNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter. Counts the number of cycles spent in exception processing.

### 32.14.5 DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

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. | SLEEPCNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **SLEEPCNT[7:0]**: Sleep cycle counter. Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

### 32.14.6 DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

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. | LSUCNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **LSUCNT[7:0]**: Load store counter. Counts additional cycles required to execute load and store instructions.

### 32.14.7 DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

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. | FOLDCNT[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **FOLDCNT[7:0]**: Folded instruction counter. Increments on each instruction that takes 0 cycles.

### 32.14.8 DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EIASAMPLE[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EIASAMPLE[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **EIASAMPLE[31:0]**: Executed instruction address sample value. Samples the current value of the program counter.

### 32.14.9 DWT comparator register x (DWT_COMPxR)

Address offset: 0x020 + x * 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COMP[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 |
| COMP[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **COMP[31:0]**: Reference value for comparison.

### 32.14.10 DWT mask register x (DWT_MASKxR)

Address offset: 0x024 + x * 0x10 (for x = 0 to 3)

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. | MASK[4:0] | | | | |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

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

Bits 4:0 **MASK[4:0]**: Comparator mask size. Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

### 32.14.11 DWT function register x (DWT_FUNCTxR)

Address offset: 0x028 + x * 0x10 (for x = 0 to 3)

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. | DATAVADDR1[3:0] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATAVADDR0[3:0] | | | | DATAVSIZE[1:0] | | LINK1 ENA | DATAV MATCH | CYC MATCH | Res. | EMIT RANGE | Res. | FUNCTION[3:0] | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | | rw | rw | rw | rw |

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

Bit 24 **MATCHED**: Comparator match (read only). Indicates if a comparator match has occurred since the register was last read.

0: No match
1: Match occurred

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **DATAVADDR1[3:0]**: When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12 **DATAVADDR0[3:0]**: When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10 **DATAVSIZE[1:0]**: For data value matching, specifies the size of the required data comparison.

0x0: Byte
0x1: Half word
0x2: Word
0x3: Reserved

Bit 9 **LNK1ENA**: Indicates whether use of a second linked comparator is supported (read only).

0x1: Supported

Bit 8 **DATAVMATCH**: Enables cycle comparison.

0x0: Perform address comparison
0x1: Perform data value comparison

Bit 7 **CYCMATCH**: Enables cycle count comparison on comparator 0. This field is reserved for other comparators.

0x0: No cycle count comparison
0x1: Compare DWT_COMP0R with the cycle counter, DWT_CYCCNTR

Bit 6 Reserved, must be kept at reset value.

Bit 5 **EMITRANGE**: Enables generation of data trace address offset packets (containing data address bits 0 to 15)

0x0: Disabled
0x1: Enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]**: Selects action to take on comparator match. The meaning of this bit field depends on the setting of the DATAVMATCH and CYCMATCH fields. See *[5]*.

### 32.14.12 DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

### 32.14.13 DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 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. | 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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x02: DWT part number

### 32.14.14 DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

| 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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x9: DWT part number

### 32.14.15 DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

### 32.14.16 DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.14.17 DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.14.18 DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.14.19 DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 32.14.20 DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

## 32.14.21 CPU1 DWT register map and reset values

The CPU1 DWT registers are located at address range 0xE0001000 to 0xE0001FFC.

**Table 209. CPU1 DWT register map and reset values**

| Offset | Register name | 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 | DWT_CTRLR | NUMCOMP [3:0] | | | | NOTRCPKT | NOEXTTRIG | NOCYCCNT | NOPRFCNT | Res. | CYCEVTENA | FOLDEVTENA | LSUEVTENA | SLEEPEVTENA | EXCEVTENA | CPIEVTENA | EXCTRCENA | Res. | Res. | Res. | PCSAMPLENA | SYNCTAP [1:0] | | CYCTAP | POSINIT[3:0] | | | | POSTPRESET[3:0] | | | CYCCNTENA |
| | Reset value | 0 | 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 | 0 |
| 0x004 | DWT_CYCCNTR | CYCCNT[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 |
| 0x008 | DWT_CPICNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CPICNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00C | DWT_EXCCNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EXCCNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x010 | DWT_SLPCNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SLEEPCNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x014 | DWT_LSUCNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LSUCNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x018 | DWT_FOLDCNTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | FOLDCNT[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x01C | DWT_PCSR | EIASAMPLE[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 | DWT_COMP0R | COMP[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 | DWT_MASK0R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MASK[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| 0x028 | DWT_FUNCT0R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MATCHED | Res. | Res. | Res. | Res. | DATAVADDR1 [3:0] | | | | DATAVADDR0 [3:0] | | | | DATAVSIZE [1:0] | | LNK1ENA | DATAVMATCH | CYCMATCH | Res. | EMITRANGE | Res. | FUNCTION [3:0] | | | |
| | Reset value | | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 |
| 0x030 | DWT_COMP1R | COMP[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 |
| 0x034 | DWT_MASK1R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MASK[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| 0x038 | DWT_FUNCT1R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MATCHED | Res. | Res. | Res. | Res. | DATAVADDR1 [3:0] | | | | DATAVADDR0 [3:0] | | | | DATAVSIZE [1:0] | | LNK1ENA | DATAVMATCH | CYCMATCH | Res. | EMITRANGE | Res. | FUNCTION [3:0] | | | |
| | Reset value | | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 |

**Table 209. CPU1 DWT register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x040 | DWT_COMP2R | COMP[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 |
| 0x044 | DWT_MASK2R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MASK[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| 0x048 | DWT_FUNCT2R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MATCHED | Res. | Res. | Res. | Res. | DATAVADDR1[3:0] | | | | DATAVADDR0[3:0] | | | | DATAVSIZE[1:0] | | LNK1ENA | DATAVMATCH | CYCMATCH | Res. | EMITRANGE | Res. | FUNCTION[3:0] | | | |
| | Reset value | | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 |
| 0x050 | DWT_COMP3R | COMP[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 | DWT_MASK3R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MASK[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| 0x058 | DWT_FUNCT3R | Res. | Res. | Res. | Res. | Res. | Res. | Res. | MATCHED | Res. | Res. | Res. | Res. | DATAVADDR1 [3:0] | | | | DATAVADDR0 [3:0] | | | | DATAVSIZE [1:0] | | LNK1ENA | DATAVMATCH | CYCMATCH | Res. | EMITRANGE | Res. | FUNCTION [3:0] | | | |
| | Reset value | | | | | | | | 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 |
| 0xFD0 | DWT_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 4KCOUNT [3:0] | | | | JEP106CON [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0xFE0 | DWT_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0xFE4 | DWT_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0xFE8 | DWT_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0xFEC | DWT_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | DWT_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | DWT_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0xFF8 | DWT_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | DWT_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Refer to *Section 32.13: CPU1 ROM table* for the register boundary addresses.

## 32.15 CPU1 instrumentation trace macrocell (ITM)

The ITM generates trace information as packets four sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. Software trace

   Software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

2. Hardware trace

   The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

3. Local timestamping

   The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

### 32.15.1 ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + x * 0x4 (x = 0 to 31)

Reset value: Unknown

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STIMULUS[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 |
| STIMULUS[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **STIMULUS[31:0]**: Write data is output on the trace bus as a software event packet. When reading, bit 0 is a FIFOREADY indicator:

0: Stimulus port buffer is full (or port is disabled)

1: Stimulus port can accept new write data

### 32.15.2 ITM trace enable register (ITM_TER)

Address offset: 0x080

Reset value: 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STIMENA[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 |
| STIMENA[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **STIMENA[31:0]**: Each bit n (0:31) enables the stimulus port associated with the ITM_STIMRn register.

0: Port disabled
1: Port enabled

### 32.15.3 ITM trace privilege register (ITM_TPR)

Address offset: 0xE00

Reset value: 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRIVMASK[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 |
| PRIVSK[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **PRIVMASK[31:0]**: Enable unprivileged access to ITM stimulus ports. Each bit controls eight stimulus ports.

0bXXX0: Unprivileged access permitted on ports 0 to 7
0bXXX1: Only privileged access permitted on ports 0 to 7
0bXX0X: Unprivileged access permitted on ports 8 to 15
0bXX1X: Only privileged access permitted on ports 8 to 15
0bX0XX: Unprivileged access permitted on ports 16 to 23
0bX1XX: Only privileged access permitted on ports 16 to 23
0b0XXX: Unprivileged access permitted on ports 24 to 31
0b1XXX: Only privileged access permitted on ports 24 to 31

### 32.15.4 ITM trace control register (ITM_TCR)

Address offset: 0xE80

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. | BUSY | TRACEBUSID[6:0] | | | | | | |
| | | | | | | | | 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. | Res. | Res. | TSPRESCALE[1:0] | | Res. | Res. | Res. | SWOENA | TXENA | SYNCENA | TSENA | ITMENA |
| | | | | | | rw | rw | | | | r | rw | rw | rw | rw |

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

Bit 23 **BUSY**: Indicates whether the ITM is currently processing events (read only).
0: Not busy
1: Busy

Bits 22:16 **TRACEBUSID[6:0]**: Identifier for multi-source trace stream formatting. If multi-source trace is in use, the debugger must write a non-zero value to this field. Note: different IDs must be used for each trace source in the system.

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

Bits 9:8 **TSPRESCALE[1:0]**: Local timestamp prescaler, used with the trace packet reference clock. The possible values are:
0x0: No prescaling.
0x1: Divide by 4.
0x2: Divide by 16.
0x3: Divide by 64.

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

Bit 4 **SWOENA**: Enables asynchronous clocking of the timestamp counter (read only).
0: Timestamp counter uses processor clock

Bit 3 **TXENA**: Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: Disabled
1: Enabled

Bit 2 **SYNCENA**: Enables synchronization packet transmission.
*Note: The debugger setting this bit must also configure the DWT_CTRLR register SYNCTAP field in the DWT for correct synchronization speed.*
0: Disabled
1: Enabled

Bit 1 **TSENA**: Enables local timestamp generation.
0: Disabled
1: Enabled

Bit 0 **ITMENA**: Enables the ITM.
0: Disabled
1: Enabled

### 32.15.5 ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

### 32.15.6 ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x01: ITM part number

### 32.15.7 ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

| 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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x0: ITM part number

### 32.15.8 ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

### 32.15.9 ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.15.10 ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.15.11 ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.15.12 ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 32.15.13 ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

### 32.15.14 ITM register map and reset values

The ITM registers are located at address range 0xE0000000 to 0xE0000FFC.

**Table 210. CPU1 ITM register map and reset values**

| Offset | Register name | 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 to 0x07C | ITM_STIM0-31R | STIMULUS[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x0E00 | ITM_TER | STIMENA[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 |
| 0x0E40 | ITM_TPR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PRIVMASK[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0xE80 | ITM_TCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BUSY | TRACEBUSID[6:0] | | | | | | | Res. | Res. | Res. | Res. | GTSFREQ[1:0] | | TSPRESCALE[1:0] | | Res. | Res. | Res. | SWOENA | TXENA | SYNCENA | TSENA | ITMENA |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 0 | | | | 0 | 0 | 0 | 0 | 0 |
| 0xFD0 | ITM_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 4KCOUNT [3:0] | | | | JEP106CON [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0xFE0 | ITM_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0xFE4 | ITM_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0xFE8 | ITM_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0xFEC | ITM_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | ITM_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | ITM_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0xFF8 | ITM_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | ITM_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Refer to *Section 32.13: CPU1 ROM table* for the register boundary addresses.

## 32.16 CPU1 breakpoint unit (FPB)

The FPB allows the user to set hardware breakpoints. It contains six comparators that monitor the instruction fetch address and two literal address comparators. If a match occurs, the address is remapped to an address in system memory, defined by the FPB_REMAPR register plus an offset corresponding to the matching comparator. Alternatively, the instruction comparators can be configured to generate a breakpoint instruction.

### 32.16.1 FPB control register (FPB_CTRLR)

Address offset: 0x000

Reset value: 0x0000 0260

| 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. | NUM_CODE[6:4] | | | NUM_LIT[3:0] | | | | NUM_CODE[3:0] | | | | Res. | Res. | KEY | ENABLE |
| | r | r | r | r | r | r | r | r | r | r | r | | | rw | rw |

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

Bits 14:12 **NUM_CODE[6:4]**: Number of instruction address comparators supported - least significant bits (read only).

0x0: 6 instruction comparators supported.

Bits 11:8 **NUM_LIT[3:0]**: Number of literal address comparators supported (read only).

0x2: Two literal comparators supported.

Bits 7:4 **NUM_CODE[3:0]**: Number of instruction address comparators supported - least significant bits (read only).

0x6: 6 instruction comparators supported

Bit 1 **KEY**: Write protect key. A write to FPB_CTRLR register is ignored if this bit is not set to 1.

Bits 0 **ENABLE**: FPB enable

0x0: Disable

0x1: Enable

### 32.16.2 FPB remap register (FPB_REMAPR)

Address offset: 0x004

Reset value: 0x2000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | RMPSPT | REMAP[23:11] | | | | | | | | | | | | |
| | | r | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REMAP[10:0] | | | | | | | | | | | Res. | Res. | Res. | Res. | Res. |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | | | |

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

Bit 29 **RMPSPT**: Indicates whether Flash memory patch remap is supported (read only).
0x1: Remapping supported.

Bits 28:5 **REMAP[23:0]**: Remap target address. Bits [28:5] of the base address in SRAM to which the FPB remaps the address. The remap base address must be aligned to the number of words required to support the implemented comparators, that is to (NUM_CODE+NUM_LIT) words, with a minimum alignment of 8 words. Because remap is into the SRAM memory region, 0x20000000-0x3FFFFFFF, bits [31:29] of the remap address are 0b001.

Bits 4:0 Reserved, must be kept at reset value.

## 32.16.3 FPB comparator registers (FPB_COMPxR)

Address offset: 0x008 + x * 0x4 (for x = 0 to 7)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REPLACE[1:0] | | Res. | COMP[26:14] | | | | | | | | | | | | |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COMP[13:0] | | | | | | | | | | | | | | Res. | ENABLE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw |

Bits 31:30 **REPLACE[1:0]**: Defines the behavior when a match occurs between the COMP field and the instruction fetch address.
0x0: Reserved
0x1: Breakpoint on lower half-word, upper half-word is unaffected.
0x2: Breakpoint on upper half-word, lower half-word is unaffected.
0x3: Breakpoint on both upper and lower half-words.

Bit 29 Reserved, must be kept at reset value.

Bits 28:2 **COMP[26:0]**: Value to compare with address bits 28:2 of accesses to instruction code memory (0x00000000 to 0x1FFFFFFF). If a match occurs, the action to be taken is defined by the REPLACE field.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Comparator enable. The comparator is only enabled if both this bit and the FPB ENABLE bit in the FPB_CTRLR register are set.
0: Disabled
1: Enabled

## 32.16.4 FPB CoreSight peripheral identity register 4 (FPB_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

### 32.16.5 FPB CoreSight peripheral identity register 0 (FPB_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0003

| 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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x03: FPB part number

### 32.16.6 FPB CoreSight peripheral identity register 1 (FPB_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

| 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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x0: FPB part number

### 32.16.7 FPB CoreSight peripheral identity register 2 (FPB_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x2: r0p3

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

### 32.16.8 FPB CoreSight peripheral identity register 3 (FPB_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.16.9 FPB CoreSight component identity register 0 (FPB_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.16.10 FPB CoreSight peripheral identity register 1 (FPB_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.16.11 FPB CoreSight component identity register 2 (FPB_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 32.16.12 FPB CoreSight component identity register 3 (FPB_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

### 32.16.13 FPB register map and reset values

The CPU1 FPB registers are located at address range 0xE0002000 to 0xE0002FFC.

**Table 211. CPU1 FPB register map and reset values**

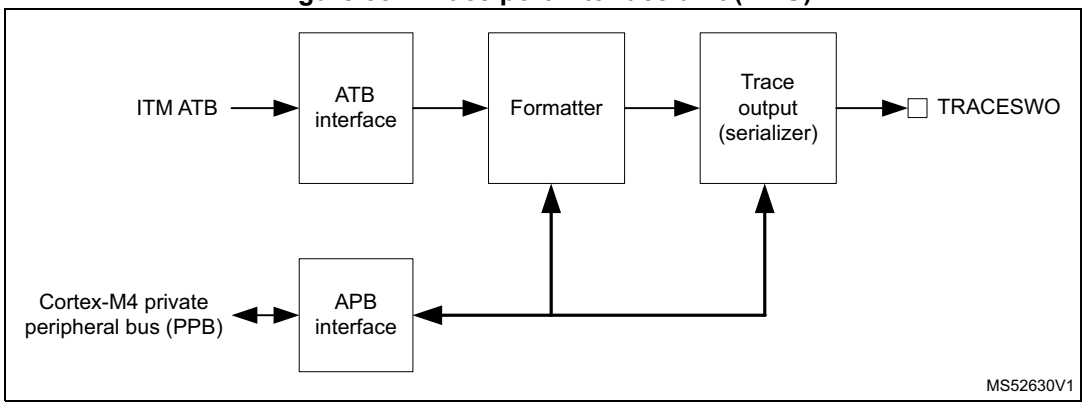
| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000 | FPB_CTRLR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | NUM_CODE[6:4] | | | NUM_LIT[3:0] | | | | NUM_CODE[3:0] | | | | Res. | Res. | KEY | ENABLE |
| | Reset value | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | | | 0 | 0 |
| 0x004 | FPB_REMAPR | Res. | Res. | RMPSP | REMAP[23:0] | | | | | | | | | | | | | | | | | | | | | | | | Res. | Res. | Res. | Res. | Res. |
| | Reset value | | | 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 | | | | | |
| 0x008 to 0x024 | FPB_COMP0-7R | REPLACE[1:0] | | Res. | COMP[26:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | Res. | ENABLE |
| | 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 |
| 0xFD0 | FPB_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 4KCOUNT [3:0] | | | | JEP106CON [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0xFE0 | FPB_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0xFE4 | FPB_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0xFE8 | FPB_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0xFEC | FPB_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | FPB_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | FPB_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0xFF8 | FPB_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | FPB_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Refer to *Section 32.13: CPU1 ROM table* for the register boundary addresses.

## 32.17 CPU1 trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in *Figure 352*, the TPIU has one ATB slave port for incoming trace data from the ITM.

**Figure 352. Trace port interface unit (TPIU)**




Trace data can be output on the serial wire output, TRACESWO.

For more information on the Trace Port Interface in the CPU1 refer to the Arm® Cortex®-M4 Technical Reference Manual *[2]*.

### 32.17.1 TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

Reset value: 0x0000 000F

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PORTSIZE[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 |
| PORTSIZE[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **PORTSIZE[31:0]**: Indicates supported trace port sizes, from 1 to 32 pins. Bit n-1 when set indicates that port size n is supported.

0x0000 000F: Port sizes 1 to 4 supported

### 32.17.2 TPIU current port size register (TPIU_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PORTSIZE[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 |
| PORTSIZE[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **PORTSIZE[31:0]**: Indicates current trace port size. Bit n-1 when set indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result. This register should only be modified when the formatter is stopped.

### 32.17.3 TPIU asynchronous clock presecaler register (TPIU_ACPR)

Address offset: 0x010

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. | PRESCALER[12:0] | | | | | | | | | | | | |
| | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 12:0 **PRESCALER[12:0]**: Selects the baud rate for the asynchronous output, TRACESWO. The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

### 32.17.4 TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0

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. | Res. | Res. | Res. | TXMODE[1:0] | |
| | | | | | | | | | | | | | | rw | rw |

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

Bits 1:0 **TXMODE[1:0]**: Selects the protocol used for trace output.
0x0: Reserved
0x1: Asynchronous SWO using Manchester encoding
0x2: Asynchronous SWO using NRZ encoding
0x3: Reserved

### 32.17.5 TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | FTNONSTOP | TCPRESENT | FTSTOPPED | FLINPROG |
| | | | | | | | | | | | | r | r | r | r |

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

Bit 3 **FTNONSTOP**: Indicates whether formatter can be stopped or not.
1: Formatter cannot be stopped

Bit 2 **TCPRESENT**: Indicates whether the optional TRACECTL output pin is available for use.
0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.
0: Formatter has not stopped
1: Formatter has stopped

Bit 0 **FLINPROG**: Flush in progress. Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.
0: No flush in progress
1: Flush in progress

### 32.17.6 TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0102

| 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. | TRIGIN | Res. | Res. | Res. | Res. | Res. | Res. | ENFCONT | Res. |
| | | | | | | | r | | | | | | | rw | |

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

Bit 8 **TRIGIN**: Trigger on trigger in.

1: Indicate a trigger in the trace stream when the TRIGIN input is asserted.

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

Bit 1 **ENFCONT:** Enable continuous formatting. Setting this bit to zero in SWO mode bypasses the formatter and only ITM/DWT trace is output. ETM trace is discarded.

0: Continuous formatting is disabled

1: Continuous formatting is enabled

Bit 0 Reserved, must be kept at reset value.

### 32.17.7 TPIU formatter synchronization counter register (TPIU_FSCR)

Address offset: 0x308

Reset value: 0x0000 0040

| 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. | CYCCOUNT[12:0] | | | | | | | | | | | | |
| | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 12:0 **CYCCOUNT[12:0]**: Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

### 32.17.8 TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

| 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. | CLAIMSET[3:0] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:

0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3

Read:

0xF: Indicates there are four bits in claim tag

## 32.17.9 TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

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. | CLAIMCLR[3:0] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:

0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

## 32.17.10 TPIU device configuration register (TPIU_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0000 0CA1

| 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. | SWO UARTNRZ | SWO MAN | TCLK DATA | FIFO SIZE[2:0] | | | CLK RELAT | MAXNUM[3:0] | | | | |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

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

Bit 11 **SWONRZ**: Indicates whether Serial wire output, NRZ, is supported.
0x1: Supported

Bit 10 **SWOMAN**: Indicates whether Serial wire output, Manchester encoded format, is supported.
0x1: Supported

Bit 9 **TCLKDATA**: Indicates whether trace clock plus data is supported
0x0: Supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT**: Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)
0x1: Asynchronous

Bits 4:0 **MAXNUM[4:0]**: Number/type of ATB input port multiplexing
0x1: Two input ports, only one port is used in this product

### 32.17.11 TPIU device type identifier register (TPIU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0011

| 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. | SUBTYPE[3:0] | | | | MAJORTYPE[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

### 32.17.12 TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

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. | 4KCOUNT[3:0] | | | | JEP106CON[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

## 32.17.13 TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00A1

| 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. | PARTNUM[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0xA1: CPU1 TPIU part number

## 32.17.14 TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

| 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. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x9: CPU1 TPIU part number

### 32.17.15 TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 004B

| 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. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
0x4: r0p5

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

### 32.17.16 TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

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. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 32.17.17 TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

| 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. | PREAMBLE[7:0] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

### 32.17.18 TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

| 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. | CLASS[3:0] | | | | PREAMBLE[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

### 32.17.19 TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

| 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. | PREAMBLE[19:12] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

## 32.17.20 TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

| 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. | PREAMBLE[27:20] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

## 32.17.21 CPU1 TPIU register map and reset values

**Table 212. CPU1 TPIU register map and reset values**

| Offset | Register name | 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 | TPIU_SSPSR | PORTSIZE[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 | 1 | 1 | 1 | 1 |
| 0x004 | TPIU_CSPSR | PORTSIZE[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 | 1 |
| 0x010 | TPIU_ACPR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PRESCALER[12:0] | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0F0 | TPIU_SPPR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TXMODE[1:0] | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 1 |
| 0x300 | TPIU_FFSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | FTNONSTOP | TCPRESENT | FTSTOPPED | FLINPROG |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 0 | 0 |
| 0x304 | TPIU_FFCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TRIGIN | Res. | Res. | Res. | Res. | Res. | Res. | ENFCONT | Res. |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | 1 | |
| 0x308 | TPIU_FSCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CYCCOUNT[11:0] | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFA0 | TPIU_CLAIMSETR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLAIMSET[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 1 |
| 0xFA4 | TPIU_CLAIMCLR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLAIMCLR[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 0xFC8 | TPIU_DEVIDR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SWONRZ | SWOMAN | TCLKDATA | FIFOSIZE[2:0] | | | CLKRELAT | MUXNUM[4:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0xFD0 | TPIU_DEVTYPER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SUBTYPE [3:0] | | | | MAJORTYPE [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0xFD0 | TPIU_PIDR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 4KCOUNT [3:0] | | | | JEP106CON [3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Table 212. CPU1 TPIU register map and reset values (continued)**

| Offset | Register name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xFD4 | TPIU_PIDR5 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFD8 | TPIU_PIDR6 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFDC | TPIU_PIDR7 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFE0 | TPIU_PIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PARTNUM[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0xFE4 | TPIU_PIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | JEP106ID [3:0] | | | | PARTNUM [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0xFE8 | TPIU_PIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVISION [3:0] | | | | JEDEC | JEP106ID [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0xFEC | TPIU_PIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xFF0 | TPIU_CIDR0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0xFF4 | TPIU_CIDR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | CLASS[3:0] | | | | PREAMBLE [11:8] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0xFF8 | TPIU_CIDR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xFFC | TPIU_CIDR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

## 32.18 CPU1 cross trigger interface (CTI)

See *Section 32.6: Cross trigger interface (CTI) and matrix (CTM)*.

## 32.19 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8 Aug 2013
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G, 16 March 2015
3. DDI 0461B (ID010111) - Arm® CoreSight™ Trace Memory Controller r0p1 Technical Reference Manual, Issue B, 10 Dec 2010
4. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009
5. DDI 0403D (ID100710) - Arm®v7-M Architecture Reference Manual, Issue E.b, 2 December 2014

# 33 Device electronic signature

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.

## 33.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial number (USB string serial number, or other end applications)
- for use as part of the security keys, to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the memory
- to activate processes such as secure boot.

The 96-bit unique device identifier provides a reference number, unique for a given device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF 7590

Address offset: 0x00

Read only = 0xXXXX XXXX, where 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]:** X and Y coordinates on the wafer expressed in BCD format

Address offset: 0x04

Read only = 0xXXXX XXXX, where X is factory-programmed

| 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 | 14 | 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 31:8 **UID[63:40]:** LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF_NUM[7:0]
Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX, where X is factory-programmed

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UID[95:80] | | | | | | | | | | | | | | | |
| 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[79:64] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **UID[95:64]:** LOT_NUM[55:24]
Lot number (ASCII encoded)

## 33.2 Memory size data register

### 33.2.1 Flash size data register

Base address: 0x1FFF 75E0

Address offset: 0x00

Read only = 0xXXXX, where X is factory-programmed

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLASH_SIZE | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 15:0 **FLASH_SIZE[15:0]**: Flash memory size
Indicates the size of the device Flash memory, expressed in Kbytes.
As an example, 0x040 corresponds to 64 Kbytes.

## 33.3 Package data register

Base address: 0x1FFF 7500

Address offset: 0x00

Read only = 0xXXXX, where X is factory-programmed

| 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. | PKG[4:0] | | | | |
| | | | | | | | | | | | r | r | r | r | r |

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]**: Package type
01010: UFQFPN48
Others: reserved

## 33.4 Part number codification register

Base address: 0x1FFF 77DC

Address offset: 0x00

Read only = 0xXXXX XXXX, where X is factory-programmed

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CODIFICATION[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CODIFICATION[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **CODIFICATION[31:0]**: Part number codification
0x0000 3033 STMicroelectronics STM32WB30xx part number codification
0xXXXX XXXX STMicroelectronics STM32WB50xx part number codification

# 34 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.

# 35 Revision history

**Table 213. Document revision history**

| Date | Revision | Changes |
|---|---|---|
| 11-Sep-2019 | 1 | Initial release. |
| 26-Feb-2020 | 2 | Updated *Section 3.5: FLASH UID64*, *CPU2 secure SRAM2 areas*, *Section 6.1.1: Independent analog peripherals supply*, *Section 6.4.12: Auto wake-up from Low-power mode*, *Section 6.6.3: PWR control register 3 (PWR_CR3)*, *Section 7.2: Clocks*, *External source*, *Section 7.2.2: HSI16 clock*, *Section 7.2.8: LSI2 clock*, *Section 7.4.1: RCC clock control register (RCC_CR)*, *Section 7.4.29: RCC control/status register (RCC_CSR)*, *Reading the temperature*, *Calculating the actual $V_{DDA}$ voltage using the internal reference voltage*, *Section 16.3.3: Random number generation*, *Section 17.1: Introduction*, *Section 17.2: AES main features*, *Section 17.4.4: AES procedure to perform a cipher operation*, *Section 17.4.5: AES decryption round key preparation*, *Section 17.4.13: AES data registers and data swapping*, *Section 17.4.16: AES DMA interface*, *Section 17.4.17: AES error management*, *Section 32.5: Access ports*, *Example configurations*, *Section 32.10: CPU2 data watchpoint and trace unit (DWT)*, *Section 32.14: CPU1 data watchpoint and trace unit (DWT)* and *Section 32.19: References*.<br>Added note in *Section 3.8: FLASH interrupts*, *Section 18.3.3: PKA reset and clocks*, *Table 149: Interrupt control bits*.<br>Updated *Table 2: Boot modes*, *Table 3: Flash memory - Single bank organization*, *Table 23: Functionalities depending on system operating mode*, *Table 37: RCC register map and reset values*, *Table 56: Wakeup interrupt table*, *Table 89: Processing latency for ECB, CBC and CTR*, *Table 117: Modular exponentiation computation times*, *Table 118: ECC scalar multiplication computation times*, *Table 119: ECDSA signature average computation times*, *Table 120: ECDSA verification average computation times*, *Table 120: ECDSA verification average computation times*, *Table 148: Effect of low-power modes on RTC*, *Table 197: CPU1 CTI inputs* and *Table 207: CPU1 ROM table*.<br>Updated *Figure 14: Clock tree*, *Figure 19: Three-volt or Five-volt tolerant GPIO structure (TT or FT)*, *Figure 20: Input floating / pull up / pull down configurations*, *Figure 21: Output configuration*, *Figure 22: Alternate function configuration*, *Figure 23: High impedance-analog configuration*, *Figure 347: Embedded cross trigger* and *Figure 351: CPU1 CoreSight™ topology*.<br>Removed former *Table 88: DMA channel configuration for memory-to-AES data transfer* and *Table 89: DMA channel configuration for AES-to-memory data transfer*.<br>Removed former *Figure 20: Basic structure of a 5-Volt tolerant I/O port bit* and *Figure 347: Debugger connection to debug components*.<br>Minor text edits across the whole document. |

**Table 213. Document revision history (continued)**

| Date | Revision | Changes |
|---|---|---|
| 16-Jul-2020 | 3 | Added STM32WB30CE devices.<br>Updated *Introduction*, *PCROP1A start address option bytes*, *PCROP1A end address option bytes*, *WRP Area A address option bytes*, *WRP Area B address option bytes*, *PCROP1B start address option bytes*, *PCROP1B end address option bytes*, *Section 3.10.5: Flash memory control register (FLASH_CR)*, sections *3.10.8* to *3.10.13*, *Section 3.10.17: Flash memory CPU2 control register (FLASH_C2CR)*,*Section 8.3.2: I/O pin alternate function multiplexer and mapping*, *Calculating the actual $V_{DDA}$ voltage using the internal reference voltage*, *Montgomery space and fast mode operations*, *Section 18.7.2: PKA status register (PKA_SR)*, *Section 18.7.2: PKA status register (PKA_SR)*, *Section 19.4.1: TIM1 control register 1 (TIM1_CR1)*, *Section 19.4.8: TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)*, *Section 19.4.20: TIM1 break and dead-time register (TIM1_BDTR)*, *Section 19.4.27: TIM1 alternate function option register 1 (TIM1_AF1)Section 19.4.28: TIM1 Alternate function register 2 (TIM1_AF2)*, *Section 20.4.8: TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)*, *Section 22.2: LPTIM main features*, *Section 22.7.1: LPTIM interrupt and status register (LPTIM_ISR)*, *Section 22.7.2: LPTIM interrupt clear register (LPTIM_ICR)*, *Section 32.1: Introduction*, *Section 32.4.7: DP data link control register (DP_DLCR)*, *Section 32.5.1: AP control/status word register (AP_CSWR)*, *Section 32.5.5: AP base address register (AP_BASER)*, *Section 32.5.6: AP identification register (AP_IDR)*, *Section 32.8.2: DBGMCU configuration register (DBGMCU_CR)*, *Section 32.8.4: DBGMCU CPU2 APB1 peripheral freeze register 1 (DBGMCU_C2APB1FZR1)* and *Section 32.8.8: DBGMCU CPU2 APB2 peripheral freeze register (DBGMCU_C2APB2FZR)*.<br>Updated *Table 3: Flash memory - Single bank organization*, *Table 117: Modular exponentiation computation times*, *Table 122: Montgomery parameters average computation times*, *Table 197: CPU1 CTI inputs* and *Table 207: CPU1 ROM table*.<br>Updated *Figure 14: Clock tree*, *Figure 138: Control circuit in normal mode, internal clock divided by 1*, *Figure 144: Capture/compare channel 1 main circuit*, *Figure 175: General-purpose timer block diagram*, *Figure 224: TIM16/TIM17 block diagram* and *Figure 238: Capture/compare channel 1 main circuit*.<br>Added *Note:* to *Section 7.4.3: RCC clock configuration register (RCC_CFGR)*, *Table 121: Point on elliptic curve Fp check average computation times*, *Table 139: LPTIM implementation* and *Section 21.3.18: Using timer output as trigger for other timers (TIM16/TIM17)*.<br>Minor text edits across the whole document. |

**Table 213. Document revision history (continued)**

| Date | Revision | Changes |
|---|---|---|
| 02-Apr-2021 | 4 | Updated *Introduction*, *Related documents*, *Section 2.1: System architecture*, *Section 3.10.5: Flash memory control register (FLASH_CR)*, *Section 3.10.6: Flash memory ECC register (FLASH_ECCR)*, *Section 3.10.8: Flash memory PCROP zone A start address register (FLASH_PCROP1ASR)*, *Section 4.1: Introduction*, *Section 4.2: Main features*, *Polynomial programmability*, *Section 6.1.1: Independent analog peripherals supply*, *Entering Stop0 mode*, *Entering Stop2 mode*, *Section 6.6.2: PWR control register 2 (PWR_CR2)*, *Section 6.6.6: PWR status register 2 (PWR_SR2)*, *Section 7.2: Clocks*, *Section 7.4.1: RCC clock control register (RCC_CR)*, *Section 7.4.4: RCC PLL configuration register (RCC_PLLCFGR)*, *Section 9.2.13: SYSCFG CPU1 interrupt mask register 2 (SYSCFG_IMR2)*, *Section 9.2.15: SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2)*, *Section 12.4.4: DMAMUX request line multiplexer*, *Section 12.4.5: DMAMUX request generator*, *Section 15.3.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*, *Triggered injection mode*, *Clock ratio constraint between ADC clock and AHB clock*, *Section 18.3.3: PKA reset and clocks*, *Montgomery space and fast mode operations*, *Enabling/disabling PKA*, *Using precomputed Montgomery parameters (PKA fast mode)*, sections *18.7.2: PKA status register (PKA_SR)* to *18.7.4: PKA RAM* and *Section 22.7.1: LPTIM interrupt and status register (LPTIM_ISR)*.<br>Updated *Figure 5: Radio system block diagram* and *Figure 14: Clock tree*.<br>Updated *Table 23: Functionalities depending on system operating mode*, *Table 33: PWR register map and reset values*, *Table 40: SYSCFG register map and reset values*, *Table 54: CPU1 vector table*, *Table 55: CPU2 vector table*, *Table 56: Wakeup interrupt table*, *Table 64: ADC input/output pins* and *Table 122: Montgomery parameters average computation times*.<br>Added footnotes *1* and *2* to *Table 1: STM32WB50CG/30CE memory map and peripheral register boundary addresses*.<br>Removed former *Section 6.2.3: Peripheral voltage monitoring (PVM)* and *Table 139: LPTIM implementation.*<br>Minor text edits across the whole document. |

**Table 213. Document revision history (continued)**

| Date | Revision | Changes |
|---|---|---|
| 28-Aug-2021 | 5 | Updated *Section 3.5: FLASH UID64*, *Section 5.2: CRC main features*, *Section 7.2.8: LSI2 clock*, *Section 7.2.12: Clock security system on LSE (LSECSS)*, *Section 19.3.16: Using the break function*, *Section 19.4.8: TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)*, *Section 20.4.8: TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)*, *Section 21.3.11: Using the break function*, *Section 24.3.14: Calibration clock output*, *Section 28.8.3: USART control register 2 (USART_CR2)* and *Section 32.8.1: DBGMCU identity code register (DBGMCU_IDCODE)*.<br>Added *Section 21.3.13: 6-step PWM generation*.<br>Changed SCLK into CK throughout *Section 28: Universal synchronous/asynchronous receiver transmitter (USART/UART)*.<br>Minor text edits across the whole document.<br>Updated *Figure 6: CRC calculation unit block diagram*, *Figure 115: Advanced-control timer block diagram*, *Figure 138: Control circuit in normal mode, internal clock divided by 1*, *Figure 234: Control circuit in normal mode, internal clock divided by 1*, *Figure 294: TC/TXE behavior when transmitting*, *Figure 295: Start bit detection when oversampling by 16 or 8* and *Figure 303: USART example of synchronous master transmission*.<br>Updated *Table 200: DBGMCU register map and reset values*. |
| 11-May-2022 | 6 | Updated *Section 3.3.1: Flash memory organization*, *Section 3.5: FLASH UID64*, *DMA operation in different operating modes*, *Section 6.4.4: Exiting Low-power mode*, *Section 7.4.31: RCC clock HSE register (RCC_HSECR)*, *Section 15.3.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*, *Section 16.2: RNG main features*, *Section 28.5.20: RS232 hardware flow control and RS485 Driver Enable*, *Section 28.8.4: USART control register 3 (USART_CR3)*, *Section 32.7.3: CTI application trigger set register (CTI_APPSETR)*, and *Section 32.8.1: DBGMCU identity code register (DBGMCU_IDCODE)*.<br>Updated *Figure 14: Clock tree*. *Figure 106: GCM authenticated encryption*, *Figure 270: Transfer bus diagrams for I2C slave transmitter (mandatory events only)*, *Figure 273: Transfer bus diagrams for I2C slave receiver (mandatory events only)*, and *Figure 280: Transfer bus diagrams for I2C master transmitter (mandatory events only)*.<br>Updated *Table 64: ADC input/output pins* and *Table 180: SPI register map and reset values*.<br>Added *Section 15.4: ADC in low-power mode*, *Section 33.4: Part number codification register*, and *Section 34: Important security notice*.<br>Minor text edits across the whole document. |
| 03-Jun-2022 | 7 | Updated *Introduction*, *Section 4.1: Introduction*, *Section 4.2: Main features*, *Section 7.4.28: RCC backup domain control register (RCC_BDCR)*, *Section 7.4.29: RCC control/status register (RCC_CSR)*, *Section 7.4.31: RCC clock HSE register (RCC_HSECR)*, *Analog watchdog*, and *Section 32.8.2: DBGMCU configuration register (DBGMCU_CR)*.<br>Updated *Figure 14: Clock tree*.<br>Minor text edits across the whole document. |

**Table 213. Document revision history (continued)**

| Date | Revision | Changes |
|---|---|---|
| 01-Feb-2023 | 8 | Added *Empty check* and Caution in *Section 3.3.6: Flash memory program and erase operations*.<br>Updated *Section 3.3.7: Flash main memory erase sequences*, *Section 3.3.8: Flash main memory programming sequences*, note in *Section 3.6.1: Read protection (RDP)*, *Section 3.10.4: Flash memory status register (FLASH_SR)*, *Section 3.10.5: Flash memory control register (FLASH_CR)*, *Section 3.10.19: Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)*, *Section 6.2.2: Programmable voltage detector (PVD)*, *Section 7.1.2: System reset*, *Section 7.4.31: RCC clock HSE register (RCC_HSECR)*, *Converting a supply-relative ADC measurement to an absolute voltage value*, *Section 19.4.7: TIM1 capture/compare mode register 1 (TIM1_CCMR1)*, *Section 20.4.7: TIM2 capture/compare mode register 1 (TIM2_CCMR1)*, *Section 26.4: WWDG interrupts*, *Section 26.5.2: WWDG configuration register (WWDG_CFR)*, *Section 32.4.11: DP access port select register (DP_SELECTR)*, and *Section 32.7.3: CTI application trigger set register (CTI_APPSETR)*.<br>Added footnote *2* to *Table 9: Option bytes organization*.<br>Updated *Figure 109: Surface charge transfer analog I/O group structure*, *Figure 110: Sampling capacitor voltage variation*, and *Figure 259: Watchdog block diagram*.<br>Updated *Table 193: Debug port register map and reset values*.<br>Minor text edits across the whole document. |
| 04-Sep-2023 | 9 | Updated *Introduction*, *Section 3.5: FLASH UID64*, *Section 3.10.16: Flash memory CPU2 status register (FLASH_C2SR)*, *Section 4.1: Introduction*, *Section 4.2: Main features*, *Entering Stop0 mode*, *Section 6.4.8: Stop1 mode*, *Entering Stop2 mode*, *Section 7.2.8: LSI2 clock*, *Section 7.2.19: Clock-out capability*, *Section 20.3.4: Charge transfer acquisition sequence*, *Section 19.3.22: Encoder interface mode*, and *Section 20.3.15: Encoder interface mode*.<br>Updated *Figure 2: Memory map*, *Figure 258: Independent watchdog block diagram*, *Figure 264: I2C initialization flow*, and *Figure 267: Slave initialization flow*.<br>Minor text edits across the whole document. |

# Index

## A

ADC_AWD2CR . . . . . .437
ADC_AWD3CR . . . . . .438
ADC_CALFACT . . . . . .439
ADC_CCR . . . . . .440
ADC_CFGR . . . . . .421
ADC_CFGR2 . . . . . .425
ADC_CR . . . . . .418
ADC_CSR . . . . . .439
ADC_DIFSEL . . . . . .438
ADC_DR . . . . . .433
ADC_IER . . . . . .416
ADC_ISR . . . . . .414
ADC_JDRy . . . . . .437
ADC_JSQR . . . . . .434
ADC_OFRy . . . . . .436
ADC_SMPR1 . . . . . .426
ADC_SMPR2 . . . . . .427
ADC_SQR1 . . . . . .430
ADC_SQR2 . . . . . .431
ADC_SQR3 . . . . . .432
ADC_SQR4 . . . . . .433
ADC_TR1 . . . . . .428
ADC_TR2 . . . . . .428
ADC_TR3 . . . . . .429
AES_CR . . . . . .494
AES_DINR . . . . . .498
AES_DOUTR . . . . . .498
AES_IVR0 . . . . . .501
AES_IVR1 . . . . . .501
AES_IVR2 . . . . . .501
AES_IVR3 . . . . . .502
AES_KEYR0 . . . . . .499
AES_KEYR1 . . . . . .500
AES_KEYR2 . . . . . .500
AES_KEYR3 . . . . . .500
AES_KEYR4 . . . . . .502
AES_KEYR5 . . . . . .502
AES_KEYR6 . . . . . .503
AES_KEYR7 . . . . . .503
AES_SR . . . . . .497
AES_SUSPxR . . . . . .503
AP_BD0-3R . . . . . .1070
AP_DRWR . . . . . .1070
AP_TAR . . . . . .1070

## B

BPU_CIDR0 . . . . . 1133
BPU_CIDR1 . . . . . 1134
BPU_CIDR2 . . . . . 1134
BPU_CIDR3 . . . . . 1135
BPU_COMPxR . . . . . 1131
BPU_CTRLR . . . . . 1130
BPU_PIDR0 . . . . . 1132
BPU_PIDR1 . . . . . 1132
BPU_PIDR2 . . . . . 1132
BPU_PIDR3 . . . . . 1133
BPU_PIDR4 . . . . . 1131
BPU_REMAPR . . . . . 1130

## C

C1ROM_CIDR0 . . . . . 1141
C1ROM_CIDR1 . . . . . 1141
C1ROM_CIDR2 . . . . . 1142
C1ROM_CIDR3 . . . . . 1142, 1144
C1ROM_MEMTYPER . . . . . 1138
C1ROM_PIDR0 . . . . . 1139
C1ROM_PIDR1 . . . . . 1140
C1ROM_PIDR2 . . . . . 1140
C1ROM_PIDR3 . . . . . 1141
C1ROM_PIDR4 . . . . . 1139
C2ROM1_CIDR0 . . . . . 1106
C2ROM1_CIDR1 . . . . . 1107
C2ROM1_CIDR2 . . . . . 1107
C2ROM1_CIDR3 . . . . . 1107
C2ROM1_MEMTYPER . . . . . 1104
C2ROM1_PIDR0 . . . . . 1104
C2ROM1_PIDR1 . . . . . 1105
C2ROM1_PIDR2 . . . . . 1105
C2ROM1_PIDR3 . . . . . 1106
C2ROM1_PIDR4 . . . . . 1104
C2ROM2_CIDR0 . . . . . 1112
C2ROM2_CIDR1 . . . . . 1113
C2ROM2_CIDR2 . . . . . 1113
C2ROM2_CIDR3 . . . . . 1113
C2ROM2_MEMTYPER . . . . . 1110
C2ROM2_PIDR0 . . . . . 1110
C2ROM2_PIDR1 . . . . . 1111
C2ROM2_PIDR2 . . . . . 1111
C2ROM2_PIDR3 . . . . . 1112
C2ROM2_PIDR4 . . . . . 1110
CRC_CR . . . . . 124
CRC_DR . . . . . 123

CRC_IDR . . . . . 123
CRC_INIT . . . . . 125
CRC_POL . . . . . 125
CTI_APPCLEAR . . . . . 1079
CTI_APPPULSER . . . . . 1080
CTI_APPSETR . . . . . 1078
CTI_AUTHSTATR . . . . . 1085
CTI_CHINSTSR . . . . . 1082
CTI_CHOUTSTSR . . . . . 1082
CTI_CIDR0 . . . . . 1089
CTI_CIDR1 . . . . . 1089
CTI_CIDR2 . . . . . 1090
CTI_CIDR3 . . . . . 1090
CTI_CLAIMCLR . . . . . 1084
CTI_CLAIMSETR . . . . . 1083
CTI_CONTROLR . . . . . 1078
CTI_DEVIDR . . . . . 1086
CTI_DEVTYPER . . . . . 1086
CTI_GATER . . . . . 1083
CTI_INENRx . . . . . 1080
CTI_INTACKR . . . . . 1078
CTI_LAR . . . . . 1084
CTI_LSR . . . . . 1085
CTI_OUTENRx . . . . . 1081
CTI_PIDR0 . . . . . 1087
CTI_PIDR1 . . . . . 1087
CTI_PIDR3 . . . . . 1088
CTI_PIDR4 . . . . . 1087
CTI_TRGISTSR . . . . . 1081
CTI_TRGOSTSR . . . . . 1082

## D

DBGMCU_APB1FZR1 . . . . . 1095
DBGMCU_APB1FZR2 . . . . . 1097
DBGMCU_APB2FZR . . . . . 1098
DBGMCU_C2APB1FZR1 . . . . . 1096
DBGMCU_C2APB1FZR2 . . . . . 1098
DBGMCU_C2APB2FZR . . . . . 1099
DBGMCU_CR . . . . . 1094
DBGMCU_IDCODE . . . . . 1094
DMA_CCRx . . . . . 304
DMA_CMARx . . . . . 308
DMA_CNDTRx . . . . . 307
DMA_CPARx . . . . . 307
DMA_IFCR . . . . . 303
DMA_ISR . . . . . 300
DMAMUX_CFR . . . . . 322
DMAMUX_CSR . . . . . 322
DMAMUX_CxCR . . . . . 321
DMAMUX_RGCFR . . . . . 324
DMAMUX_RGSR . . . . . 324
DMAMUX_RGxCR . . . . . 323
DP_ABORTR . . . . . 1059
DP_CTRL/STATR . . . . . 1060
DP_DPIDR . . . . . 1059
DWT_CIDR0 . . . . . 1125, 1154
DWT_CIDR1 . . . . . 1125, 1154
DWT_CIDR2 . . . . . 1125, 1154
DWT_CIDR3 . . . . . 1126, 1155
DWT_COMPxR . . . . . 1121, 1150
DWT_CPICNTR . . . . . 1118, 1147
DWT_CTRLR . . . . . 1116, 1145
DWT_CYCCNTR . . . . . 1118, 1147
DWT_EXCCNTR . . . . . 1119, 1148
DWT_FOLDCNTR . . . . . 1120, 1149
DWT_FUNCTxR . . . . . 1121, 1150
DWT_LSUCNTR . . . . . 1120, 1149
DWT_MASKxR . . . . . 1121, 1150
DWT_PCSR . . . . . 1120, 1149
DWT_PIDR0 . . . . . 1123, 1152
DWT_PIDR1 . . . . . 1123, 1152
DWT_PIDR2 . . . . . 1124, 1153
DWT_PIDR3 . . . . . 1124, 1153
DWT_PIDR4 . . . . . 1122, 1151
DWT_SLPCNTR . . . . . 1119, 1148

## E

EXTI_C2EMR1 . . . . . 349
EXTI_C2EMR2 . . . . . 351
EXTI_C2IMR1 . . . . . 348
EXTI_C2IMR2 . . . . . 350
EXTI_EMR1 . . . . . 349
EXTI_EMR2 . . . . . 351
EXTI_FTSR1 . . . . . 344
EXTI_FTSR2 . . . . . 346
EXTI_IMR1 . . . . . 348
EXTI_IMR2 . . . . . 350
EXTI_PR1 . . . . . 345
EXTI_PR2 . . . . . 347
EXTI_RTSR1 . . . . . 343
EXTI_RTSR2 . . . . . 345
EXTI_SWIER1 . . . . . 344
EXTI_SWIER2 . . . . . 346

## F

FLASH_ACR . . . . . 98
FLASH_C2ACR . . . . . 109
FLASH_C2CR . . . . . 112
FLASH_C2SR . . . . . 110
FLASH_CR . . . . . 101
FLASH_ECCR . . . . . 103
FLASH_IPCCBR . . . . . 109

FLASH_KEYR . . . . . 99
FLASH_OPTKEYR . . . . . 99
FLASH_OPTR . . . . . 104
FLASH_PCROP1AER . . . . . 107
FLASH_PCROP1ASR . . . . . 106
FLASH_PCROP1BER . . . . . 109
FLASH_PCROP1BSR . . . . . 108
FLASH_SFR . . . . . 113
FLASH_SR . . . . . 100
FLASH_SRRVR . . . . . 114
FLASH_WRP1AR . . . . . 107
FLASH_WRP1BR . . . . . 108
FPB_CIDR0 . . . . . 1170
FPB_CIDR1 . . . . . 1170
FPB_CIDR2 . . . . . 1170
FPB_CIDR3 . . . . . 1171
FPB_COMPxR . . . . . 1167
FPB_CTRLR . . . . . 1166
FPB_PIDR0 . . . . . 1168
FPB_PIDR1 . . . . . 1168
FPB_PIDR2 . . . . . 1169
FPB_PIDR3 . . . . . 1169
FPB_PIDR4 . . . . . 1167
FPB_REMAPR . . . . . 1166

## G

GPIOx_AFRH . . . . . 264
GPIOx_AFRL . . . . . 263
GPIOx_BRR . . . . . 265
GPIOx_BSRR . . . . . 261
GPIOx_IDR . . . . . 260
GPIOx_LCKR . . . . . 261
GPIOx_MODER . . . . . 257
GPIOx_ODR . . . . . 260
GPIOx_OSPEEDR . . . . . 258
GPIOx_OTYPER . . . . . 258
GPIOx_PUPDR . . . . . 259

## H

HSEM_CnICR . . . . . 1045
HSEM_CnIER . . . . . 1045
HSEM_CnISR . . . . . 1045
HSEM_CnMISR . . . . . 1046
HSEM_CR . . . . . 1046
HSEM_KEYR . . . . . 1047
HSEM_RLRx . . . . . 1044
HSEM_Rx . . . . . 1043

## I

I2C_CR1 . . . . . 885
I2C_CR2 . . . . . 887
I2C_ICR . . . . . 895
I2C_ISR . . . . . 893
I2C_OAR1 . . . . . 889
I2C_OAR2 . . . . . 890
I2C_PECR . . . . . 896
I2C_RXDR . . . . . 897
I2C_TIMEOUTR . . . . . 892
I2C_TIMINGR . . . . . 891
I2C_TXDR . . . . . 897
IPCC_C1CR . . . . . 1030
IPCC_C1MR . . . . . 1030
IPCC_C1SCR . . . . . 1031
IPCC_C1TOC2SR . . . . . 1032
IPCC_C2CR . . . . . 1032
IPCC_C2MR . . . . . 1033
IPCC_C2SCR . . . . . 1033
IPCC_C2TOC1SR . . . . . 1034
ITM_CIDR0 . . . . . 1163
ITM_CIDR1 . . . . . 1163
ITM_CIDR2 . . . . . 1163
ITM_CIDR3 . . . . . 1164
ITM_PIDR0 . . . . . 1161
ITM_PIDR1 . . . . . 1161
ITM_PIDR2 . . . . . 1162
ITM_PIDR3 . . . . . 1162
ITM_PIDR4 . . . . . 1160
ITM_STIMRx . . . . . 1158
ITM_TCR . . . . . 1159
ITM_TER . . . . . 1158
ITM_TPR . . . . . 1159
IWDG_KR . . . . . 823
IWDG_PR . . . . . 824
IWDG_RLR . . . . . 825
IWDG_SR . . . . . 826
IWDG_WINR . . . . . 827

## L

LPTIM_ARR . . . . . 775
LPTIM_CFGR . . . . . 770
LPTIM_CMP . . . . . 775
LPTIM_CNT . . . . . 776
LPTIM_CR . . . . . 773
LPTIM_ICR . . . . . 769
LPTIM_IER . . . . . 769
LPTIM_ISR . . . . . 768

## P

PKA_CLRFR . . . . . 530
PKA_CR . . . . . 528
PKA_SR . . . . . 529

PWR_C2CR1 . . . . . 165
PWR_C2CR3 . . . . . 166
PWR_CR1 . . . . . 154
PWR_CR2 . . . . . 155
PWR_CR3 . . . . . 156
PWR_CR4 . . . . . 157
PWR_EXTSCR . . . . . 167
PWR_PDCRA . . . . . 161
PWR_PDCRB . . . . . 162
PWR_PDCRC . . . . . 163
PWR_PDCRE . . . . . 163
PWR_PDCRH . . . . . 164
PWR_PUCRA . . . . . 160
PWR_PUCRB . . . . . 161
PWR_PUCRC . . . . . 162
PWR_PUCRE . . . . . 163
PWR_PUCRH . . . . . 164
PWR_SCR . . . . . 159
PWR_SR1 . . . . . 158
PWR_SR2 . . . . . 159

## R

RCC_AHB1ENR . . . . . 210, 230
RCC_AHB1RSTR . . . . . 205
RCC_AHB1SMENR . . . . . 215, 235
RCC_AHB2ENR . . . . . 211, 230
RCC_AHB2RSTR . . . . . 205
RCC_AHB2SMENR . . . . . 216, 236
RCC_AHB3ENR . . . . . 212, 231
RCC_AHB3RSTR . . . . . 206
RCC_AHB3SMENR . . . . . 217, 237
RCC_APB1ENR1 . . . . . 213, 232
RCC_APB1ENR2 . . . . . 213, 233, 235, 241
RCC_APB1RSTR1 . . . . . 207
RCC_APB1RSTR2 . . . . . 208
RCC_APB1SMENR1 . . . . . 218, 238
RCC_APB1SMENR2 . . . . . 219, 239
RCC_APB2ENR . . . . . 214, 234
RCC_APB2RSTR . . . . . 208-209
RCC_APB2SMENR . . . . . 219, 240
RCC_BDCR . . . . . 222
RCC_CCIPR . . . . . 220
RCC_CFGR . . . . . 195
RCC_CICR . . . . . 203
RCC_CIER . . . . . 200
RCC_CIFR . . . . . 202
RCC_CR . . . . . 191
RCC_CRRCR . . . . . 226-227
RCC_CSR . . . . . 224
RCC_ICSCR . . . . . 194
RCC_PLLCFGR . . . . . 197
RNG_CR . . . . . 454
RNG_DR . . . . . 455
RNG_SR . . . . . 454
RTC_ALRMAR . . . . . 805
RTC_ALRMASSR . . . . . 815
RTC_ALRMBR . . . . . 806
RTC_ALRMBSSR . . . . . 816
RTC_BKPxR . . . . . 817
RTC_CALR . . . . . 812
RTC_CR . . . . . 797
RTC_DR . . . . . 796
RTC_ISR . . . . . 800
RTC_OR . . . . . 817
RTC_PRER . . . . . 803
RTC_SHIFTR . . . . . 808
RTC_SSR . . . . . 807
RTC_TAMPCR . . . . . 813
RTC_TR . . . . . 795
RTC_TSDR . . . . . 810
RTC_TSSSR . . . . . 811
RTC_TSTR . . . . . 809
RTC_WPR . . . . . 807
RTC_WUTR . . . . . 804

## S

SPIx_CR1 . . . . . 1013
SPIx_CR2 . . . . . 1015
SPIx_CRCPR . . . . . 1019
SPIx_DR . . . . . 1018
SPIx_RXCRCR . . . . . 1019
SPIx_SR . . . . . 1017
SPIx_TXCRCR . . . . . 1019
SYSCFG_C2IMR1 . . . . . 281
SYSCFG_C2IMR2 . . . . . 281
SYSCFG_CFGR1 . . . . . 271
SYSCFG_CFGR2 . . . . . 278
SYSCFG_EXTICR1 . . . . . 272
SYSCFG_EXTICR2 . . . . . 273
SYSCFG_EXTICR3 . . . . . 274
SYSCFG_EXTICR4 . . . . . 276
SYSCFG_IMR1 . . . . . 280
SYSCFG_IMR2 . . . . . 280
SYSCFG_MEMRMP . . . . . 270
SYSCFG_SCSR . . . . . 277
SYSCFG_SIPCR . . . . . 282
SYSCFG_SKR . . . . . 279
SYSCFG_SWPR1 . . . . . 279
SYSCFG_SWPR2 . . . . . 279

## T

TIM1_AF1 . . . . . 626

TIM1_AF2 . . . . . 627
TIM1_ARR . . . . . 614
TIM1_BDTR . . . . . 617
TIM1_CCER . . . . . 611
TIM1_CCMR1 . . . . . 604-605
TIM1_CCMR2 . . . . . 608-609
TIM1_CCMR3 . . . . . 623
TIM1_CCR1 . . . . . 615
TIM1_CCR2 . . . . . 616
TIM1_CCR3 . . . . . 616
TIM1_CCR4 . . . . . 617
TIM1_CCR5 . . . . . 624
TIM1_CCR6 . . . . . 625
TIM1_CNT . . . . . 614
TIM1_CR1 . . . . . 593
TIM1_CR2 . . . . . 594
TIM1_DCR . . . . . 621
TIM1_DIER . . . . . 599
TIM1_DMAR . . . . . 622
TIM1_EGR . . . . . 603
TIM1_OR1 . . . . . 623
TIM1_PSC . . . . . 614
TIM1_RCR . . . . . 615
TIM1_SMCR . . . . . 597
TIM1_SR . . . . . 601
TIM1_TISEL . . . . . 627
TIM16_AF1 . . . . . 749
TIM16_OR1 . . . . . 749
TIM16_TISEL . . . . . 750
TIM17_AF1 . . . . . 750
TIM17_OR1 . . . . . 750
TIM17_TISEL . . . . . 751
TIM2_AF1 . . . . . 698
TIM2_ARR . . . . . 695
TIM2_CCER . . . . . 692
TIM2_CCMR1 . . . . . 686, 688
TIM2_CCMR2 . . . . . 690-691
TIM2_CCR1 . . . . . 695
TIM2_CCR2 . . . . . 695
TIM2_CCR3 . . . . . 696
TIM2_CCR4 . . . . . 696
TIM2_CNT . . . . . 693-694
TIM2_CR1 . . . . . 676
TIM2_CR2 . . . . . 677
TIM2_DCR . . . . . 697
TIM2_DIER . . . . . 682
TIM2_DMAR . . . . . 698
TIM2_EGR . . . . . 685
TIM2_OR1 . . . . . 698
TIM2_PSC . . . . . 694
TIM2_SMCR . . . . . 679
TIM2_SR . . . . . 683
TIM2_TISEL . . . . . 699
TIMx_ARR . . . . . 743
TIMx_BDTR . . . . . 745
TIMx_CCER . . . . . 740
TIMx_CCMR1 . . . . . 737-738
TIMx_CCR1 . . . . . 744
TIMx_CNT . . . . . 742
TIMx_CR1 . . . . . 732
TIMx_CR2 . . . . . 733
TIMx_DCR . . . . . 747
TIMx_DIER . . . . . 734
TIMx_DMAR . . . . . 748
TIMx_EGR . . . . . 736
TIMx_PSC . . . . . 743
TIMx_RCR . . . . . 744
TIMx_SR . . . . . 735
TPIU_ACPR . . . . . 1174
TPIU_CIDR0 . . . . . 1181
TPIU_CIDR1 . . . . . 1181
TPIU_CIDR2 . . . . . 1181
TPIU_CIDR3 . . . . . 1182
TPIU_CLAIMCLR . . . . . 1177
TPIU_CLAIMSETR . . . . . 1176
TPIU_CSPSR . . . . . 1173
TPIU_DEVIDR . . . . . 1177
TPIU_DEVTYPER . . . . . 1178
TPIU_FFCR . . . . . 1175
TPIU_FFSR . . . . . 1175
TPIU_FSCR . . . . . 1176
TPIU_PIDR0 . . . . . 1179
TPIU_PIDR1 . . . . . 1179
TPIU_PIDR2 . . . . . 1180
TPIU_PIDR3 . . . . . 1180
TPIU_PIDR4 . . . . . 1178
TPIU_SPPR . . . . . 1174
TPIU_SSPSR . . . . . 1173

## U

USART_BRR . . . . . 967
USART_CR1 . . . . . 951, 955
USART_CR2 . . . . . 958
USART_CR3 . . . . . 962
USART_GTPR . . . . . 967
USART_ICR . . . . . 981
USART_ISR . . . . . 970, 976
USART_PRESC . . . . . 984
USART_RDR . . . . . 983
USART_RQR . . . . . 969
USART_RTOR . . . . . 968
USART_TDR . . . . . 983

## W

WWDG_CFR . . . . . .833
WWDG_CR . . . . . .832
WWDG_SR . . . . . .834

**IMPORTANT NOTICE – READ CAREFULLY**

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2023 STMicroelectronics – All rights reserved