

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32U0 series microcontroller memory and peripherals.

The STM32U0 series are microcontroller lines with different packages and peripherals.

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

For information on the Arm® Cortex®-M0+ core, refer to the *Cortex®-M0+ Technical Reference Manual*.

The STM32U0 series microcontrollers include ST state-of-the-art patented technology.

Related documents

- Cortex®-M0+ Technical Reference Manual, available from: <http://infocenter.arm.com>
- STM32U0x1 and STM32U0x3 datasheets.
- STM32U031xx and STM32U073/83xx errata sheets.

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 | Memory and bus architecture | 53 |
| 2.1 | System architecture | 53 |
| 2.2 | Memory organization | 55 |
| 2.2.1 | Introduction | 55 |
| 2.2.2 | Memory map and register boundary addresses | 56 |
| 2.3 | Embedded SRAM | 60 |
| 2.4 | Flash memory overview | 61 |
| 2.5 | Boot configuration | 62 |
| 3 | Embedded flash memory (FLASH) | 64 |
| 3.1 | FLASH introduction | 64 |
| 3.2 | FLASH main features | 64 |
| 3.3 | FLASH functional description | 65 |
| 3.3.1 | Flash memory organization | 65 |
| 3.3.2 | FLASH empty check | 66 |
| 3.3.3 | FLASH error code correction (ECC) | 66 |
| 3.3.4 | FLASH read access latency | 67 |
| 3.3.5 | Flash memory acceleration | 68 |
| 3.3.6 | FLASH program and erase operations | 69 |
| 3.3.7 | FLASH main memory erase sequences | 70 |
| 3.3.8 | FLASH main memory programming sequences | 71 |
| 3.4 | FLASH option bytes | 75 |
| 3.4.1 | FLASH option byte description | 75 |
| 3.4.2 | FLASH option byte programming | 77 |
| 3.5 | Flash memory protection | 79 |
| 3.5.1 | FLASH read protection (RDP) | 79 |
| 3.5.2 | FLASH write protection (WRP) | 82 |

| | | |
|----------|--|------------|
| 3.5.3 | Securable memory area (HDP) | 83 |
| 3.5.4 | Securable memory area extension (HDP extension) | 84 |
| 3.5.5 | Disabling core debug access | 85 |
| 3.5.6 | Forcing boot from main flash memory | 86 |
| 3.6 | FLASH interrupts | 86 |
| 3.7 | FLASH registers | 86 |
| 3.7.1 | FLASH access control register (FLASH_ACR) | 86 |
| 3.7.2 | FLASH key register (FLASH_KEYR) | 88 |
| 3.7.3 | FLASH option key register (FLASH_OPTKEYR) | 88 |
| 3.7.4 | FLASH status register (FLASH_SR) | 88 |
| 3.7.5 | FLASH control register (FLASH_CR) | 91 |
| 3.7.6 | FLASH ECC register (FLASH_ECCR) | 92 |
| 3.7.7 | FLASH option register (FLASH_OPTR) | 93 |
| 3.7.8 | FLASH WRP area A address register (FLASH_WRP1AR) | 95 |
| 3.7.9 | FLASH WRP area B address register (FLASH_WRP1BR) | 96 |
| 3.7.10 | FLASH security register (FLASH_SECR) | 96 |
| 3.7.11 | FLASH OEM1 key register 1 (FLASH_OEM1KEYR1) | 97 |
| 3.7.12 | FLASH OEM1 key register 2 (FLASH_OEM1KEYR2) | 97 |
| 3.7.13 | FLASH OEM1 key register 3 (FLASH_OEM1KEYR3) | 98 |
| 3.7.14 | FLASH OEM1 key register 4 (FLASH_OEM1KEYR4) | 98 |
| 3.7.15 | FLASH OEM2 key register 1 (FLASH_OEM2KEYR1) | 98 |
| 3.7.16 | FLASH OEM2 key register 2 (FLASH_OEM2KEYR2) | 99 |
| 3.7.17 | FLASH OEM2 key register 3 (FLASH_OEM2KEYR3) | 99 |
| 3.7.18 | FLASH OEM2 key register 4 (FLASH_OEM2KEYR4) | 100 |
| 3.7.19 | FLASH OEM key status register (FLASH_OEMKEYSR) | 100 |
| 3.7.20 | FLASH HDP control register (FLASH_HDPCR) | 101 |
| 3.7.21 | FLASH HDP extension register (FLASH_HDPEXTR) | 101 |
| 3.7.22 | FLASH register map | 103 |
| 4 | Power control (PWR) | 105 |
| 4.1 | Power supplies | 105 |
| 4.1.1 | Independent analog peripherals supply | 106 |
| 4.1.2 | Independent USB transceivers supply | 107 |
| 4.1.3 | Independent LCD supply | 107 |
| 4.1.4 | Battery backup domain | 108 |
| 4.1.5 | Voltage regulator | 109 |
| 4.1.6 | Dynamic voltage scaling management | 110 |

| | | |
|--------|---|-----|
| 4.2 | Power supply supervisor | 110 |
| 4.2.1 | Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR) | 110 |
| 4.2.2 | Programmable voltage detector (PVD) | 111 |
| 4.2.3 | Peripheral Voltage Monitoring (PVM) | 112 |
| 4.3 | Low-power modes | 113 |
| 4.3.1 | Run mode | 119 |
| 4.3.2 | Low-power run mode (LP run) | 119 |
| 4.3.3 | Low-power modes | 120 |
| 4.3.4 | Sleep mode | 121 |
| 4.3.5 | Low-power sleep mode (LP sleep) | 122 |
| 4.3.6 | Stop 0 mode | 123 |
| 4.3.7 | Stop 1 mode | 125 |
| 4.3.8 | Stop 2 mode | 126 |
| 4.3.9 | Standby mode | 128 |
| 4.3.10 | Shutdown mode | 131 |
| 4.3.11 | Auto-wakeup from low-power mode | 132 |
| 4.4 | PWR registers | 133 |
| 4.4.1 | Power control register 1 (PWR_CR1) | 133 |
| 4.4.2 | Power control register 2 (PWR_CR2) | 134 |
| 4.4.3 | Power control register 3 (PWR_CR3) | 135 |
| 4.4.4 | Power control register 4 (PWR_CR4) | 137 |
| 4.4.5 | Power status register 1 (PWR_SR1) | 138 |
| 4.4.6 | Power status register 2 (PWR_SR2) | 139 |
| 4.4.7 | Power status clear register (PWR_SCR) | 141 |
| 4.4.8 | Power Port A pull-up control register (PWR_PUCRA) | 142 |
| 4.4.9 | Power Port A pull-down control register (PWR_PDCRA) | 142 |
| 4.4.10 | Power Port B pull-up control register (PWR_PUCRB) | 143 |
| 4.4.11 | Power Port B pull-down control register (PWR_PDCRB) | 144 |
| 4.4.12 | Power Port C pull-up control register (PWR_PUCRC) | 144 |
| 4.4.13 | Power Port C pull-down control register (PWR_PDCRC) | 145 |
| 4.4.14 | Power Port D pull-up control register (PWR_PUCRD) | 145 |
| 4.4.15 | Power Port D pull-down control register (PWR_PDCRD) | 146 |
| 4.4.16 | Power Port E pull-up control register (PWR_PUCRE) | 146 |
| 4.4.17 | Power Port E pull-down control register (PWR_PDCRE) | 147 |
| 4.4.18 | Power Port F pull-up control register (PWR_PUCRF) | 148 |
| 4.4.19 | Power Port F pull-down control register (PWR_PDCRF) | 148 |

| | | |
|----------|---|------------|
| 4.4.20 | PWR register map and reset value table | 149 |
| 5 | Reset and clock control (RCC) | 151 |
| 5.1 | Reset | 151 |
| 5.1.1 | Power reset | 151 |
| 5.1.2 | System reset | 151 |
| 5.1.3 | RTC domain reset | 153 |
| 5.2 | Clocks | 154 |
| 5.2.1 | HSE clock | 158 |
| 5.2.2 | HSI16 clock | 159 |
| 5.2.3 | HSI48 clock | 160 |
| 5.2.4 | MSI clock | 160 |
| 5.2.5 | PLL | 161 |
| 5.2.6 | LSE clock | 162 |
| 5.2.7 | LSI clock | 162 |
| 5.2.8 | System clock (SYSCLK) selection | 162 |
| 5.2.9 | Clock source frequency versus voltage scaling | 163 |
| 5.2.10 | Clock security system (CSS) | 163 |
| 5.2.11 | Clock security system for LSE clock (LSECSS) | 164 |
| 5.2.12 | ADC clock | 164 |
| 5.2.13 | RTC clock | 164 |
| 5.2.14 | Timer clock | 165 |
| 5.2.15 | Watchdog clock | 165 |
| 5.2.16 | Clock-out capability | 165 |
| 5.2.17 | Internal/external clock measurement with TIM16 | 166 |
| 5.2.18 | Peripheral clock enable registers | 167 |
| 5.3 | Low-power modes | 167 |
| 5.4 | RCC registers | 169 |
| 5.4.1 | Clock control register (RCC_CR) | 169 |
| 5.4.2 | Internal clock sources calibration register (RCC_ICSCR) | 172 |
| 5.4.3 | Clock configuration register (RCC_CFGR) | 173 |
| 5.4.4 | PLL configuration register (RCC_PLLCFGR) | 176 |
| 5.4.5 | Clock interrupt enable register (RCC_CIER) | 178 |
| 5.4.6 | Clock interrupt flag register (RCC_CIFR) | 180 |
| 5.4.7 | Clock interrupt clear register (RCC_CICR) | 181 |
| 5.4.8 | AHB peripheral reset register (RCC_AHBRSTR) | 182 |
| 5.4.9 | I/O port reset register (RCC_IOPRSTR) | 184 |

| | | |
|----------|---|------------|
| 5.4.10 | APB peripheral reset register 1 (RCC_APBRSTR1) | 185 |
| 5.4.11 | APB peripheral reset register 2 (RCC_APBRSTR2) | 188 |
| 5.4.12 | AHB peripheral clock enable register (RCC_AHBENR) | 189 |
| 5.4.13 | I/O port clock enable register (RCC_IOPENR) | 190 |
| 5.4.14 | Debug configuration register (RCC_DBGCFGR) | 191 |
| 5.4.15 | APB peripheral clock enable register 1 (RCC_APBENR1) | 192 |
| 5.4.16 | APB peripheral clock enable register 2(RCC_APBENR2) | 195 |
| 5.4.17 | AHB peripheral clock enable in Sleep/Stop mode register (RCC_AHBSMENR) | 196 |
| 5.4.18 | I/O port in Sleep mode clock enable register (RCC_IOPSMENR) | 197 |
| 5.4.19 | APB peripheral clock enable in Sleep/Stop mode register 1 (RCC_APBSMENR1) | 198 |
| 5.4.20 | APB peripheral clock enable in Sleep/Stop mode register 2 (RCC_APBSMENR2) | 201 |
| 5.4.21 | Peripherals independent clock configuration register (RCC_CCIPR) | 202 |
| 5.4.22 | RTC domain control register (RCC_BDCR) | 205 |
| 5.4.23 | Control/status register (RCC_CSR) | 207 |
| 5.4.24 | RCC clock recovery RC register (RCC_CRRCR) | 209 |
| 5.4.25 | RCC register map | 210 |
| 6 | Clock recovery system (CRS) | 214 |
| 6.1 | CRS introduction | 214 |
| 6.2 | CRS main features | 214 |
| 6.3 | CRS implementation | 214 |
| 6.4 | CRS functional description | 215 |
| 6.4.1 | CRS block diagram | 215 |
| 6.4.2 | Synchronization input | 215 |
| 6.4.3 | Frequency error measurement | 216 |
| 6.4.4 | Frequency error evaluation and automatic trimming | 217 |
| 6.4.5 | CRS initialization and configuration | 217 |
| 6.5 | CRS in low-power modes | 218 |
| 6.6 | CRS interrupts | 218 |
| 6.7 | CRS registers | 219 |
| 6.7.1 | CRS control register (CRS_CR) | 219 |
| 6.7.2 | CRS configuration register (CRS_CFGR) | 220 |
| 6.7.3 | CRS interrupt and status register (CRS_ISR) | 221 |
| 6.7.4 | CRS interrupt flag clear register (CRS_ICR) | 223 |

| | | |
|----------|---|------------|
| 6.7.5 | CRS register map | 223 |
| 7 | General-purpose I/Os (GPIO) | 225 |
| 7.1 | Introduction | 225 |
| 7.2 | GPIO main features | 225 |
| 7.3 | GPIO functional description | 225 |
| 7.3.1 | General-purpose I/O (GPIO) | 227 |
| 7.3.2 | I/O pin alternate function multiplexer and mapping | 228 |
| 7.3.3 | I/O port control registers | 229 |
| 7.3.4 | I/O port data registers | 229 |
| 7.3.5 | I/O data bitwise handling | 229 |
| 7.3.6 | GPIO locking mechanism | 229 |
| 7.3.7 | I/O alternate function input/output | 230 |
| 7.3.8 | External interrupt/wake-up lines | 230 |
| 7.3.9 | Input configuration | 230 |
| 7.3.10 | Output configuration | 231 |
| 7.3.11 | Alternate function configuration | 232 |
| 7.3.12 | Analog configuration | 233 |
| 7.3.13 | Using the HSE or LSE oscillator pins as GPIOs | 234 |
| 7.3.14 | Using the GPIO pins in the RTC supply domain | 234 |
| 7.3.15 | Using PF3 as GPIO | 234 |
| 7.4 | GPIO registers | 234 |
| 7.4.1 | GPIO port mode register (GPIOx_MODER) (x = A to F) | 234 |
| 7.4.2 | GPIO port output type register (GPIOx_OTYPER) (x = A to F) | 235 |
| 7.4.3 | GPIO port output speed register (GPIOx_OSPEEDR) (x = A to F) | 235 |
| 7.4.4 | GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to F) | 236 |
| 7.4.5 | GPIO port input data register (GPIOx_IDR) (x = A to F) | 236 |
| 7.4.6 | GPIO port output data register (GPIOx_ODR) (x = A to F) | 236 |
| 7.4.7 | GPIO port bit set/reset register (GPIOx_BSRR) (x = A to F) | 237 |
| 7.4.8 | GPIO port configuration lock register (GPIOx_LCKR) (x = A to F) | 237 |
| 7.4.9 | GPIO alternate function low register (GPIOx_AFRL) (x = A to F) | 239 |
| 7.4.10 | GPIO alternate function high register (GPIOx_AFRH) (x = A to F) | 239 |
| 7.4.11 | GPIO port bit reset register (GPIOx_BRR) (x = A to F) | 240 |
| 7.4.12 | GPIO register map | 240 |
| 8 | System configuration controller (SYSCFG) | 242 |
| 8.1 | SYSCFG registers | 242 |

| | | |
|--------|--|-----|
| 8.1.1 | SYSCFG configuration register 1 (SYSCFG_CFGR1) | 242 |
| 8.1.2 | SYSCFG configuration register 2 (SYSCFG_CFGR2) | 245 |
| 8.1.3 | SYSCFG SRAM2 control and status register (SYSCFG_SCSR) | 246 |
| 8.1.4 | SYSCFG SRAM2 key register (SYSCFG_SKR) | 247 |
| 8.1.5 | SYSCFG TSC comparator register (SYSCFG_TSCCR) | 247 |
| 8.1.6 | SYSCFG interrupt line 0 status register (SYSCFG_ITLINE0) | 248 |
| 8.1.7 | SYSCFG interrupt line 1 status register (SYSCFG_ITLINE1) | 248 |
| 8.1.8 | SYSCFG interrupt line 2 status register (SYSCFG_ITLINE2) | 249 |
| 8.1.9 | SYSCFG interrupt line 3 status register (SYSCFG_ITLINE3) | 249 |
| 8.1.10 | SYSCFG interrupt line 4 status register (SYSCFG_ITLINE4) | 250 |
| 8.1.11 | SYSCFG interrupt line 5 status register (SYSCFG_ITLINE5) | 250 |
| 8.1.12 | SYSCFG interrupt line 6 status register (SYSCFG_ITLINE6) | 250 |
| 8.1.13 | SYSCFG interrupt line 7 status register (SYSCFG_ITLINE7) | 251 |
| 8.1.14 | SYSCFG interrupt line 8 status register (SYSCFG_ITLINE8) | 251 |
| 8.1.15 | SYSCFG interrupt line 9 status register (SYSCFG_ITLINE9) | 252 |
| 8.1.16 | SYSCFG interrupt line 10 status register (SYSCFG_ITLINE10) | 252 |
| 8.1.17 | SYSCFG interrupt line 11 status register (SYSCFG_ITLINE11) | 253 |
| 8.1.18 | SYSCFG interrupt line 12 status register (SYSCFG_ITLINE12) | 253 |
| 8.1.19 | SYSCFG interrupt line 13 status register (SYSCFG_ITLINE13) | 254 |
| 8.1.20 | SYSCFG interrupt line 14 status register (SYSCFG_ITLINE14) | 254 |
| 8.1.21 | SYSCFG interrupt line 15 status register (SYSCFG_ITLINE15) | 255 |
| 8.1.22 | SYSCFG interrupt line 16 status register (SYSCFG_ITLINE16) | 255 |
| 8.1.23 | SYSCFG interrupt line 17 status register (SYSCFG_ITLINE17) | 255 |
| 8.1.24 | SYSCFG interrupt line 18 status register (SYSCFG_ITLINE18) | 256 |
| 8.1.25 | SYSCFG interrupt line 19 status register (SYSCFG_ITLINE19) | 256 |
| 8.1.26 | SYSCFG interrupt line 20 status register (SYSCFG_ITLINE20) | 256 |
| 8.1.27 | SYSCFG interrupt line 21 status register (SYSCFG_ITLINE21) | 257 |
| 8.1.28 | SYSCFG interrupt line 22 status register (SYSCFG_ITLINE22) | 257 |
| 8.1.29 | SYSCFG interrupt line 23 status register (SYSCFG_ITLINE23) | 257 |
| 8.1.30 | SYSCFG interrupt line 24 status register (SYSCFG_ITLINE24) | 258 |
| 8.1.31 | SYSCFG interrupt line 25 status register (SYSCFG_ITLINE25) | 258 |
| 8.1.32 | SYSCFG interrupt line 26 status register (SYSCFG_ITLINE26) | 258 |
| 8.1.33 | SYSCFG interrupt line 27 status register (SYSCFG_ITLINE27) | 259 |
| 8.1.34 | SYSCFG interrupt line 28 status register (SYSCFG_ITLINE28) | 259 |
| 8.1.35 | SYSCFG interrupt line 29 status register (SYSCFG_ITLINE29) | 260 |
| 8.1.36 | SYSCFG interrupt line 30 status register (SYSCFG_ITLINE30) | 260 |
| 8.1.37 | SYSCFG interrupt line 31 status register (SYSCFG_ITLINE31) | 260 |

| | | |
|-----------|--|------------|
| 8.1.38 | SYSCFG register map | 261 |
| 9 | Direct memory access controller (DMA) | 264 |
| 9.1 | Introduction | 264 |
| 9.2 | DMA main features | 264 |
| 9.3 | DMA implementation | 265 |
| 9.3.1 | DMA | 265 |
| 9.3.2 | DMA request mapping | 265 |
| 9.4 | DMA functional description | 266 |
| 9.4.1 | DMA block diagram | 266 |
| 9.4.2 | DMA pins and internal signals | 266 |
| 9.4.3 | DMA transfers | 267 |
| 9.4.4 | DMA arbitration | 268 |
| 9.4.5 | DMA channels | 268 |
| 9.4.6 | DMA data width, alignment, and endianness | 272 |
| 9.4.7 | DMA error management | 273 |
| 9.5 | DMA interrupts | 274 |
| 9.6 | DMA registers | 274 |
| 9.6.1 | DMA interrupt status register (DMA_ISR) | 274 |
| 9.6.2 | DMA interrupt flag clear register (DMA_IFCR) | 277 |
| 9.6.3 | DMA channel x configuration register (DMA_CCRx) | 278 |
| 9.6.4 | DMA channel x number of data to transfer register (DMA_CNDTRx) | 281 |
| 9.6.5 | DMA channel x peripheral address register (DMA_CPARx) | 281 |
| 9.6.6 | DMA channel x memory address register (DMA_CMARx) | 282 |
| 9.6.7 | DMA register map | 282 |
| 10 | DMA request multiplexer (DMAMUX) | 285 |
| 10.1 | Introduction | 285 |
| 10.2 | DMAMUX main features | 286 |
| 10.3 | DMAMUX implementation | 286 |
| 10.3.1 | DMAMUX instantiation | 286 |
| 10.3.2 | DMAMUX mapping | 286 |
| 10.4 | DMAMUX functional description | 289 |
| 10.4.1 | DMAMUX block diagram | 289 |
| 10.4.2 | DMAMUX signals | 290 |
| 10.4.3 | DMAMUX channels | 290 |

| | | |
|-----------|--|------------|
| 10.4.4 | DMAMUX request line multiplexer | 290 |
| 10.4.5 | DMAMUX request generator | 293 |
| 10.5 | DMAMUX interrupts | 294 |
| 10.6 | DMAMUX registers | 295 |
| 10.6.1 | DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR) | 295 |
| 10.6.2 | DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR) | 296 |
| 10.6.3 | DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR) | 296 |
| 10.6.4 | DMAMUX request generator channel x configuration register (DMAMUX_RGxCR) | 297 |
| 10.6.5 | DMAMUX request generator interrupt status register (DMAMUX_RGSR) | 298 |
| 10.6.6 | DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR) | 298 |
| 10.6.7 | DMAMUX register map | 299 |
| 11 | Nested vectored interrupt controller (NVIC) | 301 |
| 11.1 | Main features | 301 |
| 11.2 | SysTick calibration value register | 301 |
| 11.3 | Interrupt and exception vectors | 301 |
| 12 | Extended interrupt and event controller (EXTI) | 304 |
| 12.1 | EXTI main features | 304 |
| 12.2 | EXTI block diagram | 304 |
| 12.2.1 | EXTI connections between peripherals and CPU | 306 |
| 12.3 | EXTI functional description | 306 |
| 12.3.1 | EXTI configurable event input wake-up | 307 |
| 12.3.2 | EXTI direct event input wake-up | 308 |
| 12.3.3 | EXTI mux | 308 |
| 12.4 | EXTI functional behavior | 310 |
| 12.5 | EXTI registers | 311 |
| 12.5.1 | EXTI rising trigger selection register (EXTI_RTSR1) | 311 |
| 12.5.2 | EXTI falling trigger selection register 1 (EXTI_FTSR1) | 312 |
| 12.5.3 | EXTI software interrupt event register 1 (EXTI_SWIER1) | 312 |
| 12.5.4 | EXTI rising edge pending register 1 (EXTI_RPR1) | 313 |
| 12.5.5 | EXTI falling edge pending register 1 (EXTI_FPR1) | 313 |

| | | |
|-----------|---|------------|
| 12.5.6 | EXTI external interrupt selection register x (EXTI_EXTICRx) | 314 |
| 12.5.7 | EXTI CPU wake-up with interrupt mask register (EXTI_IMR1) | 315 |
| 12.5.8 | EXTI CPU wake-up with event mask register (EXTI_EMR1) | 315 |
| 12.5.9 | EXTI CPU wake-up with interrupt mask register (EXTI_IMR2) | 316 |
| 12.5.10 | EXTI CPU wake-up with event mask register (EXTI_EMR2) | 316 |
| 12.5.11 | EXTI register map | 317 |
| 13 | Cyclic redundancy check calculation unit (CRC) | 319 |
| 13.1 | Introduction | 319 |
| 13.2 | CRC main features | 319 |
| 13.3 | CRC functional description | 320 |
| 13.3.1 | CRC block diagram | 320 |
| 13.3.2 | CRC internal signals | 320 |
| 13.3.3 | CRC operation | 320 |
| 13.4 | CRC registers | 322 |
| 13.4.1 | CRC data register (CRC_DR) | 322 |
| 13.4.2 | CRC independent data register (CRC_IDR) | 322 |
| 13.4.3 | CRC control register (CRC_CR) | 323 |
| 13.4.4 | CRC initial value (CRC_INIT) | 324 |
| 13.4.5 | CRC polynomial (CRC_POL) | 324 |
| 13.4.6 | CRC register map | 325 |
| 14 | Analog-to-digital converter (ADC) | 326 |
| 14.1 | Introduction | 326 |
| 14.2 | ADC main features | 327 |
| 14.3 | ADC implementation | 328 |
| 14.4 | ADC functional description | 329 |
| 14.4.1 | ADC pins and internal signals | 330 |
| 14.4.2 | ADC voltage regulator (ADVREGEN) | 331 |
| 14.4.3 | Calibration (ADCAL) | 331 |
| 14.4.4 | ADC on-off control (ADEN, ADDIS, ADRDY) | 333 |
| 14.4.5 | ADC clock (CKMODE, PRESC[3:0]) | 334 |
| 14.4.6 | ADC connectivity | 336 |
| 14.4.7 | Configuring the ADC | 336 |
| 14.4.8 | Channel selection (CHSEL, SCANDIR, CHSELRMOD) | 337 |
| 14.4.9 | Programmable sampling time (SMPx[2:0]) | 338 |

| | | |
|---------|--|-----|
| 14.4.10 | Single conversion mode (CONT = 0) | 338 |
| 14.4.11 | Continuous conversion mode (CONT = 1) | 339 |
| 14.4.12 | Starting conversions (ADSTART) | 339 |
| 14.4.13 | Timings | 340 |
| 14.4.14 | Stopping an ongoing conversion (ADSTP) | 341 |
| 14.5 | Conversion on external trigger and trigger polarity (EXTSEL, EXTEN) . | 342 |
| 14.5.1 | Discontinuous mode (DISCEN) | 342 |
| 14.5.2 | Programmable resolution (RES) - Fast conversion mode | 343 |
| 14.5.3 | End of conversion, end of sampling phase (EOC, EOSMP flags) | 344 |
| 14.5.4 | End of conversion sequence (EOS flag) | 344 |
| 14.5.5 | Example timing diagrams (single/continuous modes hardware/software triggers) | 344 |
| 14.5.6 | Low frequency trigger mode | 346 |
| 14.6 | Data management | 347 |
| 14.6.1 | Data register and data alignment (ADC_DR, ALIGN) | 347 |
| 14.6.2 | ADC overrun (OVR, OVRMOD) | 347 |
| 14.6.3 | Managing a sequence of data converted without using the DMA | 348 |
| 14.6.4 | Managing converted data without using the DMA without overrun | 348 |
| 14.6.5 | Managing converted data using the DMA | 348 |
| 14.7 | Low-power features | 350 |
| 14.7.1 | Wait mode conversion | 350 |
| 14.7.2 | Auto-off mode (AUTOFF) | 350 |
| 14.8 | Analog window watchdogs | 352 |
| 14.8.1 | Description of analog watchdog 1 | 352 |
| 14.8.2 | Description of analog watchdog 2 and 3 | 353 |
| 14.8.3 | ADC_AWDx_OUT output signal generation | 354 |
| 14.8.4 | Analog Watchdog threshold control | 355 |
| 14.9 | Oversampler | 356 |
| 14.9.1 | ADC operating modes supported when oversampling | 358 |
| 14.9.2 | Analog watchdog | 358 |
| 14.9.3 | Triggered mode | 358 |
| 14.10 | Temperature sensor and internal reference voltage | 359 |
| 14.11 | Battery voltage monitoring | 361 |
| 14.12 | ADC interrupts | 362 |
| 14.13 | ADC registers | 363 |
| 14.13.1 | ADC interrupt and status register (ADC_ISR) | 363 |

| | | |
|-----------|---|------------|
| 14.13.2 | ADC interrupt enable register (ADC_IER) | 365 |
| 14.13.3 | ADC control register (ADC_CR) | 367 |
| 14.13.4 | ADC configuration register 1 (ADC_CFGR1) | 369 |
| 14.13.5 | ADC configuration register 2 (ADC_CFGR2) | 372 |
| 14.13.6 | ADC sampling time register (ADC_SMPR) | 373 |
| 14.13.7 | ADC watchdog threshold register (ADC_AWD1TR) | 374 |
| 14.13.8 | ADC watchdog threshold register (ADC_AWD2TR) | 375 |
| 14.13.9 | ADC channel selection register (ADC_CHSELR) | 376 |
| 14.13.10 | ADC channel selection register [alternate] (ADC_CHSELR) | 377 |
| 14.13.11 | ADC watchdog threshold register (ADC_AWD3TR) | 379 |
| 14.13.12 | ADC data register (ADC_DR) | 379 |
| 14.13.13 | ADC analog watchdog 2 configuration register (ADC_AWD2CR) | 380 |
| 14.13.14 | ADC Analog Watchdog 3 Configuration register (ADC_AWD3CR) | 380 |
| 14.13.15 | ADC calibration factor (ADC_CALFACT) | 381 |
| 14.13.16 | ADC common configuration register (ADC_CCR) | 381 |
| 14.14 | ADC register map | 382 |
| 15 | Digital-to-analog converter (DAC) | 385 |
| 15.1 | Introduction | 385 |
| 15.2 | DAC main features | 385 |
| 15.3 | DAC implementation | 386 |
| 15.4 | DAC functional description | 386 |
| 15.4.1 | DAC block diagram | 386 |
| 15.4.2 | DAC pins and internal signals | 387 |
| 15.4.3 | DAC channel enable | 388 |
| 15.4.4 | DAC data format | 388 |
| 15.4.5 | DAC conversion | 389 |
| 15.4.6 | DAC output voltage | 389 |
| 15.4.7 | DAC trigger selection | 390 |
| 15.4.8 | DMA requests | 390 |
| 15.4.9 | Noise generation | 390 |
| 15.4.10 | Triangle-wave generation | 392 |
| 15.4.11 | DAC channel modes | 393 |
| 15.4.12 | DAC channel buffer calibration | 396 |
| 15.4.13 | DAC channel conversion modes | 397 |
| 15.5 | DAC in low-power modes | 398 |

| | | |
|-----------|---|------------|
| 15.6 | DAC interrupts | 399 |
| 15.7 | DAC registers | 400 |
| 15.7.1 | DAC control register (DAC_CR) | 400 |
| 15.7.2 | DAC software trigger register (DAC_SWTRGR) | 402 |
| 15.7.3 | DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) | 402 |
| 15.7.4 | DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) | 403 |
| 15.7.5 | DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) | 403 |
| 15.7.6 | DAC channel1 data output register (DAC_DOR1) | 404 |
| 15.7.7 | DAC status register (DAC_SR) | 404 |
| 15.7.8 | DAC calibration control register (DAC_CCR) | 405 |
| 15.7.9 | DAC mode control register (DAC_MCR) | 405 |
| 15.7.10 | DAC channel1 sample and hold sample time register (DAC_SHSR1) | 406 |
| 15.7.11 | DAC sample and hold time register (DAC_SHHR) | 407 |
| 15.7.12 | DAC sample and hold refresh time register (DAC_SHRR) | 407 |
| 15.7.13 | DAC register map | 408 |
| 16 | Voltage reference buffer (VREFBUF) | 410 |
| 16.1 | Introduction | 410 |
| 16.2 | VREFBUF functional description | 410 |
| 16.3 | VREFBUF registers | 411 |
| 16.3.1 | VREFBUF control and status register (VREFBUF_CSR) | 411 |
| 16.3.2 | VREFBUF calibration control register (VREFBUF_CCR) | 411 |
| 16.3.3 | VREFBUF register map | 412 |
| 17 | Comparator (COMP) | 413 |
| 17.1 | Introduction | 413 |
| 17.2 | COMP main features | 413 |
| 17.3 | COMP functional description | 414 |
| 17.3.1 | COMP block diagram | 414 |
| 17.3.2 | COMP pins and internal signals | 414 |
| 17.3.3 | COMP reset and clocks | 416 |
| 17.3.4 | Comparator LOCK mechanism | 416 |
| 17.3.5 | Window comparator | 416 |
| 17.3.6 | Hysteresis | 417 |

| | | |
|-----------|--|------------|
| 17.3.7 | Comparator output blanking function | 417 |
| 17.3.8 | COMP power and speed modes | 418 |
| 17.4 | COMP low-power modes | 418 |
| 17.5 | COMP interrupts | 419 |
| 17.6 | COMP registers | 420 |
| 17.6.1 | Comparator 1 control and status register (COMP1_CSR) | 420 |
| 17.6.2 | Comparator 2 control and status register (COMP2_CSR) | 421 |
| 17.6.3 | COMP register map | 424 |
| 18 | Operational amplifiers (OPAMP) | 425 |
| 18.1 | Introduction | 425 |
| 18.2 | OPAMP main features | 425 |
| 18.3 | OPAMP functional description | 425 |
| 18.3.1 | OPAMP reset and clocks | 425 |
| 18.3.2 | Initial configuration | 426 |
| 18.3.3 | Signal routing | 426 |
| 18.3.4 | OPAMP modes | 426 |
| 18.3.5 | Calibration | 430 |
| 18.4 | OPAMP low-power modes | 432 |
| 18.5 | OPAMP registers | 433 |
| 18.5.1 | OPAMP1 control/status register (OPAMP1_CSR) | 433 |
| 18.5.2 | OPAMP1 offset trimming register in normal mode (OPAMP1_OTR) | 434 |
| 18.5.3 | OPAMP1 offset trimming register in low-power mode (OPAMP1_LPOTR) | 434 |
| 18.5.4 | OPAMP register map | 435 |
| 19 | Liquid crystal display controller (LCD) | 436 |
| 19.1 | LCD introduction | 436 |
| 19.2 | LCD main features | 436 |
| 19.3 | LCD functional description | 438 |
| 19.3.1 | General description | 438 |
| 19.3.2 | Frequency generator | 439 |
| 19.3.3 | Common driver | 440 |
| 19.3.4 | Segment driver | 443 |
| 19.3.5 | Voltage generator and contrast control | 447 |
| 19.3.6 | Double-buffer memory | 449 |

| | | |
|-----------|---|------------|
| 19.3.7 | COM and SEG multiplexing | 450 |
| 19.3.8 | Flowchart | 451 |
| 19.4 | LCD low-power modes | 452 |
| 19.5 | LCD interrupts | 452 |
| 19.6 | LCD registers | 453 |
| 19.6.1 | LCD control register (LCD_CR) | 453 |
| 19.6.2 | LCD frame control register (LCD_FCR) | 454 |
| 19.6.3 | LCD status register (LCD_SR) | 456 |
| 19.6.4 | LCD clear register (LCD_CLR) | 458 |
| 19.6.5 | LCD display memory (LCD_RAMx) | 458 |
| 19.6.6 | LCD display memory (LCD_RAMx) | 459 |
| 19.6.7 | LCD display memory (LCD_RAMx) | 459 |
| 19.6.8 | LCD register map | 459 |
| 20 | Touch sensing controller (TSC) | 462 |
| 20.1 | Introduction | 462 |
| 20.2 | TSC main features | 462 |
| 20.3 | TSC implementation | 463 |
| 20.4 | TSC functional description | 464 |
| 20.4.1 | TSC block diagram | 464 |
| 20.4.2 | Surface charge transfer acquisition overview | 464 |
| 20.4.3 | Reset and clocks | 467 |
| 20.4.4 | Charge transfer acquisition sequence | 467 |
| 20.4.5 | Spread spectrum feature | 468 |
| 20.4.6 | Max count error | 469 |
| 20.4.7 | Sampling capacitor I/O and channel I/O mode selection | 469 |
| 20.4.8 | Acquisition mode | 470 |
| 20.4.9 | I/O hysteresis and analog switch control | 470 |
| 20.5 | TSC low-power modes | 471 |
| 20.5.1 | Comparator usage overview | 471 |
| 20.6 | TSC interrupts | 474 |
| 20.7 | TSC registers | 474 |
| 20.7.1 | TSC control register (TSC_CR) | 474 |
| 20.7.2 | TSC interrupt enable register (TSC_IER) | 477 |
| 20.7.3 | TSC interrupt clear register (TSC_ICR) | 478 |
| 20.7.4 | TSC interrupt status register (TSC_ISR) | 478 |

| | | |
|-----------|--|------------|
| 20.7.5 | TSC I/O hysteresis control register (TSC_IOHCR) | 479 |
| 20.7.6 | TSC I/O analog switch control register (TSC_IOASCR) | 479 |
| 20.7.7 | TSC I/O sampling control register (TSC_IOSCR) | 480 |
| 20.7.8 | TSC I/O channel control register (TSC_IOCCR) | 480 |
| 20.7.9 | TSC I/O group control status register (TSC_IOGCSR) | 481 |
| 20.7.10 | TSC I/O group x counter register (TSC_IOGxCR) | 481 |
| 20.7.11 | TSC register map | 482 |
| 21 | True random number generator (RNG) | 484 |
| 21.1 | Introduction | 484 |
| 21.2 | RNG main features | 484 |
| 21.3 | RNG functional description | 485 |
| 21.3.1 | RNG block diagram | 485 |
| 21.3.2 | RNG internal signals | 485 |
| 21.3.3 | Random number generation | 486 |
| 21.3.4 | RNG initialization | 489 |
| 21.3.5 | RNG operation | 490 |
| 21.3.6 | RNG clocking | 491 |
| 21.3.7 | Error management | 491 |
| 21.3.8 | RNG low-power use | 492 |
| 21.4 | RNG interrupts | 493 |
| 21.5 | RNG processing time | 494 |
| 21.6 | RNG entropy source validation | 494 |
| 21.6.1 | Introduction | 494 |
| 21.6.2 | Validation conditions | 495 |
| 21.7 | RNG registers | 496 |
| 21.7.1 | RNG control register (RNG_CR) | 496 |
| 21.7.2 | RNG status register (RNG_SR) | 498 |
| 21.7.3 | RNG data register (RNG_DR) | 499 |
| 21.7.4 | RNG noise source control register (RNG_NSSCR) | 499 |
| 21.7.5 | RNG health test control register (RNG_HTCR) | 500 |
| 21.7.6 | RNG register map | 501 |
| 22 | AES hardware accelerator (AES) | 502 |
| 22.1 | Introduction | 502 |

| | | |
|---------|---|-----|
| 22.2 | AES main features | 502 |
| 22.3 | AES implementation | 502 |
| 22.4 | AES functional description | 503 |
| 22.4.1 | AES block diagram | 503 |
| 22.4.2 | AES internal signals | 503 |
| 22.4.3 | AES cryptographic core | 503 |
| 22.4.4 | AES procedure to perform a cipher operation | 509 |
| 22.4.5 | AES decryption round key preparation | 512 |
| 22.4.6 | AES ciphertext stealing and data padding | 512 |
| 22.4.7 | AES task suspend and resume | 513 |
| 22.4.8 | AES basic chaining modes (ECB, CBC) | 513 |
| 22.4.9 | AES counter (CTR) mode | 518 |
| 22.4.10 | AES Galois/counter mode (GCM) | 520 |
| 22.4.11 | AES Galois message authentication code (GMAC) | 525 |
| 22.4.12 | AES counter with CBC-MAC (CCM) | 527 |
| 22.4.13 | AES data registers and data swapping | 532 |
| 22.4.14 | AES key registers | 534 |
| 22.4.15 | AES initialization vector registers | 534 |
| 22.4.16 | AES DMA interface | 535 |
| 22.4.17 | AES error management | 536 |
| 22.5 | AES interrupts | 537 |
| 22.6 | AES processing latency | 537 |
| 22.7 | AES registers | 538 |
| 22.7.1 | AES control register (AES_CR) | 538 |
| 22.7.2 | AES status register (AES_SR) | 540 |
| 22.7.3 | AES data input register (AES_DINR) | 542 |
| 22.7.4 | AES data output register (AES_DOUTR) | 542 |
| 22.7.5 | AES key register 0 (AES_KEYR0) | 543 |
| 22.7.6 | AES key register 1 (AES_KEYR1) | 543 |
| 22.7.7 | AES key register 2 (AES_KEYR2) | 544 |
| 22.7.8 | AES key register 3 (AES_KEYR3) | 544 |
| 22.7.9 | AES initialization vector register 0 (AES_IVR0) | 544 |
| 22.7.10 | AES initialization vector register 1 (AES_IVR1) | 545 |
| 22.7.11 | AES initialization vector register 2 (AES_IVR2) | 545 |
| 22.7.12 | AES initialization vector register 3 (AES_IVR3) | 545 |
| 22.7.13 | AES key register 4 (AES_KEYR4) | 546 |

| | | |
|-----------|---|------------|
| 22.7.14 | AES key register 5 (AES_KEYR5) | 546 |
| 22.7.15 | AES key register 6 (AES_KEYR6) | 546 |
| 22.7.16 | AES key register 7 (AES_KEYR7) | 547 |
| 22.7.17 | AES suspend registers (AES_SUSPxR) | 547 |
| 22.7.18 | AES register map | 548 |
| 23 | Advanced-control timer (TIM1) | 550 |
| 23.1 | TIM1 introduction | 550 |
| 23.2 | TIM1 main features | 551 |
| 23.3 | TIM1 functional description | 553 |
| 23.3.1 | Time-base unit | 553 |
| 23.3.2 | Counter modes | 555 |
| 23.3.3 | Repetition counter | 566 |
| 23.3.4 | External trigger input | 568 |
| 23.3.5 | Clock selection | 569 |
| 23.3.6 | Capture/compare channels | 573 |
| 23.3.7 | Input capture mode | 575 |
| 23.3.8 | PWM input mode | 576 |
| 23.3.9 | Forced output mode | 577 |
| 23.3.10 | Output compare mode | 578 |
| 23.3.11 | PWM mode | 579 |
| 23.3.12 | Asymmetric PWM mode | 582 |
| 23.3.13 | Combined PWM mode | 583 |
| 23.3.14 | Combined 3-phase PWM mode | 584 |
| 23.3.15 | Complementary outputs and dead-time insertion | 585 |
| 23.3.16 | Using the break function | 587 |
| 23.3.17 | Bidirectional break inputs | 593 |
| 23.3.18 | Clearing the OCxREF signal on an external event | 595 |
| 23.3.19 | 6-step PWM generation | 596 |
| 23.3.20 | One-pulse mode | 597 |
| 23.3.21 | Retriggerable one pulse mode | 598 |
| 23.3.22 | Encoder interface mode | 599 |
| 23.3.23 | UIF bit remapping | 601 |
| 23.3.24 | Timer input XOR function | 602 |
| 23.3.25 | Interfacing with Hall sensors | 602 |
| 23.3.26 | Timer synchronization | 605 |
| 23.3.27 | ADC synchronization | 609 |

| | | |
|---------|--|-----|
| 23.3.28 | DMA burst mode | 609 |
| 23.3.29 | Debug mode | 610 |
| 23.4 | TIM1 registers | 611 |
| 23.4.1 | TIM1 control register 1 (TIM1_CR1) | 611 |
| 23.4.2 | TIM1 control register 2 (TIM1_CR2) | 612 |
| 23.4.3 | TIM1 slave mode control register (TIM1_SMCR) | 615 |
| 23.4.4 | TIM1 DMA/interrupt enable register (TIM1_DIER) | 617 |
| 23.4.5 | TIM1 status register (TIM1_SR) | 619 |
| 23.4.6 | TIM1 event generation register (TIM1_EGR) | 621 |
| 23.4.7 | TIM1 capture/compare mode register 1 (TIM1_CCMR1) | 622 |
| 23.4.8 | TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) | 623 |
| 23.4.9 | TIM1 capture/compare mode register 2 (TIM1_CCMR2) | 626 |
| 23.4.10 | TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) | 627 |
| 23.4.11 | TIM1 capture/compare enable register (TIM1_CCER) | 629 |
| 23.4.12 | TIM1 counter (TIM1_CNT) | 632 |
| 23.4.13 | TIM1 prescaler (TIM1_PSC) | 632 |
| 23.4.14 | TIM1 auto-reload register (TIM1_ARR) | 632 |
| 23.4.15 | TIM1 repetition counter register (TIM1_RCR) | 633 |
| 23.4.16 | TIM1 capture/compare register 1 (TIM1_CCR1) | 633 |
| 23.4.17 | TIM1 capture/compare register 2 (TIM1_CCR2) | 634 |
| 23.4.18 | TIM1 capture/compare register 3 (TIM1_CCR3) | 634 |
| 23.4.19 | TIM1 capture/compare register 4 (TIM1_CCR4) | 635 |
| 23.4.20 | TIM1 break and dead-time register (TIM1_BDTR) | 635 |
| 23.4.21 | TIM1 DMA control register (TIM1_DCR) | 639 |
| 23.4.22 | TIM1 DMA address for full transfer (TIM1_DMAR) | 640 |
| 23.4.23 | TIM1 option register 1 (TIM1_OR1) | 641 |
| 23.4.24 | TIM1 capture/compare mode register 3 (TIM1_CCMR3) | 641 |

| | | |
|-----------|--|------------|
| 23.4.25 | TIM1 capture/compare register 5 (TIM1_CCR5) | 642 |
| 23.4.26 | TIM1 capture/compare register 6 (TIM1_CCR6) | 643 |
| 23.4.27 | TIM1 alternate function option register 1 (TIM1_AF1) | 644 |
| 23.4.28 | TIM1 Alternate function register 2 (TIM1_AF2) | 645 |
| 23.4.29 | TIM1 timer input selection register (TIM1_TISEL) | 647 |
| 23.4.30 | TIM1 register map | 648 |
| 24 | General-purpose timers (TIM2/TIM3) | 651 |
| 24.1 | TIM2/TIM3 introduction | 651 |
| 24.2 | TIM2/TIM3 main features | 651 |
| 24.3 | TIM2/TIM3 functional description | 653 |
| 24.3.1 | Time-base unit | 653 |
| 24.3.2 | Counter modes | 655 |
| 24.3.3 | Clock selection | 665 |
| 24.3.4 | Capture/Compare channels | 669 |
| 24.3.5 | Input capture mode | 671 |
| 24.3.6 | PWM input mode | 672 |
| 24.3.7 | Forced output mode | 673 |
| 24.3.8 | Output compare mode | 673 |
| 24.3.9 | PWM mode | 674 |
| 24.3.10 | Asymmetric PWM mode | 678 |
| 24.3.11 | Combined PWM mode | 678 |
| 24.3.12 | Clearing the OCxREF signal on an external event | 679 |
| 24.3.13 | One-pulse mode | 681 |
| 24.3.14 | Retriggerable one pulse mode | 682 |
| 24.3.15 | Encoder interface mode | 683 |
| 24.3.16 | UIF bit remapping | 685 |
| 24.3.17 | Timer input XOR function | 685 |
| 24.3.18 | Timers and external trigger synchronization | 686 |
| 24.3.19 | Timer synchronization | 689 |
| 24.3.20 | DMA burst mode | 694 |
| 24.3.21 | Debug mode | 695 |
| 24.4 | TIM2/TIM3 registers | 696 |
| 24.4.1 | TIMx control register 1 (TIMx_CR1)(x = 2 to 3) | 696 |
| 24.4.2 | TIMx control register 2 (TIMx_CR2)(x = 2 to 3) | 697 |

| | | |
|-----------|---|------------|
| 24.4.3 | TIMx slave mode control register (TIMx_SMCR)(x = 2 to 3) | 699 |
| 24.4.4 | TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 3) | 702 |
| 24.4.5 | TIMx status register (TIMx_SR)(x = 2 to 3) | 703 |
| 24.4.6 | TIMx event generation register (TIMx_EGR)(x = 2 to 3) | 705 |
| 24.4.7 | TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 3) . . | 706 |
| 24.4.8 | TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 3) | 708 |
| 24.4.9 | TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 3) . . | 710 |
| 24.4.10 | TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 3) | 711 |
| 24.4.11 | TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 3) | 712 |
| 24.4.12 | TIMx counter (TIMx_CNT)(x = 2 to 3) | 713 |
| 24.4.13 | TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3) | 714 |
| 24.4.14 | TIMx prescaler (TIMx_PSC)(x = 2 to 3) | 714 |
| 24.4.15 | TIMx auto-reload register (TIMx_ARR)(x = 2 to 3) | 715 |
| 24.4.16 | TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 3) | 715 |
| 24.4.17 | TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 3) | 716 |
| 24.4.18 | TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 3) | 716 |
| 24.4.19 | TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 3) | 717 |
| 24.4.20 | TIMx DMA control register (TIMx_DCR)(x = 2 to 3) | 718 |
| 24.4.21 | TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 3) | 718 |
| 24.4.22 | TIM2 option register 1 (TIM2_OR1) | 718 |
| 24.4.23 | TIM3 option register 1 (TIM3_OR1) | 719 |
| 24.4.24 | TIM2 alternate function option register 1 (TIM2_AF1) | 719 |
| 24.4.25 | TIM3 alternate function option register 1 (TIM3_AF1) | 720 |
| 24.4.26 | TIM2 timer input selection register (TIM2_TISEL) | 720 |
| 24.4.27 | TIM3 timer input selection register (TIM3_TISEL) | 721 |
| 24.4.28 | TIMx register map | 723 |
| 25 | Basic timers (TIM6/TIM7) | 726 |
| 25.1 | TIM6/TIM7 introduction | 726 |
| 25.2 | TIM6/TIM7 main features | 726 |
| 25.3 | TIM6/TIM7 functional description | 727 |
| 25.3.1 | Time-base unit | 727 |
| 25.3.2 | Counting mode | 729 |
| 25.3.3 | UIF bit remapping | 732 |
| 25.3.4 | Clock source | 732 |

| | | |
|-----------|--|------------|
| 25.3.5 | Debug mode | 733 |
| 25.4 | TIM6/TIM7 registers | 733 |
| 25.4.1 | TIMx control register 1 (TIMx_CR1)(x = 6 to 7) | 733 |
| 25.4.2 | TIMx control register 2 (TIMx_CR2)(x = 6 to 7) | 735 |
| 25.4.3 | TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) | 735 |
| 25.4.4 | TIMx status register (TIMx_SR)(x = 6 to 7) | 736 |
| 25.4.5 | TIMx event generation register (TIMx_EGR)(x = 6 to 7) | 736 |
| 25.4.6 | TIMx counter (TIMx_CNT)(x = 6 to 7) | 736 |
| 25.4.7 | TIMx prescaler (TIMx_PSC)(x = 6 to 7) | 737 |
| 25.4.8 | TIMx auto-reload register (TIMx_ARR)(x = 6 to 7) | 737 |
| 25.4.9 | TIMx register map | 738 |
| 26 | General-purpose timers (TIM15/TIM16) | 739 |
| 26.1 | TIM15/TIM16 introduction | 739 |
| 26.2 | TIM15 main features | 739 |
| 26.3 | TIM16 main features | 740 |
| 26.4 | TIM15/TIM16 functional description | 743 |
| 26.4.1 | Time-base unit | 743 |
| 26.4.2 | Counter modes | 745 |
| 26.4.3 | Repetition counter | 749 |
| 26.4.4 | Clock selection | 750 |
| 26.4.5 | Capture/compare channels | 752 |
| 26.4.6 | Input capture mode | 754 |
| 26.4.7 | PWM input mode (only for TIM15) | 755 |
| 26.4.8 | Forced output mode | 756 |
| 26.4.9 | Output compare mode | 757 |
| 26.4.10 | PWM mode | 758 |
| 26.4.11 | Combined PWM mode (TIM15 only) | 759 |
| 26.4.12 | Complementary outputs and dead-time insertion | 760 |
| 26.4.13 | Using the break function | 762 |
| 26.4.14 | Bidirectional break inputs | 767 |
| 26.4.15 | 6-step PWM generation | 768 |
| 26.4.16 | One-pulse mode | 770 |
| 26.4.17 | Retriggerable one pulse mode (TIM15 only) | 771 |
| 26.4.18 | UIF bit remapping | 772 |
| 26.4.19 | Timer input XOR function (TIM15 only) | 773 |
| 26.4.20 | External trigger synchronization (TIM15 only) | 774 |

| | | |
|---------|--|-----|
| 26.4.21 | Slave mode – combined reset + trigger mode | 776 |
| 26.4.22 | DMA burst mode | 776 |
| 26.4.23 | Timer synchronization (TIM15) | 778 |
| 26.4.24 | Using timer output as trigger for other timers (TIM16) | 778 |
| 26.4.25 | Debug mode | 778 |
| 26.5 | TIM15 registers | 779 |
| 26.5.1 | TIM15 control register 1 (TIM15_CR1) | 779 |
| 26.5.2 | TIM15 control register 2 (TIM15_CR2) | 780 |
| 26.5.3 | TIM15 slave mode control register (TIM15_SMCR) | 782 |
| 26.5.4 | TIM15 DMA/interrupt enable register (TIM15_DIER) | 783 |
| 26.5.5 | TIM15 status register (TIM15_SR) | 784 |
| 26.5.6 | TIM15 event generation register (TIM15_EGR) | 786 |
| 26.5.7 | TIM15 capture/compare mode register 1 (TIM15_CCMR1) | 787 |
| 26.5.8 | TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) | 788 |
| 26.5.9 | TIM15 capture/compare enable register (TIM15_CCER) | 791 |
| 26.5.10 | TIM15 counter (TIM15_CNT) | 794 |
| 26.5.11 | TIM15 prescaler (TIM15_PSC) | 794 |
| 26.5.12 | TIM15 auto-reload register (TIM15_ARR) | 794 |
| 26.5.13 | TIM15 repetition counter register (TIM15_RCR) | 795 |
| 26.5.14 | TIM15 capture/compare register 1 (TIM15_CCR1) | 795 |
| 26.5.15 | TIM15 capture/compare register 2 (TIM15_CCR2) | 796 |
| 26.5.16 | TIM15 break and dead-time register (TIM15_BDTR) | 796 |
| 26.5.17 | TIM15 DMA control register (TIM15_DCR) | 799 |
| 26.5.18 | TIM15 DMA address for full transfer (TIM15_DMAR) | 799 |
| 26.5.19 | TIM15 alternate register 1 (TIM15_AF1) | 800 |
| 26.5.20 | TIM15 input selection register (TIM15_TISEL) | 801 |
| 26.5.21 | TIM15 register map | 802 |
| 26.6 | TIM16 registers | 804 |
| 26.6.1 | TIM16 control register 1 (TIM16_CR1) | 804 |
| 26.6.2 | TIM16 control register 2 (TIM16_CR2) | 805 |
| 26.6.3 | TIM16 DMA/interrupt enable register (TIM16_DIER) | 806 |
| 26.6.4 | TIM16 status register (TIM16_SR) | 807 |
| 26.6.5 | TIM16 event generation register (TIM16_EGR) | 808 |
| 26.6.6 | TIM16 capture/compare mode register 1 (TIM16_CCMR1) | 809 |
| 26.6.7 | TIM16 capture/compare mode register 1 [alternate] (TIM16_CCMR1) | 810 |
| 26.6.8 | TIM16 capture/compare enable register (TIM16_CCER) | 812 |

| | | |
|-----------|--|------------|
| 26.6.9 | TIM16 counter (TIM16_CNT) | 814 |
| 26.6.10 | TIM16 prescaler (TIM16_PSC) | 815 |
| 26.6.11 | TIM16 auto-reload register (TIM16_ARR) | 815 |
| 26.6.12 | TIM16 repetition counter register (TIM16_RCR) | 816 |
| 26.6.13 | TIM16 capture/compare register 1 (TIM16_CCR1) | 816 |
| 26.6.14 | TIM16 break and dead-time register (TIM16_BDTR) | 817 |
| 26.6.15 | TIM16 DMA control register (TIM16_DCR) | 820 |
| 26.6.16 | TIM16 DMA address for full transfer (TIM16_DMAR) | 820 |
| 26.6.17 | TIM16 alternate function register 1 (TIM16_AF1) | 821 |
| 26.6.18 | TIM16 input selection register (TIM16_TISEL) | 822 |
| 26.6.19 | TIM16 register map | 823 |
| 27 | Low-power timer (LPTIM) | 825 |
| 27.1 | Introduction | 825 |
| 27.2 | LPTIM main features | 825 |
| 27.3 | LPTIM implementation | 826 |
| 27.4 | LPTIM functional description | 827 |
| 27.4.1 | LPTIM block diagram | 827 |
| 27.4.2 | LPTIM pins and internal signals | 827 |
| 27.4.3 | LPTIM input and trigger mapping | 829 |
| 27.4.4 | LPTIM reset and clocks | 830 |
| 27.4.5 | Glitch filter | 831 |
| 27.4.6 | Prescaler | 831 |
| 27.4.7 | Trigger multiplexer | 832 |
| 27.4.8 | Operating mode | 832 |
| 27.4.9 | Timeout function | 834 |
| 27.4.10 | Waveform generation | 834 |
| 27.4.11 | Register update | 836 |
| 27.4.12 | Counter mode | 836 |
| 27.4.13 | Timer enable | 837 |
| 27.4.14 | Timer counter reset | 837 |
| 27.4.15 | Encoder mode | 838 |
| 27.4.16 | Repetition Counter | 839 |
| 27.4.17 | Capture/compare channels | 841 |
| 27.4.18 | Input capture mode | 841 |
| 27.4.19 | PWM mode | 843 |
| 27.4.20 | DMA requests | 845 |

| | | |
|-----------|--|------------|
| 27.4.21 | Debug mode | 846 |
| 27.5 | LPTIM low-power modes | 846 |
| 27.6 | LPTIM interrupts | 846 |
| 27.7 | LPTIM registers | 847 |
| 27.7.1 | LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) ($x = 1$ to 3) | 848 |
| 27.7.2 | LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) ($x = 1$ to 3) | 850 |
| 27.7.3 | LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) ($x = 1$ to 3) | 853 |
| 27.7.4 | LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) ($x = 1$ to 3) | 855 |
| 27.7.5 | LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) ($x = 1$ to 3) | 856 |
| 27.7.6 | LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) ($x = 1$ to 3) | 858 |
| 27.7.7 | LPTIM configuration register (LPTIM_CFGR) | 860 |
| 27.7.8 | LPTIM control register (LPTIM_CR) | 863 |
| 27.7.9 | LPTIM compare register 1 (LPTIM_CCR1) | 864 |
| 27.7.10 | LPTIM autoreload register (LPTIM_ARR) | 865 |
| 27.7.11 | LPTIM counter register (LPTIM_CNT) | 865 |
| 27.7.12 | LPTIM configuration register 2 (LPTIM_CFGR2) | 865 |
| 27.7.13 | LPTIM repetition register (LPTIM_RCR) | 867 |
| 27.7.14 | LPTIM capture/compare mode register 1 (LPTIM_CCMR1) | 867 |
| 27.7.15 | LPTIM capture/compare mode register 2 (LPTIM_CCMR2) | 870 |
| 27.7.16 | LPTIM compare register 2 (LPTIM_CCR2) | 872 |
| 27.7.17 | LPTIM compare register 3 (LPTIM_CCR3) | 873 |
| 27.7.18 | LPTIM compare register 4 (LPTIM_CCR4) | 874 |
| 27.7.19 | LPTIM register map | 874 |
| 28 | Independent watchdog (IWDG) | 877 |
| 28.1 | Introduction | 877 |
| 28.2 | IWDG main features | 877 |
| 28.3 | IWDG implementation | 877 |
| 28.4 | IWDG functional description | 878 |
| 28.4.1 | IWDG block diagram | 878 |
| 28.4.2 | IWDG internal signals | 879 |
| 28.4.3 | Software and hardware watchdog modes | 879 |

| | | |
|-----------|---|------------|
| 28.4.4 | Window option | 880 |
| 28.4.5 | Debug | 883 |
| 28.4.6 | Register access protection | 883 |
| 28.5 | IWDG low power modes | 883 |
| 28.6 | IWDG interrupts | 884 |
| 28.7 | IWDG registers | 886 |
| 28.7.1 | IWDG key register (IWDG_KR) | 887 |
| 28.7.2 | IWDG prescaler register (IWDG_PR) | 887 |
| 28.7.3 | IWDG reload register (IWDG_RLR) | 888 |
| 28.7.4 | IWDG status register (IWDG_SR) | 888 |
| 28.7.5 | IWDG window register (IWDG_WINR) | 890 |
| 28.7.6 | IWDG early wake-up interrupt register (IWDG_EWCR) | 890 |
| 28.7.7 | IWDG register map | 892 |
| 29 | System window watchdog (WWDG) | 893 |
| 29.1 | Introduction | 893 |
| 29.2 | WWDG main features | 893 |
| 29.3 | WWDG implementation | 893 |
| 29.4 | WWDG functional description | 894 |
| 29.4.1 | WWDG block diagram | 894 |
| 29.4.2 | WWDG internal signals | 894 |
| 29.4.3 | Enabling the watchdog | 895 |
| 29.4.4 | Controlling the down-counter | 895 |
| 29.4.5 | How to program the watchdog timeout | 895 |
| 29.4.6 | Debug mode | 896 |
| 29.5 | WWDG interrupts | 897 |
| 29.6 | WWDG registers | 897 |
| 29.6.1 | WWDG control register (WWDG_CR) | 897 |
| 29.6.2 | WWDG configuration register (WWDG_CFR) | 898 |
| 29.6.3 | WWDG status register (WWDG_SR) | 898 |
| 29.6.4 | WWDG register map | 899 |
| 30 | Real-time clock (RTC) | 900 |
| 30.1 | Introduction | 900 |
| 30.2 | RTC main features | 900 |
| 30.3 | RTC functional description | 900 |

| | | |
|---------|---|-----|
| 30.3.1 | RTC block diagram | 900 |
| 30.3.2 | RTC pins and internal signals | 902 |
| 30.3.3 | GPIOs controlled by the RTC and TAMP | 903 |
| 30.3.4 | Clock and prescalers | 905 |
| 30.3.5 | Real-time clock and calendar | 906 |
| 30.3.6 | Calendar ultra-low power mode | 906 |
| 30.3.7 | Programmable alarms | 907 |
| 30.3.8 | Periodic auto-wake-up | 907 |
| 30.3.9 | RTC initialization and configuration | 908 |
| 30.3.10 | Reading the calendar | 911 |
| 30.3.11 | Resetting the RTC | 912 |
| 30.3.12 | RTC synchronization | 912 |
| 30.3.13 | RTC reference clock detection | 912 |
| 30.3.14 | RTC smooth digital calibration | 913 |
| 30.3.15 | Timestamp function | 915 |
| 30.3.16 | Calibration clock output | 916 |
| 30.3.17 | Tamper and alarm output | 917 |
| 30.4 | RTC low-power modes | 917 |
| 30.5 | RTC interrupts | 918 |
| 30.6 | RTC registers | 918 |
| 30.6.1 | RTC time register (RTC_TR) | 919 |
| 30.6.2 | RTC date register (RTC_DR) | 920 |
| 30.6.3 | RTC subsecond register (RTC_SSR) | 921 |
| 30.6.4 | RTC initialization control and status register (RTC_ICSR) | 921 |
| 30.6.5 | RTC prescaler register (RTC_PRER) | 923 |
| 30.6.6 | RTC wake-up timer register (RTC_WUTR) | 924 |
| 30.6.7 | RTC control register (RTC_CR) | 924 |
| 30.6.8 | RTC write protection register (RTC_WPR) | 928 |
| 30.6.9 | RTC calibration register (RTC_CALR) | 928 |
| 30.6.10 | RTC shift control register (RTC_SHIFTR) | 929 |
| 30.6.11 | RTC timestamp time register (RTC_TSTR) | 930 |
| 30.6.12 | RTC timestamp date register (RTC_TSDDR) | 931 |
| 30.6.13 | RTC timestamp subsecond register (RTC_TSSSR) | 931 |
| 30.6.14 | RTC alarm A register (RTC_ALRMAR) | 932 |
| 30.6.15 | RTC alarm A subsecond register (RTC_ALRMASSR) | 933 |
| 30.6.16 | RTC alarm B register (RTC_ALRMBR) | 934 |
| 30.6.17 | RTC alarm B subsecond register (RTC_ALRMBSSR) | 935 |

| | | |
|-----------|--|------------|
| 30.6.18 | RTC status register (RTC_SR) | 936 |
| 30.6.19 | RTC masked interrupt status register (RTC_MISR) | 937 |
| 30.6.20 | RTC status clear register (RTC_SCR) | 938 |
| 30.6.21 | RTC alarm A binary mode register (RTC_ALRABINR) | 939 |
| 30.6.22 | RTC alarm B binary mode register (RTC_ALRBBINR) | 939 |
| 30.6.23 | RTC register map | 940 |
| 31 | Tamper and backup registers (TAMP) | 942 |
| 31.1 | Introduction | 942 |
| 31.2 | TAMP main features | 942 |
| 31.3 | TAMP functional description | 943 |
| 31.3.1 | TAMP block diagram | 943 |
| 31.3.2 | TAMP pins and internal signals | 944 |
| 31.3.3 | GPIOs controlled by the RTC and TAMP | 945 |
| 31.3.4 | TAMP register write protection | 945 |
| 31.3.5 | Tamper detection | 945 |
| 31.3.6 | TAMP backup registers and other device secrets erase | 946 |
| 31.3.7 | Tamper detection configuration and initialization | 947 |
| 31.4 | TAMP low-power modes | 948 |
| 31.5 | TAMP interrupts | 949 |
| 31.6 | TAMP registers | 949 |
| 31.6.1 | TAMP control register 1 (TAMP_CR1) | 949 |
| 31.6.2 | TAMP control register 2 (TAMP_CR2) | 951 |
| 31.6.3 | TAMP control register 3 (TAMP_CR3) | 953 |
| 31.6.4 | TAMP filter control register (TAMP_FLTCR) | 954 |
| 31.6.5 | TAMP interrupt enable register (TAMP_IER) | 955 |
| 31.6.6 | TAMP status register (TAMP_SR) | 956 |
| 31.6.7 | TAMP masked interrupt status register (TAMP_MISR) | 957 |
| 31.6.8 | TAMP status clear register (TAMP_SCR) | 959 |
| 31.6.9 | TAMP backup x register (TAMP_BKPxR) | 960 |
| 31.6.10 | TAMP register map | 961 |
| 32 | Inter-integrated circuit interface (I2C) | 962 |
| 32.1 | Introduction | 962 |
| 32.2 | I2C main features | 962 |
| 32.3 | I2C implementation | 962 |

| | | |
|---------|---|-------------|
| 32.4 | I2C functional description | 963 |
| 32.4.1 | I2C block diagram | 963 |
| 32.4.2 | I2C pins and internal signals | 964 |
| 32.4.3 | I2C clock requirements | 964 |
| 32.4.4 | I2C mode selection | 964 |
| 32.4.5 | I2C initialization | 965 |
| 32.4.6 | I2C reset | 969 |
| 32.4.7 | I2C data transfer | 970 |
| 32.4.8 | I2C slave mode | 972 |
| 32.4.9 | I2C master mode | 981 |
| 32.4.10 | I2C_TIMINGR register configuration examples | 992 |
| 32.4.11 | Wake-up from Stop mode on address match | 994 |
| 32.4.12 | Error conditions | 995 |
| 32.5 | I2C in low-power modes | 996 |
| 32.6 | I2C interrupts | 996 |
| 32.7 | I2C DMA requests | 997 |
| 32.7.1 | Transmission using DMA | 997 |
| 32.7.2 | Reception using DMA | 997 |
| 32.8 | I2C registers | 998 |
| 32.8.1 | I2C control register 1 (I2C_CR1) | 998 |
| 32.8.2 | I2C control register 2 (I2C_CR2) | 1000 |
| 32.8.3 | I2C own address 1 register (I2C_OAR1) | 1002 |
| 32.8.4 | I2C own address 2 register (I2C_OAR2) | 1003 |
| 32.8.5 | I2C timing register (I2C_TIMINGR) | 1004 |
| 32.8.6 | I2C interrupt and status register (I2C_ISR) | 1005 |
| 32.8.7 | I2C interrupt clear register (I2C_ICR) | 1007 |
| 32.8.8 | I2C receive data register (I2C_RXDR) | 1007 |
| 32.8.9 | I2C transmit data register (I2C_TXDR) | 1008 |
| 32.8.10 | I2C register map | 1009 |
| 33 | Universal synchronous/asynchronous receiver transmitter (USART/UART) | 1010 |
| 33.1 | Introduction | 1010 |
| 33.2 | USART main features | 1010 |
| 33.3 | USART extended features | 1011 |
| 33.4 | USART implementation | 1011 |

| | | |
|---------|---|------|
| 33.5 | USART functional description | 1013 |
| 33.5.1 | USART block diagram | 1013 |
| 33.5.2 | USART pins and internal signals | 1013 |
| 33.5.3 | USART clocks | 1015 |
| 33.5.4 | USART character description | 1015 |
| 33.5.5 | USART FIFOs and thresholds | 1018 |
| 33.5.6 | USART transmitter | 1018 |
| 33.5.7 | USART receiver | 1021 |
| 33.5.8 | USART baud rate generation | 1028 |
| 33.5.9 | Tolerance of the USART receiver to clock deviation | 1030 |
| 33.5.10 | USART auto baud rate detection | 1031 |
| 33.5.11 | USART multiprocessor communication | 1033 |
| 33.5.12 | USART Modbus communication | 1035 |
| 33.5.13 | USART parity control | 1036 |
| 33.5.14 | USART LIN (local interconnection network) mode | 1037 |
| 33.5.15 | USART synchronous mode | 1039 |
| 33.5.16 | USART single-wire half-duplex communication | 1043 |
| 33.5.17 | USART receiver timeout | 1043 |
| 33.5.18 | USART smartcard mode | 1044 |
| 33.5.19 | USART IrDA SIR ENDEC block | 1048 |
| 33.5.20 | Continuous communication using USART and DMA | 1051 |
| 33.5.21 | RS232 hardware flow control and RS485 driver enable | 1053 |
| 33.5.22 | USART low-power management | 1056 |
| 33.6 | USART in low-power modes | 1059 |
| 33.7 | USART interrupts | 1059 |
| 33.8 | USART registers | 1062 |
| 33.8.1 | USART control register 1 (USART_CR1) | 1062 |
| 33.8.2 | USART control register 1 [alternate] (USART_CR1) | 1066 |
| 33.8.3 | USART control register 2 (USART_CR2) | 1069 |
| 33.8.4 | USART control register 3 (USART_CR3) | 1073 |
| 33.8.5 | USART control register 3 [alternate] (USART_CR3) | 1078 |
| 33.8.6 | USART baud rate register (USART_BRR) | 1081 |
| 33.8.7 | USART guard time and prescaler register (USART_GTPR) | 1082 |
| 33.8.8 | USART receiver timeout register (USART_RTOR) | 1083 |
| 33.8.9 | USART request register (USART_RQR) | 1084 |
| 33.8.10 | USART interrupt and status register (USART_ISR) | 1085 |
| 33.8.11 | USART interrupt and status register [alternate] (USART_ISR) | 1091 |

| | | |
|-----------|---|-------------|
| 33.8.12 | USART interrupt flag clear register (USART_ICR) | 1096 |
| 33.8.13 | USART receive data register (USART_RDR) | 1098 |
| 33.8.14 | USART transmit data register (USART_TDR) | 1098 |
| 33.8.15 | USART prescaler register (USART_PRESC) | 1099 |
| 33.8.16 | USART register map | 1100 |
| 34 | Low-power universal asynchronous receiver transmitter (LPUART) | 1102 |
| 34.1 | Introduction | 1102 |
| 34.2 | LPUART main features | 1102 |
| 34.3 | LPUART implementation | 1103 |
| 34.4 | LPUART functional description | 1105 |
| 34.4.1 | LPUART block diagram | 1105 |
| 34.4.2 | LPUART pins and internal signals | 1106 |
| 34.4.3 | LPUART clocks | 1107 |
| 34.4.4 | LPUART character description | 1107 |
| 34.4.5 | LPUART FIFOs and thresholds | 1109 |
| 34.4.6 | LPUART transmitter | 1109 |
| 34.4.7 | LPUART receiver | 1113 |
| 34.4.8 | LPUART baud rate generation | 1117 |
| 34.4.9 | Tolerance of the LPUART receiver to clock deviation | 1119 |
| 34.4.10 | LPUART multiprocessor communication | 1120 |
| 34.4.11 | LPUART parity control | 1122 |
| 34.4.12 | LPUART single-wire half-duplex communication | 1123 |
| 34.4.13 | Continuous communication using DMA and LPUART | 1123 |
| 34.4.14 | RS232 hardware flow control and RS485 driver enable | 1126 |
| 34.4.15 | LPUART low-power management | 1128 |
| 34.5 | LPUART in low-power modes | 1131 |
| 34.6 | LPUART interrupts | 1132 |
| 34.7 | LPUART registers | 1133 |
| 34.7.1 | LPUART control register 1 (LPUART_CR1) | 1133 |
| 34.7.2 | LPUART control register 1 [alternate] (LPUART_CR1) | 1136 |
| 34.7.3 | LPUART control register 2 (LPUART_CR2) | 1139 |
| 34.7.4 | LPUART control register 3 (LPUART_CR3) | 1141 |
| 34.7.5 | LPUART control register 3 [alternate] (LPUART_CR3) | 1144 |
| 34.7.6 | LPUART baud rate register (LPUART_BRR) | 1147 |

| | | |
|-----------|---|-------------|
| 34.7.7 | LPUART request register (LPUART_RQR) | 1147 |
| 34.7.8 | LPUART interrupt and status register (LPUART_ISR) | 1148 |
| 34.7.9 | LPUART interrupt and status register [alternate] (LPUART_ISR) | 1152 |
| 34.7.10 | LPUART interrupt flag clear register (LPUART_ICR) | 1155 |
| 34.7.11 | LPUART receive data register (LPUART_RDR) | 1156 |
| 34.7.12 | LPUART transmit data register (LPUART_TDR) | 1157 |
| 34.7.13 | LPUART prescaler register (LPUART_PRESC) | 1157 |
| 34.7.14 | LPUART register map | 1158 |
| 35 | Serial peripheral interface (SPI) | 1160 |
| 35.1 | Introduction | 1160 |
| 35.2 | SPI main features | 1160 |
| 35.3 | SPI implementation | 1161 |
| 35.4 | SPI functional description | 1161 |
| 35.4.1 | General description | 1161 |
| 35.4.2 | Communications between one master and one slave | 1162 |
| 35.4.3 | Standard multislave communication | 1164 |
| 35.4.4 | Multimaster communication | 1165 |
| 35.4.5 | Slave select (NSS) pin management | 1166 |
| 35.4.6 | Communication formats | 1167 |
| 35.4.7 | Configuration of SPI | 1169 |
| 35.4.8 | Procedure for enabling SPI | 1170 |
| 35.4.9 | Data transmission and reception procedures | 1170 |
| 35.4.10 | SPI status flags | 1180 |
| 35.4.11 | SPI error flags | 1181 |
| 35.4.12 | NSS pulse mode | 1182 |
| 35.4.13 | TI mode | 1182 |
| 35.4.14 | CRC calculation | 1183 |
| 35.5 | SPI interrupts | 1185 |
| 35.6 | SPI registers | 1186 |
| 35.6.1 | SPI control register 1 (SPIx_CR1) | 1186 |
| 35.6.2 | SPI control register 2 (SPIx_CR2) | 1188 |
| 35.6.3 | SPI status register (SPIx_SR) | 1190 |
| 35.6.4 | SPI data register (SPIx_DR) | 1191 |
| 35.6.5 | SPI CRC polynomial register (SPIx_CRCPR) | 1192 |
| 35.6.6 | SPI Rx CRC register (SPIx_RXCRCR) | 1192 |

| | | |
|-----------|---|-------------|
| 35.6.7 | SPI Tx CRC register (SPIx_TXCRCR) | 1192 |
| 35.6.8 | SPI register map | 1194 |
| 36 | Universal serial bus full-speed device interface (USB) | 1195 |
| 36.1 | Introduction | 1195 |
| 36.2 | USB main features | 1195 |
| 36.3 | USB implementation | 1195 |
| 36.4 | USB functional description | 1196 |
| 36.4.1 | USB block diagram | 1196 |
| 36.4.2 | USB pins and internal signals | 1197 |
| 36.4.3 | USB reset and clocks | 1197 |
| 36.4.4 | General description and Device mode functionality | 1197 |
| 36.4.5 | Description of USB blocks used in both Device and Host modes | 1198 |
| 36.4.6 | Description of host frame scheduler (HFS) specific to Host mode | 1199 |
| 36.5 | Programming considerations for Device and Host modes | 1200 |
| 36.5.1 | Generic USB Device programming | 1200 |
| 36.5.2 | System and power-on reset | 1201 |
| 36.5.3 | Double-buffered endpoints and usage in Device mode | 1208 |
| 36.5.4 | Double buffered channels: usage in Host mode | 1210 |
| 36.5.5 | Isochronous transfers in Device mode | 1211 |
| 36.5.6 | Isochronous transfers in Host mode | 1212 |
| 36.5.7 | Suspend/resume events | 1213 |
| 36.6 | USB and USB SRAM registers | 1217 |
| 36.6.1 | Common registers | 1217 |
| 36.6.2 | USBSRAM registers | 1236 |
| 36.6.3 | USB register map | 1240 |
| 37 | Debug support (DBG) | 1242 |
| 37.1 | Introduction | 1242 |
| 37.2 | DBG functional description | 1242 |
| 37.2.1 | DBG block diagram | 1242 |
| 37.2.2 | DBG pins and internal signals | 1243 |
| 37.2.3 | ID codes and locking mechanism | 1243 |
| 37.2.4 | DBG reset and clocks | 1243 |
| 37.2.5 | DBG power domains | 1244 |
| 37.2.6 | Debug in low-power modes | 1244 |

| | | |
|-----------|--|-------------|
| 37.2.7 | Security | 1244 |
| 37.3 | Serial-wire debug port (SW-DP) | 1245 |
| 37.3.1 | Serial-wire debug port | 1245 |
| 37.3.2 | Debug port registers | 1246 |
| 37.3.3 | DEBUG port registers | 1247 |
| 37.3.4 | Debug port register map and reset values | 1253 |
| 37.4 | Access ports | 1254 |
| 37.4.1 | Access port registers | 1254 |
| 37.4.2 | Access port register map | 1260 |
| 37.5 | ROM tables | 1262 |
| 37.5.1 | System ROM table registers | 1264 |
| 37.5.2 | System ROM table register map | 1268 |
| 37.5.3 | MCU ROM table registers | 1269 |
| 37.5.4 | MCU ROM table register map | 1273 |
| 37.5.5 | Processor ROM table registers | 1274 |
| 37.5.6 | Processor ROM table register map | 1278 |
| 37.6 | Data watchpoint and trace unit (DWT) | 1279 |
| 37.6.1 | DWT registers | 1279 |
| 37.6.2 | DWT register map | 1285 |
| 37.7 | Breakpoint unit (BPU) | 1287 |
| 37.7.1 | BPU registers | 1287 |
| 37.7.2 | BPU register map | 1292 |
| 37.8 | System control space (SCS) | 1293 |
| 37.8.1 | SCS registers | 1293 |
| 37.8.2 | SCS register map | 1298 |
| 37.9 | Microcontroller debug unit (DBGMCU) | 1299 |
| 37.9.1 | Device ID | 1299 |
| 37.9.2 | Low-power mode emulation | 1299 |
| 37.9.3 | Peripheral clock freeze | 1300 |
| 37.9.4 | DBGMCU registers | 1301 |
| 37.9.5 | DBGMCU register map | 1310 |
| 37.10 | Reference documents | 1312 |
| 38 | Device electronic signature | 1313 |
| 38.1 | Unique device ID register (96 bits) | 1313 |
| 38.2 | Flash memory size data register | 1314 |

| | | |
|-------------------|--|-------------|
| 38.3 | Package data register | 1314 |
| Appendix A | OEM key CRC calculation source code | 1316 |

List of tables

| | | |
|-----------|---|-----|
| Table 1. | Peripherals versus products | 52 |
| Table 2. | STM32U073xx and STM32U083xx memory boundary addresses..... | 57 |
| Table 3. | STM32U031xx memory boundary addresses | 57 |
| Table 4. | STM32U0 series peripheral register boundary addresses | 58 |
| Table 5. | SRAM size | 60 |
| Table 6. | Boot modes..... | 62 |
| Table 7. | Flash memory organization: information block | 65 |
| Table 8. | Flash memory organization: main memory..... | 66 |
| Table 9. | Number of wait states according to flash memory clock (HCLK) frequency..... | 67 |
| Table 10. | Page erase overview | 70 |
| Table 11. | Mass erase overview | 70 |
| Table 12. | Option byte format | 75 |
| Table 13. | Organization of option bytes | 75 |
| Table 14. | Flash memory read protection status | 79 |
| Table 15. | Access status versus protection level and execution modes | 81 |
| Table 17. | HDP extension protection | 84 |
| Table 18. | FLASH interrupt requests | 86 |
| Table 19. | FLASH register map and reset values | 103 |
| Table 20. | PVM features | 112 |
| Table 21. | Low-power mode summary | 116 |
| Table 22. | Functionalities depending on the working mode..... | 117 |
| Table 23. | Low-power run | 120 |
| Table 24. | Sleep..... | 122 |
| Table 25. | Low-power sleep..... | 123 |
| Table 26. | Stop 0 mode | 125 |
| Table 27. | Stop 1 mode | 126 |
| Table 28. | Stop 2 mode | 128 |
| Table 29. | Standby mode..... | 130 |
| Table 30. | Shutdown mode | 132 |
| Table 31. | PWR register map and reset values | 149 |
| Table 32. | Clock source frequency | 163 |
| Table 33. | RCC register map and reset values | 210 |
| Table 34. | CRS features | 214 |
| Table 35. | CRS internal input/output signals | 215 |
| Table 36. | Effect of low-power modes on CRS | 218 |
| Table 37. | Interrupt control bits | 218 |
| Table 38. | CRS register map and reset values | 223 |
| Table 39. | Port bit configuration table | 227 |
| Table 40. | GPIO register map and reset values | 241 |
| Table 41. | SYSCFG register map and reset values..... | 261 |
| Table 42. | DMA implementation | 265 |
| Table 43. | DMA internal input/output signals..... | 266 |
| Table 44. | Programmable data width and endian behavior (when PINC = MINC = 1) | 272 |
| Table 45. | DMA interrupt requests..... | 274 |
| Table 46. | DMA register map and reset values | 282 |
| Table 47. | DMAMUX instantiation | 286 |
| Table 48. | DMAMUX: assignment of multiplexer inputs to resources | 287 |
| Table 49. | DMAMUX: assignment of trigger inputs to resources | 287 |

| | | |
|------------|---|-----|
| Table 50. | DMAMUX: assignment of synchronization inputs to resources | 288 |
| Table 51. | DMAMUX signals | 290 |
| Table 52. | DMAMUX interrupts | 294 |
| Table 53. | DMAMUX register map and reset values | 299 |
| Table 54. | Vector table | 301 |
| Table 55. | EXTI signal overview | 305 |
| Table 56. | EVG pin overview | 305 |
| Table 57. | EXTI event input configurations and register control | 307 |
| Table 58. | EXTI line connections | 309 |
| Table 59. | Masking functionality | 310 |
| Table 60. | EXTI register map sections | 311 |
| Table 61. | EXTI controller register map and reset values | 317 |
| Table 62. | CRC internal input/output signals | 320 |
| Table 63. | CRC register map and reset values | 325 |
| Table 64. | ADC main features | 328 |
| Table 65. | ADC input/output pins | 330 |
| Table 66. | ADC internal input/output signals | 330 |
| Table 67. | External triggers | 330 |
| Table 68. | Latency between trigger and start of conversion | 335 |
| Table 69. | Configuring the trigger polarity | 342 |
| Table 70. | tSAR timings depending on resolution | 343 |
| Table 71. | Analog watchdog comparison | 353 |
| Table 72. | Analog watchdog 1 channel selection | 353 |
| Table 73. | Maximum output results vs N and M. Grayed values indicates truncation | 357 |
| Table 74. | ADC interrupts | 362 |
| Table 75. | ADC register map and reset values | 382 |
| Table 76. | DAC features | 386 |
| Table 77. | DAC input/output pins | 387 |
| Table 78. | DAC internal input/output signals | 387 |
| Table 79. | DAC interconnection | 387 |
| Table 80. | Sample and refresh timings | 394 |
| Table 81. | Channel output modes summary | 395 |
| Table 82. | Effect of low-power modes on DAC | 398 |
| Table 83. | DAC interrupts | 399 |
| Table 84. | DAC register map and reset values | 408 |
| Table 85. | VREF buffer modes | 410 |
| Table 86. | VREFBUF register map and reset values | 412 |
| Table 87. | COMP internal input/output signals | 414 |
| Table 88. | COMP1 noninverting input assignment | 415 |
| Table 89. | COMP1 inverting input assignment | 415 |
| Table 90. | COMP2 noninverting input assignment | 415 |
| Table 91. | COMP2 inverting input assignment | 415 |
| Table 92. | Comparator behavior in the low-power modes | 418 |
| Table 93. | Interrupt control bits | 419 |
| Table 94. | COMP register map and reset values | 424 |
| Table 95. | Operational amplifier possible connections | 426 |
| Table 96. | Operating modes and calibration | 431 |
| Table 97. | Effect of low-power modes on the OPAMP | 432 |
| Table 98. | OPAMP register map and reset values | 435 |
| Table 99. | Example of frame rate calculation | 439 |
| Table 100. | Blink frequency | 447 |
| Table 101. | LCD interrupt requests | 452 |

| | |
|--|-----|
| Table 102. LCD register map and reset values | 459 |
| Table 103. TSC implementation | 463 |
| Table 104. Acquisition sequence summary | 466 |
| Table 105. Spread spectrum deviation versus AHB clock frequency | 468 |
| Table 106. I/O state depending on its mode and IODEF bit value | 469 |
| Table 107. Effect of low-power modes on TSC | 471 |
| Table 108. Interrupt control bits | 474 |
| Table 109. TSC register map and reset values | 482 |
| Table 110. RNG internal input/output signals | 485 |
| Table 111. RNG interrupt requests | 493 |
| Table 112. RNG initialization times | 494 |
| Table 113. RNG configurations | 495 |
| Table 114. Configuration selection | 495 |
| Table 115. RNG register map and reset map | 501 |
| Table 116. AES internal input/output signals | 503 |
| Table 117. CTR mode initialization vector definition | 519 |
| Table 118. GCM last block definition | 521 |
| Table 119. Initialization of AES_IVRx registers in GCM mode | 522 |
| Table 120. Initialization of AES_IVRx registers in CCM mode | 529 |
| Table 121. Key endianness in AES_KEYRx registers (128- or 256-bit key length) | 534 |
| Table 122. AES interrupt requests | 537 |
| Table 123. Processing latency for ECB, CBC and CTR | 537 |
| Table 124. Processing latency for GCM and CCM (in clock cycles) | 538 |
| Table 125. AES register map and reset values | 548 |
| Table 126. Behavior of timer outputs versus BRK/BRK2 inputs | 592 |
| Table 127. Break protection disarming conditions | 594 |
| Table 128. Counting direction versus encoder signals | 600 |
| Table 129. TIM1 internal trigger connection | 617 |
| Table 130. Output control bits for complementary OCx and OCxN channels with break feature | 631 |
| Table 131. TIM1 register map and reset values | 648 |
| Table 132. Counting direction versus encoder signals | 684 |
| Table 133. TIMx internal trigger connection | 702 |
| Table 134. Output control bit for standard OCx channels | 713 |
| Table 135. TIM2/TIM3 register map and reset values | 723 |
| Table 136. TIMx register map and reset values | 738 |
| Table 137. Break protection disarming conditions | 767 |
| Table 138. TIMx Internal trigger connection | 783 |
| Table 139. Output control bits for complementary OCx and OCxN channels with break feature (TIM15) | 793 |
| Table 140. TIM15 register map and reset values | 802 |
| Table 141. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) | 814 |
| Table 142. TIM16 register map and reset values | 823 |
| Table 143. STM32U0 series LPTIM features | 826 |
| Table 144. LPTIM1/2/3 input/output pins | 827 |
| Table 145. LPTIM1/2/3 internal signals | 828 |
| Table 146. LPTIM1/2/3 external trigger connection | 829 |
| Table 147. LPTIM1/2/3 input 1 connection | 829 |
| Table 148. LPTIM1/2/3 input 2 connection | 829 |
| Table 149. LPTIM1/2/3 input capture 1 connection | 829 |
| Table 150. LPTIM1 input capture 2 connection | 830 |
| Table 151. LPTIM2 input capture 2 connection | 830 |

| | |
|---|------|
| Table 152. LPTIM3 input capture 2 connection | 830 |
| Table 153. Prescaler division ratios | 831 |
| Table 154. Encoder counting scenarios | 838 |
| Table 155. Input capture Glitch filter latency (in counter step unit) | 842 |
| Table 156. Effect of low-power modes on the LPTIM | 846 |
| Table 157. Interrupt events | 847 |
| Table 158. LPTIM register map and reset values | 874 |
| Table 159. IWDG features | 877 |
| Table 160. IWDG internal input/output signals | 879 |
| Table 161. Effect of low power modes on IWDG | 884 |
| Table 162. IWDG interrupt request | 886 |
| Table 163. IWDG register map and reset values | 892 |
| Table 164. WWDG features | 893 |
| Table 165. WWDG internal input/output signals | 894 |
| Table 166. WWDG register map and reset values | 899 |
| Table 167. RTC input/output pins | 902 |
| Table 168. RTC internal input/output signals | 902 |
| Table 169. RTC interconnection | 903 |
| Table 170. RTC pin configuration | 903 |
| Table 171. RTC_OUT mapping | 904 |
| Table 172. Effect of low-power modes on RTC | 917 |
| Table 173. RTC pins functionality over modes | 917 |
| Table 174. Interrupt requests | 918 |
| Table 175. RTC register map and reset values | 940 |
| Table 176. TAMP input/output pins | 944 |
| Table 177. TAMP internal input/output signals | 944 |
| Table 178. TAMP interconnection | 945 |
| Table 179. Effect of low-power modes on TAMP | 948 |
| Table 180. TAMP pins functionality over modes | 948 |
| Table 181. Interrupt requests | 949 |
| Table 182. TAMP register map and reset values | 961 |
| Table 183. I2C implementation | 962 |
| Table 184. I2C input/output pins | 964 |
| Table 185. I2C internal input/output signals | 964 |
| Table 186. Comparison of analog and digital filters | 966 |
| Table 187. I ² C-bus specification data setup and hold times | 968 |
| Table 188. I2C configuration | 972 |
| Table 189. I ² C-bus specification clock timings | 983 |
| Table 190. Timing settings for f _{I2CCLK} of 8 MHz | 993 |
| Table 191. Timing settings for f _{I2CCLK} of 16 MHz | 993 |
| Table 192. Timing settings for f _{I2CCLK} of 48 MHz | 994 |
| Table 193. Effect of low-power modes to I2C | 996 |
| Table 194. I2C interrupt requests | 996 |
| Table 195. I2C register map and reset values | 1009 |
| Table 196. Instance implementation on STM32U0 series | 1011 |
| Table 197. USART/LPUART features | 1012 |
| Table 198. USART/UART input/output pins | 1014 |
| Table 199. USART internal input/output signals | 1015 |
| Table 200. Noise detection from sampled data | 1027 |
| Table 201. Tolerance of the USART receiver when BRR [3:0] = 0000 | 1031 |
| Table 202. Tolerance of the USART receiver when BRR[3:0] is different from 0000 | 1031 |
| Table 203. USART frame formats | 1036 |

| | |
|--|------|
| Table 204. Effect of low-power modes on the USART | 1059 |
| Table 205. USART interrupt requests. | 1060 |
| Table 206. USART register map and reset values | 1100 |
| Table 207. Instance implementation on STM32U0 series | 1103 |
| Table 208. USART/LPUART features | 1104 |
| Table 209. LPUART input/output pins | 1106 |
| Table 210. LPUART internal input/output signals. | 1106 |
| Table 211. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz | 1117 |
| Table 212. Error calculation for programmed baud rates at fCK = 100 MHz | 1118 |
| Table 213. Tolerance of the LPUART receiver. | 1119 |
| Table 215. Effect of low-power modes on the LPUART | 1131 |
| Table 216. LPUART interrupt requests. | 1132 |
| Table 217. LPUART register map and reset values | 1158 |
| Table 218. STM32U0 series SPI implementation. | 1161 |
| Table 219. SPI interrupt requests | 1185 |
| Table 220. SPI register map and reset values | 1194 |
| Table 221. STM32U073/83 USB implementation. | 1195 |
| Table 222. USB input/output pins | 1197 |
| Table 223. Double-buffering buffer flag definition. | 1209 |
| Table 224. Bulk double-buffering memory buffers usage (Device mode) | 1209 |
| Table 225. Bulk double-buffering memory buffers usage (Host mode) | 1211 |
| Table 226. Isochronous memory buffers usage | 1212 |
| Table 227. Isochronous memory buffers usage | 1213 |
| Table 228. Resume event detection | 1215 |
| Table 229. Resume event detection for host | 1216 |
| Table 230. Reception status encoding | 1234 |
| Table 231. Endpoint/channel type encoding. | 1234 |
| Table 232. Endpoint/channel kind meaning | 1234 |
| Table 233. Transmission status encoding | 1234 |
| Table 234. Definition of allocated buffer memory | 1238 |
| Table 235. USB register map and reset values | 1240 |
| Table 236. SW debug port pins | 1243 |
| Table 237. Authentication signal states | 1245 |
| Table 238. Life cycle state and debug states | 1245 |
| Table 239. Packet request | 1246 |
| Table 240. ACK response. | 1246 |
| Table 241. Data transfer. | 1246 |
| Table 242. Debug port registers | 1247 |
| Table 243. Debug port register map and reset values | 1253 |
| Table 244. MEM-AP registers. | 1254 |
| Table 245. Access port register map and reset values. | 1260 |
| Table 246. System ROM table | 1262 |
| Table 247. MCU ROM table | 1262 |
| Table 248. Processor ROM table | 1263 |
| Table 249. System ROM table register map and reset values | 1268 |
| Table 250. MCU ROM table register map and reset values | 1273 |
| Table 251. CPU ROM table register map and reset values | 1278 |
| Table 252. DWT register map and reset values | 1285 |
| Table 253. BPU register map and reset values | 1292 |
| Table 254. SCS register map and reset values | 1298 |
| Table 255. Peripheral clock freeze control bits. | 1300 |
| Table 256. Peripheral behavior in debug mode | 1300 |

| | |
|---|------|
| Table 257. DBGMCU register map and reset values | 1310 |
| Table 258. Document revision history | 1318 |

List of figures

| | | |
|------------|---|-----|
| Figure 1. | System architecture | 53 |
| Figure 2. | Memory map | 56 |
| Figure 3. | Changing read protection (RDP) level | 80 |
| Figure 4. | Example of HDP extension protection | 85 |
| Figure 5. | Example of disabling core debug access | 85 |
| Figure 6. | Power supply overview | 106 |
| Figure 7. | Brown-out reset waveform | 111 |
| Figure 8. | PVD thresholds | 112 |
| Figure 9. | Low-power modes possible transitions | 115 |
| Figure 10. | Simplified diagram of the reset circuit | 152 |
| Figure 11. | Clock tree | 157 |
| Figure 12. | HSE/ LSE clock sources | 158 |
| Figure 13. | Frequency measurement with TIM16 in capture mode | 166 |
| Figure 14. | CRS block diagram | 215 |
| Figure 15. | CRS counter behavior | 216 |
| Figure 16. | Basic structure of an I/O port bit | 226 |
| Figure 17. | Basic structure of a 5-Volt tolerant I/O port bit | 226 |
| Figure 18. | Input floating/pull up/pull down configurations | 231 |
| Figure 19. | Output configuration | 232 |
| Figure 20. | Alternate function configuration | 233 |
| Figure 21. | High impedance-analog configuration | 233 |
| Figure 22. | DMA block diagram | 266 |
| Figure 23. | DMAMUX block diagram | 289 |
| Figure 24. | Synchronization mode of the DMAMUX request line multiplexer channel | 292 |
| Figure 25. | Event generation of the DMA request line multiplexer channel | 292 |
| Figure 26. | EXTI block diagram | 305 |
| Figure 27. | Configurable event trigger logic CPU wake-up | 307 |
| Figure 28. | Direct event trigger logic CPU wake-up | 308 |
| Figure 29. | EXTI GPIO mux | 309 |
| Figure 30. | CRC calculation unit block diagram | 320 |
| Figure 31. | ADC block diagram | 329 |
| Figure 32. | DC calibration | 332 |
| Figure 33. | Calibration factor forcing | 333 |
| Figure 34. | Enabling/disabling the ADC | 334 |
| Figure 35. | ADC clock scheme | 334 |
| Figure 36. | ADC connectivity | 336 |
| Figure 37. | Analog-to-digital conversion time | 340 |
| Figure 38. | ADC conversion timings | 341 |
| Figure 39. | Stopping an ongoing conversion | 341 |
| Figure 40. | Single conversions of a sequence, software trigger | 344 |
| Figure 41. | Continuous conversion of a sequence, software trigger | 345 |
| Figure 42. | Single conversions of a sequence, hardware trigger | 345 |
| Figure 43. | Continuous conversions of a sequence, hardware trigger | 346 |
| Figure 44. | Data alignment and resolution (oversampling disabled: OVSE = 0) | 347 |
| Figure 45. | Example of overrun (OVR) | 348 |
| Figure 46. | Wait mode conversion (continuous mode, software trigger) | 350 |
| Figure 47. | Behavior with WAIT = 0, AUTOFF = 1 | 351 |
| Figure 48. | Behavior with WAIT = 1, AUTOFF = 1 | 352 |

| | | |
|------------|--|-----|
| Figure 49. | Analog watchdog guarded area | 353 |
| Figure 50. | ADC_AWDx_OUT signal generation | 354 |
| Figure 51. | ADC_AWDx_OUT signal generation (AWDx flag not cleared by software) | 355 |
| Figure 52. | ADC_AWDx_OUT signal generation (on a single channel) | 355 |
| Figure 53. | Analog watchdog threshold update | 356 |
| Figure 54. | 20-bit to 16-bit result truncation | 356 |
| Figure 55. | Numerical example with 5-bits shift and rounding | 357 |
| Figure 56. | Triggered oversampling mode (TOVS bit = 1) | 359 |
| Figure 57. | Temperature sensor and VREFINT channel block diagram | 360 |
| Figure 58. | DAC block diagram | 386 |
| Figure 59. | Data registers in single DAC channel mode | 389 |
| Figure 60. | Timing diagram for conversion with trigger disabled TEN = 0 | 389 |
| Figure 61. | DAC LFSR register calculation algorithm | 391 |
| Figure 62. | DAC conversion (SW trigger enabled) with LFSR wave generation | 391 |
| Figure 63. | DAC triangle wave generation | 392 |
| Figure 64. | DAC conversion (SW trigger enabled) with triangle wave generation | 392 |
| Figure 65. | DAC Sample and hold mode phase diagram | 395 |
| Figure 66. | Comparator block diagram | 414 |
| Figure 67. | Window mode | 417 |
| Figure 68. | Comparator hysteresis | 417 |
| Figure 69. | Comparator output blanking | 418 |
| Figure 70. | Standalone mode: external gain setting mode | 427 |
| Figure 71. | Follower configuration | 428 |
| Figure 72. | PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used | 429 |
| Figure 73. | PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering | 430 |
| Figure 74. | LCD controller block diagram | 438 |
| Figure 75. | 1/3 bias, 1/4 duty | 440 |
| Figure 76. | Static duty case 1 | 441 |
| Figure 77. | Static duty case 2 | 442 |
| Figure 78. | 1/2 duty, 1/2 bias | 443 |
| Figure 79. | 1/3 duty, 1/3 bias | 444 |
| Figure 80. | 1/4 duty, 1/3 bias | 445 |
| Figure 81. | 1/8 duty, 1/4 bias | 446 |
| Figure 82. | LCD voltage control | 448 |
| Figure 83. | Deadtime | 449 |
| Figure 84. | Flowchart example | 451 |
| Figure 85. | TSC block diagram | 464 |
| Figure 86. | Surface charge transfer analog I/O group structure | 465 |
| Figure 87. | Sampling capacitor voltage variation | 466 |
| Figure 88. | Charge transfer acquisition sequence | 467 |
| Figure 89. | Spread spectrum variation principle | 468 |
| Figure 90. | Surface charge transfer with comparator analog I/O group structure | 472 |
| Figure 91. | Sensor voltage variation for both normal and comparator mode | 473 |
| Figure 92. | RNG block diagram | 485 |
| Figure 93. | NIST SP800-90B entropy source model | 486 |
| Figure 94. | RNG initialization overview | 489 |
| Figure 95. | AES block diagram | 503 |
| Figure 96. | ECB encryption and decryption principle | 505 |
| Figure 97. | CBC encryption and decryption principle | 506 |
| Figure 98. | CTR encryption and decryption principle | 507 |
| Figure 99. | GCM encryption and authentication principle | 508 |

| | |
|---|-----|
| Figure 100. GMAC authentication principle | 508 |
| Figure 101. CCM encryption and authentication principle | 509 |
| Figure 102. Example of suspend mode management | 513 |
| Figure 103. ECB encryption | 514 |
| Figure 104. ECB decryption | 514 |
| Figure 105. CBC encryption | 515 |
| Figure 106. CBC decryption | 515 |
| Figure 107. ECB/CBC encryption (Mode 1) | 516 |
| Figure 108. ECB/CBC decryption (Mode 3) | 517 |
| Figure 109. Message construction in CTR mode | 518 |
| Figure 110. CTR encryption | 519 |
| Figure 111. CTR decryption | 519 |
| Figure 112. Message construction in GCM | 521 |
| Figure 113. GCM authenticated encryption | 522 |
| Figure 114. Message construction in GMAC mode | 526 |
| Figure 115. GMAC authentication mode | 526 |
| Figure 116. Message construction in CCM mode | 527 |
| Figure 117. CCM mode authenticated encryption | 529 |
| Figure 118. 128-bit block construction with respect to data swap | 533 |
| Figure 119. DMA transfer of a 128-bit data block during input phase | 535 |
| Figure 120. DMA transfer of a 128-bit data block during output phase | 536 |
| Figure 121. Advanced-control timer block diagram | 552 |
| Figure 122. Counter timing diagram with prescaler division change from 1 to 2 | 554 |
| Figure 123. Counter timing diagram with prescaler division change from 1 to 4 | 554 |
| Figure 124. Counter timing diagram, internal clock divided by 1 | 556 |
| Figure 125. Counter timing diagram, internal clock divided by 2 | 556 |
| Figure 126. Counter timing diagram, internal clock divided by 4 | 557 |
| Figure 127. Counter timing diagram, internal clock divided by N | 557 |
| Figure 128. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) | 558 |
| Figure 129. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) | 558 |
| Figure 130. Counter timing diagram, internal clock divided by 1 | 560 |
| Figure 131. Counter timing diagram, internal clock divided by 2 | 560 |
| Figure 132. Counter timing diagram, internal clock divided by 4 | 561 |
| Figure 133. Counter timing diagram, internal clock divided by N | 561 |
| Figure 134. Counter timing diagram, update event when repetition counter is not used | 562 |
| Figure 135. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 | 563 |
| Figure 136. Counter timing diagram, internal clock divided by 2 | 564 |
| Figure 137. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 | 564 |
| Figure 138. Counter timing diagram, internal clock divided by N | 565 |
| Figure 139. Counter timing diagram, update event with ARPE=1 (counter underflow) | 565 |
| Figure 140. Counter timing diagram, Update event with ARPE=1 (counter overflow) | 566 |
| Figure 141. Update rate examples depending on mode and TIMx_RCR register settings | 567 |
| Figure 142. External trigger input block | 568 |
| Figure 143. TIM1 ETR input circuitry | 568 |
| Figure 144. Control circuit in normal mode, internal clock divided by 1 | 569 |
| Figure 145. TI2 external clock connection example | 570 |
| Figure 146. Control circuit in external clock mode 1 | 571 |
| Figure 147. External trigger input block | 571 |
| Figure 148. Control circuit in external clock mode 2 | 572 |
| Figure 149. Capture/compare channel (example: channel 1 input stage) | 573 |
| Figure 150. Capture/compare channel 1 main circuit | 573 |
| Figure 151. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3) | 574 |

| | |
|---|-----|
| Figure 152. Output stage of capture/compare channel (channel 4) | 574 |
| Figure 153. Output stage of capture/compare channel (channel 5, idem ch. 6) | 575 |
| Figure 154. PWM input mode timing | 577 |
| Figure 155. Output compare mode, toggle on OC1 | 579 |
| Figure 156. Edge-aligned PWM waveforms (ARR=8) | 580 |
| Figure 157. Center-aligned PWM waveforms (ARR=8) | 581 |
| Figure 158. Generation of 2 phase-shifted PWM signals with 50% duty cycle | 583 |
| Figure 159. Combined PWM mode on channel 1 and 3 | 584 |
| Figure 160. 3-phase combined PWM signals with multiple trigger pulses per period | 585 |
| Figure 161. Complementary output with dead-time insertion | 586 |
| Figure 162. Dead-time waveforms with delay greater than the negative pulse | 586 |
| Figure 163. Dead-time waveforms with delay greater than the positive pulse | 587 |
| Figure 164. Break and Break2 circuitry overview | 589 |
| Figure 165. Various output behavior in response to a break event on BRK (OSSI = 1) | 591 |
| Figure 166. PWM output state following BRK and BRK2 pins assertion (OSSI=1) | 592 |
| Figure 167. PWM output state following BRK assertion (OSSI=0) | 593 |
| Figure 168. Output redirection (BRK2 request not represented) | 594 |
| Figure 169. Clearing TIMx OCxREF | 595 |
| Figure 170. 6-step generation, COM example (OSSR=1) | 596 |
| Figure 171. Example of one pulse mode | 597 |
| Figure 172. Retriggerable one pulse mode | 599 |
| Figure 173. Example of counter operation in encoder interface mode | 600 |
| Figure 174. Example of encoder interface mode with TI1FP1 polarity inverted | 601 |
| Figure 175. Measuring time interval between edges on 3 signals | 602 |
| Figure 176. Example of Hall sensor interface | 604 |
| Figure 177. Control circuit in reset mode | 605 |
| Figure 178. Control circuit in Gated mode | 606 |
| Figure 179. Control circuit in trigger mode | 607 |
| Figure 180. Control circuit in external clock mode 2 + trigger mode | 608 |
| Figure 181. General-purpose timer block diagram | 652 |
| Figure 182. Counter timing diagram with prescaler division change from 1 to 2 | 654 |
| Figure 183. Counter timing diagram with prescaler division change from 1 to 4 | 654 |
| Figure 184. Counter timing diagram, internal clock divided by 1 | 655 |
| Figure 185. Counter timing diagram, internal clock divided by 2 | 656 |
| Figure 186. Counter timing diagram, internal clock divided by 4 | 656 |
| Figure 187. Counter timing diagram, internal clock divided by N | 657 |
| Figure 188. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded) | 657 |
| Figure 189. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded) | 658 |
| Figure 190. Counter timing diagram, internal clock divided by 1 | 659 |
| Figure 191. Counter timing diagram, internal clock divided by 2 | 659 |
| Figure 192. Counter timing diagram, internal clock divided by 4 | 660 |
| Figure 193. Counter timing diagram, internal clock divided by N | 660 |
| Figure 194. Counter timing diagram, Update event | 661 |
| Figure 195. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 | 662 |
| Figure 196. Counter timing diagram, internal clock divided by 2 | 663 |
| Figure 197. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 | 663 |
| Figure 198. Counter timing diagram, internal clock divided by N | 664 |
| Figure 199. Counter timing diagram, Update event with ARPE=1 (counter underflow) | 664 |
| Figure 200. Counter timing diagram, Update event with ARPE=1 (counter overflow) | 665 |
| Figure 201. Control circuit in normal mode, internal clock divided by 1 | 666 |
| Figure 202. TI2 external clock connection example | 666 |
| Figure 203. Control circuit in external clock mode 1 | 667 |

| | |
|---|-----|
| Figure 204. External trigger input block | 668 |
| Figure 205. Control circuit in external clock mode 2 | 669 |
| Figure 206. Capture/Compare channel (example: channel 1 input stage) | 669 |
| Figure 207. Capture/Compare channel 1 main circuit | 670 |
| Figure 208. Output stage of Capture/Compare channel (channel 1) | 670 |
| Figure 209. PWM input mode timing | 672 |
| Figure 210. Output compare mode, toggle on OC1 | 674 |
| Figure 211. Edge-aligned PWM waveforms (ARR=8) | 675 |
| Figure 212. Center-aligned PWM waveforms (ARR=8) | 677 |
| Figure 213. Generation of 2 phase-shifted PWM signals with 50% duty cycle | 678 |
| Figure 214. Combined PWM mode on channels 1 and 3 | 679 |
| Figure 215. Clearing TIMx OCxREF | 680 |
| Figure 216. Example of one-pulse mode | 681 |
| Figure 217. Reriggerable one-pulse mode | 683 |
| Figure 218. Example of counter operation in encoder interface mode | 684 |
| Figure 219. Example of encoder interface mode with TI1FP1 polarity inverted | 685 |
| Figure 220. Control circuit in reset mode | 686 |
| Figure 221. Control circuit in gated mode | 687 |
| Figure 222. Control circuit in trigger mode | 688 |
| Figure 223. Control circuit in external clock mode 2 + trigger mode | 689 |
| Figure 224. Master/Slave timer example | 690 |
| Figure 225. Master/slave connection example with 1 channel only timers | 690 |
| Figure 226. Gating TIM2 with OC1REF of TIM3 | 691 |
| Figure 227. Gating TIM2 with Enable of TIM3 | 692 |
| Figure 228. Triggering TIM2 with update of TIM3 | 693 |
| Figure 229. Triggering TIM2 with Enable of TIM3 | 693 |
| Figure 230. Triggering TIM3 and TIM2 with TIM3 TI1 input | 694 |
| Figure 231. Basic timer block diagram | 726 |
| Figure 232. Counter timing diagram with prescaler division change from 1 to 2 | 728 |
| Figure 233. Counter timing diagram with prescaler division change from 1 to 4 | 728 |
| Figure 234. Counter timing diagram, internal clock divided by 1 | 729 |
| Figure 235. Counter timing diagram, internal clock divided by 2 | 730 |
| Figure 236. Counter timing diagram, internal clock divided by 4 | 730 |
| Figure 237. Counter timing diagram, internal clock divided by N | 731 |
| Figure 238. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) | 731 |
| Figure 239. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) | 732 |
| Figure 240. Control circuit in normal mode, internal clock divided by 1 | 733 |
| Figure 241. TIM15 block diagram | 741 |
| Figure 242. TIM16 block diagram | 742 |
| Figure 243. Counter timing diagram with prescaler division change from 1 to 2 | 744 |
| Figure 244. Counter timing diagram with prescaler division change from 1 to 4 | 744 |
| Figure 245. Counter timing diagram, internal clock divided by 1 | 746 |
| Figure 246. Counter timing diagram, internal clock divided by 2 | 746 |
| Figure 247. Counter timing diagram, internal clock divided by 4 | 747 |
| Figure 248. Counter timing diagram, internal clock divided by N | 747 |
| Figure 249. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) | 748 |
| Figure 250. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) | 748 |
| Figure 251. Update rate examples depending on mode and TIMx_RCR register settings | 750 |

| | |
|---|-----|
| Figure 252. Control circuit in normal mode, internal clock divided by 1 | 751 |
| Figure 253. TI2 external clock connection example | 751 |
| Figure 254. Control circuit in external clock mode 1 | 752 |
| Figure 255. Capture/compare channel (example: channel 1 input stage) | 753 |
| Figure 256. Capture/compare channel 1 main circuit | 753 |
| Figure 257. Output stage of capture/compare channel (channel 1) | 754 |
| Figure 258. Output stage of capture/compare channel (channel 2 for TIM15) | 754 |
| Figure 259. PWM input mode timing | 756 |
| Figure 260. Output compare mode, toggle on OC1 | 758 |
| Figure 261. Edge-aligned PWM waveforms (ARR=8) | 759 |
| Figure 262. Combined PWM mode on channel 1 and 2 | 760 |
| Figure 263. Complementary output with dead-time insertion | 761 |
| Figure 264. Dead-time waveforms with delay greater than the negative pulse | 761 |
| Figure 265. Dead-time waveforms with delay greater than the positive pulse | 762 |
| Figure 266. Break circuitry overview | 764 |
| Figure 267. Output behavior in response to a break | 766 |
| Figure 268. Output redirection | 768 |
| Figure 269. 6-step generation, COM example (OSSR=1) | 769 |
| Figure 270. Example of one pulse mode | 770 |
| Figure 271. Retriggerable one pulse mode | 772 |
| Figure 272. Measuring time interval between edges on 2 signals | 773 |
| Figure 273. Control circuit in reset mode | 774 |
| Figure 274. Control circuit in gated mode | 775 |
| Figure 275. Control circuit in trigger mode | 776 |
| Figure 276. LPTIM1/2/3 timer block diagram ⁽¹⁾ | 827 |
| Figure 277. Glitch filter timing diagram | 831 |
| Figure 278. LPTIM output waveform, single counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) | 833 |
| Figure 279. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set) | 833 |
| Figure 280. LPTIM output waveform, Continuous counting mode configuration | 834 |
| Figure 281. Waveform generation | 835 |
| Figure 282. Encoder mode counting sequence | 839 |
| Figure 283. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1) | 840 |
| Figure 284. Capture/compare input stage (channel 1) | 841 |
| Figure 285. Capture/compare output stage (channel 1) | 841 |
| Figure 286. Edge-aligned PWM mode (PRELOAD = 1) | 843 |
| Figure 287. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) | 844 |
| Figure 288. PWM mode with immediate update versus preloaded update | 845 |
| Figure 289. Independent watchdog block diagram | 878 |
| Figure 290. Reset timing due to timeout | 880 |
| Figure 291. Reset timing due to refresh in the not allowed area | 881 |
| Figure 292. Example of window comparator update | 883 |
| Figure 293. Independent watchdog interrupt timing diagram | 885 |
| Figure 294. Example of early wake-up comparator update | 886 |
| Figure 295. Watchdog block diagram | 894 |
| Figure 296. Window watchdog timing diagram | 896 |
| Figure 297. RTC block diagram | 901 |
| Figure 298. TAMP block diagram | 943 |
| Figure 299. Block diagram | 963 |
| Figure 300. I ² C-bus protocol | 965 |

| | |
|---|------|
| Figure 301. Setup and hold timings | 967 |
| Figure 302. I2C initialization flow | 969 |
| Figure 303. Data reception | 970 |
| Figure 304. Data transmission | 971 |
| Figure 305. Slave initialization flow | 974 |
| Figure 306. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0 | 976 |
| Figure 307. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1 | 977 |
| Figure 308. Transfer bus diagrams for I2C slave transmitter (mandatory events only) | 978 |
| Figure 309. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 0 | 979 |
| Figure 310. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 1 | 980 |
| Figure 311. Transfer bus diagrams for I2C slave receiver (mandatory events only) | 980 |
| Figure 312. Master clock generation | 982 |
| Figure 313. Master initialization flow | 984 |
| Figure 314. 10-bit address read access with HEAD10R = 0 | 984 |
| Figure 315. 10-bit address read access with HEAD10R = 1 | 985 |
| Figure 316. Transfer sequence flow for I2C master transmitter, $N \leq 255$ bytes | 986 |
| Figure 317. Transfer sequence flow for I2C master transmitter, $N > 255$ bytes | 987 |
| Figure 318. Transfer bus diagrams for I2C master transmitter (mandatory events only) | 988 |
| Figure 319. Transfer sequence flow for I2C master receiver, $N \leq 255$ bytes | 990 |
| Figure 320. Transfer sequence flow for I2C master receiver, $N > 255$ bytes | 991 |
| Figure 321. Transfer bus diagrams for I2C master receiver (mandatory events only) | 992 |
| Figure 322. USART block diagram | 1013 |
| Figure 323. Word length programming | 1017 |
| Figure 324. Configurable stop bits | 1019 |
| Figure 325. TC/TXE behavior when transmitting | 1021 |
| Figure 326. Start bit detection when oversampling by 16 or 8 | 1022 |
| Figure 327. usart_ker_ck clock divider block diagram | 1025 |
| Figure 328. Data sampling when oversampling by 16 | 1026 |
| Figure 329. Data sampling when oversampling by 8 | 1027 |
| Figure 330. Mute mode using Idle line detection | 1034 |
| Figure 331. Mute mode using address mark detection | 1035 |
| Figure 332. Break detection in LIN mode (11-bit break length - LBDL bit is set) | 1038 |
| Figure 333. Break detection in LIN mode vs. Framing error detection | 1039 |
| Figure 334. USART example of synchronous master transmission | 1040 |
| Figure 335. USART data clock timing diagram in synchronous master mode (M bits =00) | 1040 |
| Figure 336. USART data clock timing diagram in synchronous master mode (M bits = 01) | 1041 |
| Figure 337. USART data clock timing diagram in synchronous slave mode (M bits =00) | 1042 |
| Figure 338. ISO 7816-3 asynchronous protocol | 1044 |
| Figure 339. Parity error detection using the 1.5 stop bits | 1046 |
| Figure 340. IrDA SIR ENDEC block diagram | 1050 |
| Figure 341. IrDA data modulation (3/16) - normal mode | 1050 |
| Figure 342. Transmission using DMA | 1052 |
| Figure 343. Reception using DMA | 1053 |
| Figure 344. Hardware flow control between 2 USARTs | 1053 |
| Figure 345. RS232 RTS flow control | 1054 |
| Figure 346. RS232 CTS flow control | 1055 |

| | |
|---|------|
| Figure 347. Wake-up event verified (wake-up event = address match, FIFO disabled) | 1058 |
| Figure 348. Wake-up event not verified (wake-up event = address match, FIFO disabled) | 1058 |
| Figure 349. LPUART block diagram | 1105 |
| Figure 350. LPUART word length programming | 1108 |
| Figure 351. Configurable stop bits | 1110 |
| Figure 352. TC/TXE behavior when transmitting | 1112 |
| Figure 353. Ipuart_ker_ck clock divider block diagram | 1116 |
| Figure 354. Mute mode using Idle line detection | 1121 |
| Figure 355. Mute mode using address mark detection | 1122 |
| Figure 356. Transmission using DMA | 1124 |
| Figure 357. Reception using DMA | 1125 |
| Figure 358. Hardware flow control between two LPUARTs. | 1126 |
| Figure 359. RS232 RTS flow control | 1126 |
| Figure 360. RS232 CTS flow control | 1127 |
| Figure 361. Wake-up event verified (wake-up event = address match, FIFO disabled) | 1130 |
| Figure 362. Wake-up event not verified (wake-up event = address match, FIFO disabled) | 1130 |
| Figure 363. SPI block diagram | 1161 |
| Figure 364. Full-duplex single master/ single slave application | 1162 |
| Figure 365. Half-duplex single master/ single slave application | 1163 |
| Figure 366. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) | 1164 |
| Figure 367. Master and three independent slaves | 1165 |
| Figure 368. Multimaster application | 1166 |
| Figure 369. Hardware/software slave select management | 1167 |
| Figure 370. Data clock timing diagram | 1168 |
| Figure 371. Data alignment when data length is not equal to 8-bit or 16-bit | 1169 |
| Figure 372. Packing data in FIFO for transmission and reception | 1173 |
| Figure 373. Master full-duplex communication | 1176 |
| Figure 374. Slave full-duplex communication | 1177 |
| Figure 375. Master full-duplex communication with CRC | 1178 |
| Figure 376. Master full-duplex communication in packed mode | 1179 |
| Figure 377. NSSP pulse generation in Motorola SPI master mode | 1182 |
| Figure 378. TI mode transfer | 1183 |
| Figure 379. USB peripheral block diagram | 1196 |
| Figure 380. Packet buffer areas with examples of buffer description table locations | 1203 |
| Figure 381. Block diagram of debug support infrastructure | 1242 |
| Figure 382. CoreSight topology | 1263 |

1 Documentation conventions

1.1 General information

The STM32U0 series devices have an Arm^{®(a)} Cortex[®]-M0+ 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.
- **SWD-DP (SWD DEBUG PORT)**: SWD-DP provides a 2-pin (clock and data) interface based on the Serial Wire Debug (SWD) protocol. Please refer to the Cortex®-M0+ technical reference manual.
- **IAP (in-application programming)**: IAP is the ability to re-program the flash memory of a microcontroller while the user program is running.
- **ICP (in-circuit programming)**: ICP is the ability to program the flash memory of a microcontroller using the SWD protocol or the bootloader while the device is mounted on the user application board.
- **Option bytes**: product configuration bits stored in the flash memory.
- **OBL**: option byte loader.
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.

1.4 Availability of peripherals

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

The following table shows per-product availability of peripherals that are not common to all STM32U0 series products.

Table 1. Peripherals versus products

| Feature | STM32U031xx | STM32U073xx | STM32U083xx |
|-----------|-------------|-------------|-------------|
| AES | No | No | Yes |
| USB | No | Yes | Yes |
| HSI48/CRS | No | Yes | Yes |
| LCD | No | Yes | Yes |
| LPTIM3 | No | Yes | Yes |
| LPUART3 | No | Yes | Yes |
| COMP2 | No | Yes | Yes |
| SPI3 | No | Yes | Yes |
| I2C4 | No | Yes | Yes |

2 Memory and bus architecture

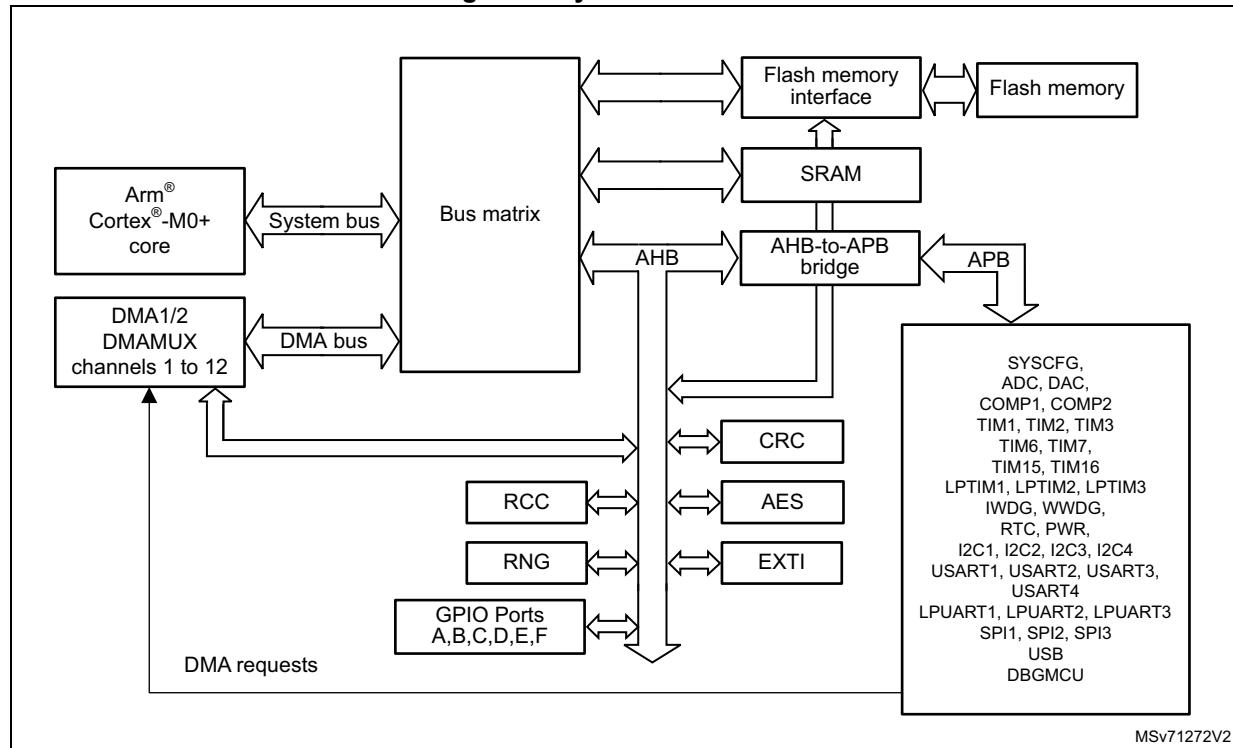
2.1 System architecture

The main system consists of:

- Two masters:
 - Cortex®-M0+ core
 - General-purpose DMA
- Three slaves:
 - Internal SRAM
 - Internal flash memory
 - AHB with AHB-to-APB bridge that connects all the APB peripherals

These are interconnected using a multilayer AHB bus architecture as shown in [Figure 1](#).

Figure 1. System architecture



System bus (S-bus)

This bus connects the system bus of the Cortex®-M0+ core (peripheral bus) to a bus matrix that manages the arbitration between the core and the DMA.

DMA bus

This bus connects the AHB master interface of the DMA to the bus matrix that manages the access of CPU and DMA to SRAM, flash memory and AHB/APB peripherals.

Bus matrix

The bus matrix manages the access arbitration between the core system bus and the DMA master bus. The arbitration uses a Round Robin algorithm. The bus matrix is composed of masters (CPU, DMA) and slaves (flash memory interface, SRAM and AHB-to-APB bridge).

AHB peripherals are connected on system bus through the bus matrix to allow DMA access.

AHB-to-APB bridge (APB)

The AHB-to-APB bridge provides full synchronous connections between the AHB and the APB bus.

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 SRAM and flash memory). Before using a peripheral its clock in the RCC_AHBENR, RCC_APBENRx or RCC_IOPENR register must first be enabled.

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

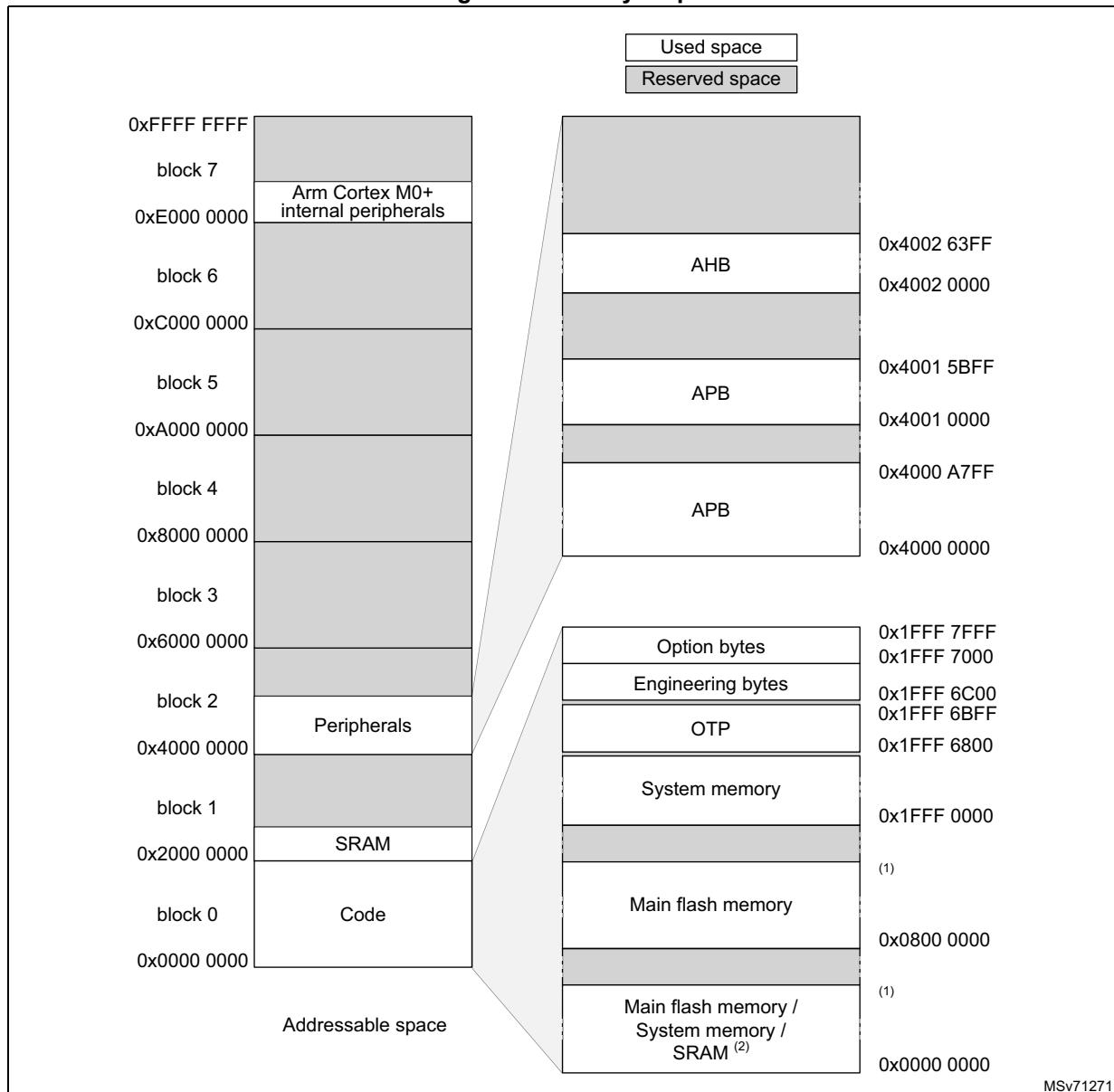
2.2.1 Introduction

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.

2.2.2 Memory map and register boundary addresses

Figure 2. Memory map



1. STM32U073xx and STM32U083xx: 0x0803 FFFF; STM32U031xx: 0x0800 FFFF.

2. Depends on boot configuration.

All the memory map areas that are not allocated to on-chip memories and peripherals are considered as reserved. For the detailed mapping of available memory and register areas, refer to the following tables.

Table 2. STM32U073xx and STM32U083xx memory boundary addresses

| Type | Boundary address | Size | Memory Area | Register description |
|------|--|--------------|---|--|
| SRAM | 0x2002 A000 - 0x3FFF FFFF | ~512 Mbytes | Reserved | - |
| | 0x2000 0000 - 0x2002 9FFF | 40 Kbytes | SRAM | Section 2.3 on page 60 |
| Code | 0x1FFF 8000 - 0x1FFF FFFF | ~34 Kbytes | Reserved | - |
| | 0x1FFF 7000 - 0x1FFF 7FFF | 4 Kbytes | Option bytes | Section 3.4 on page 75 |
| | 0x1FFF 6C00 - 0x1FFF 6FFF | 1 Kbyte | Engineering bytes | - |
| | 0x1FFF 6800 - 0x1FFF 6BFF | 1 Kbyte | OTP | - |
| | 0x1FFF 0000 - 0x1FFF 67FF | 26 Kbytes | System memory | - |
| | 0x0808 0000 - 0x1FFF D7FF | ~384 Mbytes | Reserved | - |
| | 0x0800 0000 - 0x0803 FFFF | 256 Kbytes | Main flash memory | Section 3.3.1 on page 65 |
| | 0x0008 0000 - 0x07FF FFFF | ~7.75 Mbytes | Reserved | - |
| | 0x0000 0000 - 0x0003 FFFF ⁽¹⁾ | 256 Kbytes | Main flash memory, system memory or SRAM depending on BOOT configuration | - |

1. The memory boundary depends on the memory size of the ordered devices.

Table 3. STM32U031xx memory boundary addresses

| Type | Boundary address | Size | Memory Area | Register description |
|------|--|-------------|---|--|
| SRAM | 0x2000 A000 - 0x3FFF FFFF | ~512 Mbytes | Reserved | - |
| | 0x2000 0000 - 0x2000 2FFF | 12 Kbytes | SRAM | Section 2.3 on page 60 |
| Code | 0x1FFF 8000 - 0x1FFF FFFF | ~34 Kbytes | Reserved | - |
| | 0x1FFF 7000 - 0x1FFF 7FFF | 4 Kbytes | Option bytes | Section 3.4 on page 75 |
| | 0x1FFF 6C00 - 0x1FFF 6FFF | 1 Kbyte | Engineering bytes | - |
| | 0x1FFF 6800 - 0x1FFF 6BFF | 1 Kbyte | OTP | - |
| | 0x1FFF 0000 - 0x1FFF 67FF | 26 Kbytes | System memory | - |
| | 0x0802 0000 - 0x1FFF D7FF | ~384 Mbytes | Reserved | - |
| | 0x0800 0000 - 0x0801 FFFF | 64 Kbytes | Main flash memory | Section 3.3.1 on page 65 |
| | 0x0002 0000 - 0x07FF FFFF | ~8 Mbytes | Reserved | - |
| | 0x0000 0000 - 0x0001 FFFF ⁽¹⁾ | 64 Kbytes | Main flash memory, system memory or SRAM depending on BOOT configuration | - |

1. The memory boundary depends on the memory size of the ordered devices.

The following table gives the boundary addresses of the peripherals.

Table 4. STM32U0 series peripheral register boundary addresses

| Bus | Boundary address | Size | Peripheral | Peripheral register map |
|-----|---------------------------|-------------|----------------------------------|---|
| - | 0xE000 0000 - 0xE00F FFFF | 1Mbytes | Cortex®-M0+ internal peripherals | - |
| AHB | 0x5000 1800 - 0x5FFF FFFF | ~256 Mbytes | Reserved | - |
| | 0x5000 1400 - 0x5000 17FF | 1 Kbyte | GPIOF | Section 7.4.12 on page 240 |
| | 0x5000 1000 - 0x5000 13FF | 1 Kbyte | GPIOE | Section 7.4.12 on page 240 |
| | 0x5000 0C00 - 0x5000 0FFF | 1 Kbyte | GPIOD | Section 7.4.12 on page 240 |
| | 0x5000 0800 - 0x5000 0BFF | 1 Kbyte | GPIOC | Section 7.4.12 on page 240 |
| | 0x5000 0400 - 0x5000 07FF | 1 Kbyte | GPIOB | Section 7.4.12 on page 240 |
| | 0x5000 0000 - 0x5000 03FF | 1 Kbyte | GPIOA | Section 7.4.12 on page 240 |
| | 0x4002 6400 - 0x4FFF FFFF | ~256 Mbytes | Reserved | - |
| | 0x4002 6000 - 0x4002 63FF | 1 Kbyte | AES | Section 22.7.18 on page 548 |
| | 0x4002 5400 - 0x4002 5FFF | 3 Kbytes | Reserved | - |
| | 0x4002 5000 - 0x4002 53FF | 1 Kbyte | RNG | Section 21.7.6 on page 501 |
| | 0x4002 4400 - 0x4002 4FFF | 3 Kbytes | Reserved | - |
| | 0x4002 4000 - 0x4002 43FF | 1 Kbyte | TSC | Section 20.7.11 on page 482 |
| | 0x4002 3400 - 0x4002 3FFF | 3 Kbytes | Reserved | - |
| | 0x4002 3000 - 0x4002 33FF | 1 Kbyte | CRC | Section 13.4.6 on page 325 |
| | 0x4002 2400 - 0x4002 2FFF | 3 Kbyte | Reserved | - |
| | 0x4002 2000 - 0x4002 23FF | 1 Kbyte | FLASH | Section 3.7.22 on page 103 |
| | 0x4002 1C00 - 0x4002 1FFF | 3 Kbytes | Reserved | - |
| | 0x4002 1800 - 0x4002 1BFF | 1 Kbyte | EXTI | Section 12.5.11 on page 317 |
| | 0x4002 1400 - 0x4002 17FF | 1 Kbyte | Reserved | - |
| | 0x4002 1000 - 0x4002 13FF | 1 Kbyte | RCC | Section 5.4.25 on page 210 |
| | 0x4002 0C00 - 0x4002 0FFF | 1 Kbyte | Reserved | - |
| | 0x4002 0800 - 0x4002 0BFF | 1 Kbyte | DMAMUX | Section 10.6.7 on page 299 |
| | 0x4002 0400 - 0x4002 07FF | 1 Kbyte | DMA2 | Section 9.6.7 on page 282 |
| | 0x4002 0000 - 0x4002 03FF | 1 Kbyte | DMA1 | Section 9.6.7 on page 282 |

Table 4. STM32U0 series peripheral register boundary addresses (continued)

| Bus | Boundary address | Size | Peripheral | Peripheral register map |
|-----|---------------------------|-----------|-------------------------------|--|
| APB | 0x4001 5C00 - 0x4001 FFFF | 32 Kbytes | Reserved | - |
| | 0x4001 5800 - 0x4001 5BFF | 1 Kbyte | DBG | Section 37.9.5 on page 1310 |
| | 0x4001 4800 - 0x4001 57FF | 4 Kbytes | Reserved | - |
| | 0x4001 4400 - 0x4001 47FF | 1 Kbyte | TIM16 | Section 26.6.19 on page 823 |
| | 0x4001 4000 - 0x4001 43FF | 1 Kbyte | TIM15 | Section 26.6.19 on page 823 |
| | 0x4001 3C00 - 0x4001 3FFF | 1 Kbyte | Reserved | - |
| | 0x4001 3800 - 0x4001 3BFF | 1 Kbyte | USART1 | Section 33.8.16 on page 1100 |
| | 0x4001 3400 - 0x4001 37FF | 1 Kbyte | Reserved | - |
| | 0x4001 3000 - 0x4001 33FF | 1 Kbyte | SPI1 | Section 35.6.8 on page 1194 |
| | 0x4001 2C00 - 0x4001 2FFF | 1 Kbyte | TIM1 | Section 23.4 on page 611 |
| | 0x4001 2800 - 0x4001 2BFF | 1 Kbyte | Reserved | - |
| | 0x4001 2400 - 0x4001 27FF | 1 Kbyte | ADC | Section 14.14 on page 382 |
| | 0x4001 0400 - 0x4001 23FF | 8 Kbytes | Reserved | - |
| | 0x4001 0200 - 0x4001 03FF | 1 Kbyte | COMP | Section 17.6 on page 420 |
| | 0x4001 0080 - 0x4001 01FF | | SYSCFG(ITLINE) ⁽¹⁾ | Section 8.1.38 on page 261 |
| | 0x4001 0030 - 0x4001 007F | | VREFBUF | Section 16.3.3 on page 412 |
| | 0x4001 0000 - 0x4001 002F | | SYSCFG | Section 8.1.38 on page 261 |
| | 0x4000 B400- 0x4000 FFFF | 19 Kbytes | Reserved | - |
| | 0x4000 B000 - 0x4000 B3FF | 1 Kbyte | TAMP (+ BKP registers) | Section 31.6.10 on page 961 |
| | 0x4000 9C00 - 0x4000 AFFF | 5 Kbytes | Reserved | - |
| | 0x4000 9800 - 0x4000 9BFF | 1 Kbyte | USB RAM1 | - |
| | 0x4000 9400 - 0x4000 97FF | 1 Kbyte | LPTIM2 | Section 27.7.19 on page 874 |
| | 0x4000 9000 - 0x4000 93FF | 1 Kbyte | LPTIM3 | Section 27.7.19 on page 874 |
| | 0x4000 8C00 - 0x4000 8FFF | 1 Kbyte | LPUART3 | Section 34.7.14 on page 1158 |
| | 0x4000 8800 - 0x4000 8BFF | 1 Kbyte | I2C3 | Section 32.8.10 on page 1009 |
| | 0x4000 8400 - 0x4000 87FF | 1 Kbyte | LPUART2 | Section 34.7.14 on page 1158 |
| | 0x4000 8000 - 0x4000 83FF | 1 Kbyte | LPUART1 | Section 34.7.14 on page 1158 |
| | 0x4000 7C00 - 0x4000 7FFF | 1 Kbyte | LPTIM1 | Section 27.7.19 on page 874 |
| | 0x4000 7800 - 0x4000 7BFF | 1 Kbyte | OPAMP | Section 18.5.4 on page 435 |
| | 0x4000 7400 - 0x4000 77FF | 1 Kbyte | DAC | Section 15.7.13 on page 408 |
| | 0x4000 7000 - 0x4000 73FF | 1 Kbyte | PWR | Section 4.4.20 on page 149 |
| | 0x4000 6C00 - 0x4000 6FFF | 1 Kbyte | CRS | Section 6.7.5 on page 223 |
| | 0x4000 6000 - 0x4000 6BFF | 3 Kbyte | Reserved | - |
| | 0x4000 5C00 - 0x4000 5FFF | 1 Kbyte | USB | Section 36.6.3 on page 1240 |

Table 4. STM32U0 series peripheral register boundary addresses (continued)

| Bus | Boundary address | Size | Peripheral | Peripheral register map |
|-----|---------------------------|----------|------------|--|
| APB | 0x4000 5800 - 0x4000 5BFF | 1 Kbyte | I2C2 | Section 32.8.10 on page 1009 |
| | 0x4000 5400 - 0x4000 57FF | 1 Kbyte | I2C1 | Section 32.8.10 on page 1009 |
| | 0x4000 5000 - 0x4000 53FF | 1 Kbyte | Reserved | - |
| | 0x4000 4C00 - 0x4000 4FFF | 1 Kbyte | USART4 | Section 33.8.16 on page 1100 |
| | 0x4000 4800 - 0x4000 4BFF | 1 Kbyte | USART3 | Section 33.8.16 on page 1100 |
| | 0x4000 4400 - 0x4000 47FF | 1 Kbyte | USART2 | Section 33.8.16 on page 1100 |
| | 0x4000 4000 - 0x4000 43FF | 1 Kbyte | Reserved | - |
| | 0x4000 3C00 - 0x4000 3FFF | 1 Kbyte | SPI3 | Section 35.6.8 on page 1194 |
| | 0x4000 3800 - 0x4000 3BFF | 1 Kbyte | SPI2 | Section 35.6.8 on page 1194 |
| | 0x4000 3400 - 0x4000 37FF | 1 Kbyte | Reserved | - |
| | 0x4000 3000 - 0x4000 33FF | 1 Kbyte | IWDG | Section 28.7.7 on page 892 |
| | 0x4000 2C00 - 0x4000 2FFF | 1 Kbyte | WWDG | Section 29.6.4 on page 899 |
| | 0x4000 2800 - 0x4000 2BFF | 1 Kbyte | RTC | Section 30.6.23 on page 940 |
| | 0x4000 2400 - 0x4000 27FF | 1 Kbyte | LCD | Section 19.6.8 on page 459 |
| | 0x4000 1800 - 0x4000 23FF | 3 Kbytes | Reserved | - |
| | 0x4000 1400 - 0x4000 17FF | 1 Kbyte | TIM7 | Section 25.4.9 on page 738 |
| | 0x4000 1000 - 0x4000 13FF | 1 Kbyte | TIM6 | Section 25.4.9 on page 738 |
| | 0x4000 0800 - 0x4000 0FFF | 2 Kbyte | Reserved | - |
| | 0x4000 0400 - 0x4000 07FF | 1 Kbyte | TIM3 | Section 24.4.28 on page 723 |
| | 0x4000 0000 - 0x4000 03FF | 1 Kbyte | TIM2 | Section 24.4.28 on page 723 |

1. SYSCFG (ITLINE) registers use 0x4001 0000 as reference peripheral base address.

2.3 Embedded SRAM

The following table summarizes the SRAM resources on the devices. Enabling or disabling the parity check does not impact the size of the available SRAM.

Table 5. SRAM size

| Device | SRAM1 (Kbyte) | SRAM2 (Kbyte) |
|-------------|---------------|---------------|
| STM32U083xx | 32 | 8 |
| STM32U073xx | 32 | 8 |
| STM32U031xx | 8 | 4 |

The SRAM can be accessed by bytes, half-words (16 bits) or full words (32 bits), at maximum system clock frequency without wait state and thus by both CPU and DMA.

Parity check

The user can enable the parity check using the option bit RAM_PARITY_CHECK in the user option byte (refer to [Section 3.4: FLASH option bytes](#)).

The data bus width is 36 bits because 4 bits are available for parity check (1 bit per byte) in order to increase memory robustness, as required for instance by Class B or SIL norms.

The parity bits are computed and stored when writing into the SRAM. Then, they are automatically checked when reading. If one bit fails, an NMI is generated. In addition, to get the SRAM parity error at the same cycle time that it is occurring, a bus error is generated (triggering a HardFault exception) together with the NMI. This avoids the corrupted data to be used by the application, but with the side effect of having both NMI and HardFault interrupts generated. The same error can also be linked to the BRK_IN Break input of TIM1/15, with the SRAM_PARITY_LOCK control bit in the [SYSCFG configuration register 2 \(SYSCFG_CFGR2\)](#). The SRAM Parity Error flag (SRAM_PEF) is available in the [SYSCFG configuration register 2 \(SYSCFG_CFGR2\)](#).

Note: When enabling the SRAM parity check, it is advised to initialize by software the whole SRAM at the beginning of the code, to avoid getting parity errors when reading non-initialized locations.

2.4 Flash memory overview

The flash memory is composed of two distinct physical areas:

- The main flash memory block. It contains the application program and user data if necessary.
 - The information block. It is composed of three parts:
 - Option bytes for hardware and memory protection user configuration.
 - System memory which contains the proprietary bootloader code.
 - OTP (one-time programmable) area
- Refer to [Section 3: Embedded flash memory \(FLASH\)](#) for more details.

The flash interface implements instruction access and data access based on the AHB protocol. It implements the prefetch buffer that speeds up CPU code execution. It also implements the logic necessary to carry out the flash memory operations (Program/Erase) controlled through the flash registers.

2.5 Boot configuration

In the STM32U0 series, three different boot modes can be selected through the BOOT0 pin, BOOT_LOCK bit in FLASH_SECR register, and boot configuration bits nBOOT1, BOOT_SEL and nBOOT0 in the User option byte, as shown in the following table.

Table 6. Boot modes

| Boot mode configuration | | | | | Selected boot area |
|-------------------------|------------|-----------|---------------|------------|--------------------------|
| BOOT_LOCK bit | nBOOT1 bit | BOOT0 pin | nBOOT_SEL bit | nBOOT0 bit | |
| 0 | x | 0 | 0 | x | Main flash memory |
| 0 | 1 | 1 | 0 | x | System memory |
| 0 | 0 | 1 | 0 | x | Embedded SRAM |
| 0 | x | x | 1 | 1 | Main flash memory |
| 0 | 1 | x | 1 | 0 | System memory |
| 0 | 0 | x | 1 | 0 | Embedded SRAM |
| 1 | x | x | x | x | Main flash memory forced |

The boot mode configuration is latched on the 4th rising edge of SYCLK after a reset. It is up to the user to set boot mode configuration related to the required boot mode.

The boot mode configuration is also re-sampled when exiting from Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory at 0x0000 0004.

Depending on the selected boot mode, main flash memory, system memory or SRAM is accessible as follows:

- Boot from main flash memory: the main flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x0800 0000). In other words, the flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space 0x1FFF0000.
- Boot from the embedded SRAM: the SRAM is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x2000 0000).

Caution: BOOT0 pin shares the same GPIO with serial wire clock (SWCLK) that is used by the debugger to connect with the device, based on the fact that these functionalities can be considered almost completely disjoint. Nevertheless, to ensure system robustness, the STM32U0 series devices provide a hardware mechanism to force BOOT0 low (boot from user flash memory) if a debugger access is detected (and BOOT0 information is taken from the pin), in order to use SWCLK clock for debugger serial communications and at the same time have a safe boot configuration for the device itself. This configuration is kept until next power-on following debugger access.

Forcing boot from flash memory

The BOOT_LOCK bit allows forcing a unique entry point in the main flash memory for boot, regardless of the other boot mode configuration bits (refer to [Section 3.5.6: Forcing boot from main flash memory](#)).

Empty check

The internal empty check flag (the EMPTY bit of the [FLASH access control register \(FLASH_ACR\)](#)) is implemented to allow easy programming of virgin devices by the bootloader. This flag is checked when the boot configuration defines the main flash memory as the target boot area and the BOOT_LOCK bit is not set. When the EMPTY flag is set, the device is considered empty and the system memory (bootloader) is selected as boot area instead of the main flash memory, to allow the programming of the device by the user. Refer to AN2606 for more details concerning the bootloader and GPIO configuration in system memory boot mode (some of the GPIOs are reconfigured from the high-Z state).

The EMPTY flag is updated by hardware only during the loading of option bytes: it is set when the full 72-bit content (including ECC) of the address 0x0800 0000 is read as 0xFF FFFF FFFF FFFF, otherwise it is cleared. It means that, after programming of a virgin device, a power-on reset or setting of the OBL_LAUNCH bit in the FLASH_CR register is needed to clear the EMPTY flag (a system reset has no impact on this flag). The software can also modify the EMPTY flag directly in the FLASH_ACR register.

Note: If the device is programmed for the first time but the EMPTY flag is not updated, the device still selects system memory as a boot area after a system reset.

Physical remap

Once the boot mode is selected, the application software can modify the memory accessible in the code area. This modification is performed by programming the MEM_MODE bits in the [SYSCFG configuration register 1 \(SYSCFG_CFGR1\)](#).

Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. It is used to reprogram the flash memory using one of the following serial interfaces:

- USART, I2C and SPI (applies to all devices)
- USB (DFU) (applies to STM32U073xx and STM32U083xx)

For further details, refer to the device datasheets and to AN2606.

3 Embedded flash memory (FLASH)

3.1 FLASH introduction

The flash memory interface manages CPU (Cortex[®]-M0+) AHB accesses to the flash memory. It implements erase and program flash memory operations, read and write protection, and 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 256 Kbytes of flash memory (main memory)
- Memory organization:
 - Main memory block:
 - Up to 32 Kbytes (4 Kbytes x 72 bits) on STM32U031xx devices
 - Up to 256 Kbytes (32 bytes x 72 bits) on STM32U073xx and STM32U083xx devices
 - Information block:
 - 32 Kbytes (4 Kwords x 72 bits)
 - Page size: 2 Kbytes
- 72-bit-wide data read (64 bits plus eight ECC bits)
- Page erase (2 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)
- Securable memory area
- Flash memory empty check
- Prefetch buffer
- CPU instruction cache: 32 cache lines of 4 x 64 bits (1-Kbyte RAM)
- Error code correction (ECC): eight bits for 64 bits
- Option byte loader

3.3 FLASH functional description

3.3.1 Flash memory organization

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

The flash memory is organized as follows:

- Main memory block containing up to 128 pages of 2 Kbytes, each page with eight rows of 256 bytes
- Information block containing:
 - System memory from which the CPU boots in system memory bootloader used to reprogram the flash memory through one of the following interfaces: USART, SPI, I2C, and USB. On the manufacturing line, the devices are programmed 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. The OTP area cannot be read when the RDP level is 1 and a boot source is not the main flash memory area.
 - Option bytes for user configuration.

The following table shows the mapping of the flash memory into information block and main memory area.

Table 7. Flash memory organization: information block

| Area | Addresses | Size (Kbytes) | Memory type |
|-------------------|---------------------------|---------------|---------------|
| Information block | 0x1FFF 6C00 - 0x1FFF 7FFF | 5 | Non-user area |
| | 0x1FFF 6800 - 0x1FFF 6BFF | 1 | OTP area |
| | 0x1FFF 0000 - 0x1FFF 67FF | 26 | System memory |

Table 8. Flash memory organization: main memory

| Area | Addresses | Size (Kbytes) | 64 Kbyte devices | 128 Kbyte devices | 256 Kbyte devices |
|-------------|---------------------------|---------------|------------------|-------------------|-------------------|
| Main memory | 0x0803 F800 - 0x0803 FFFF | 2 | - | - | Page 127 |
| | ... | 2 | | | ... |
| | 0x0802 0000 - 0x0802 7FFF | 2 | | Page 64 | |
| | 0x0801 F800 - 0x0801 FFFF | 2 | | | Page 63 |
| | ... | ... | | Page 32 | ... |
| | 0x0801 0000 - 0x0801 07FF | 2 | | | |
| | 0x0800 F800 - 0x0800 FFFF | 2 | | Page 31 | |
| | ... | ... | | ... | |
| | 0x0800 1000 - 0x0800 17FF | 2 | | Page 2 | |
| | 0x0800 0800 - 0x0800 0FFF | 2 | | Page 1 | |
| | 0x0800 0000 - 0x0800 07FF | 2 | | Page 0 | |

3.3.2 FLASH empty check

During the option byte loading phase, after loading all options, the flash memory interface checks whether the first location (72 bits) 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 main flash memory area when no user code has been programmed.

The main flash memory empty check status can be read from the EMPTY bit in [FLASH access control register \(FLASH_ACR\)](#). Software can modify the main flash memory empty status by writing an appropriate value to the EMPTY bit.

3.3.3 FLASH error code correction (ECC)

Data in flash memory words are 72-bits 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 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 ECC register \(FLASH_ECCR\)](#). In this case, an NMI is generated.

When an ECC error is detected, the address of the failing double word is saved in ADDR_ECC[16:0] bitfield of the FLASH_ECCR register. ADDR_ECC[2:0] are always cleared.

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 FLASH read access latency

To correctly read data from flash memory, the number of wait states (LATENCY) must be correctly programmed in the [FLASH access control register \(FLASH_ACR\)](#) according to the frequency of the flash (HCLK) memory clock and the internal voltage range of the device V_{CORE}. Refer to [Section 4.1.6: Dynamic voltage scaling management](#). [Table 9](#) shows the correspondence between wait states and flash memory clock frequency.

Table 9. Number of wait states according to flash memory clock (HCLK) frequency

| Wait states (WS) (LATENCY) | HCLK (MHz) | |
|-------------------------------|---------------------------|---------------------------|
| | V _{CORE} Range 1 | V _{CORE} Range 2 |
| 0 WS (1 CPU cycle) | ≤ 24 | ≤ 8 |
| 1 WS (1 CPU cycle) | ≤ 48 | ≤ 16 |
| 2 WS (1 CPU cycle) | ≤ 56 | ≤ 18 |

After power reset, the HCLK clock frequency is 16 MHz in Range 1 and 0 wait state (WS) is configured in the FLASH_ACR register.

When waking up from Standby, the HCLK clock frequency is 16 MHz in Range 1 and 0 wait state (WS) is configured in the FLASH_ACR register.

When changing the flash memory clock frequency or Range, the following software sequences 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 of the [FLASH 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 bits of the [FLASH access control register \(FLASH_ACR\)](#), and wait until the programmed new number is read.
3. Modify the system clock source by writing the SW bits of the RCC_CFGR register.
4. If needed, modify the core clock prescaler by writing the HPRE bits of the RCC_CFGR register.
5. Optionally, check that the new system clock source or/and the new core clock prescaler value is/are taken into account by reading the clock source status (SWS bits) of the RCC_CFGR register, or/and the AHB prescaler value (HPREF bit), of the RCC_CFGR register.

Decreasing the CPU frequency

1. Modify the system clock source by writing the SW bits of the RCC_CFGR register.
2. If needed, modify the core clock prescaler by writing the HPRE bits of RCC_CFGR.
3. Check that the new system clock source or/and the new core clock prescaler value is/are taken into account by reading the clock source status (SWS bits) of the RCC_CFGR register, or/and the AHB prescaler value (HPREF bit), of the RCC_CFGR 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 of the *FLASH 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 bits of the *FLASH access control register (FLASH_ACR)*.

3.3.5 Flash memory acceleration

Instruction prefetch

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 CPU 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 of the *FLASH access control register (FLASH_ACR)*. This feature is useful if at least one wait state is needed to access the flash memory.

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.

Cache memory

To limit the time lost due to jumps, the STM32U0 microcontrollers feature 1 Kbytes of instruction cache memory. This feature can be enabled by setting the instruction cache enable (ICEN) bit of the *FLASH 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 are 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 enabled after system reset.

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

3.3.6 FLASH program and erase operations

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

The **in-circuit programming (ICP)** method is used to update the entire contents of the flash memory, using SWD protocol or the supported interfaces by the system bootloader, to load the user application for the CPU, 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 (I/Os, UART, I²C, SPI, etc.) to download programming data into memory. IAP allows the user to reprogram 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 success of a data word programming operation and a page/bank erase operation is not guaranteed if aborted due to device reset or power loss.

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 has completed.

Unlocking the flash memory

After reset, write into the *FLASH control register (FLASH_CR)* is not allowed so as to protect the flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence unlocks these registers:

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

Any wrong sequence locks 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 register 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 BSY1 bit of the FLASH status register (FLASH_SR) is set. Any attempt to write to this register with the BSY1 bit set causes the AHB bus to stall until the BSY1 bit is cleared.*

Caution: Due to the structure of the flash memory (64 bits of data associated to 8 bits of ECC), any ongoing memory operation that is interrupted by a physical event (such as a reset or a power off) may result in flash memory data corruption, and/or inconsistent ECC bits. The target address may trigger an ECC interrupt or an NMI when accessed later on. It is consequently important to always handle flash memory write/erase operations with care, and manage ECC events in the firmware.

3.3.7 FLASH main memory erase sequences

The flash memory 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 memory, OTP, and option bytes).

Flash memory page erase

When a page is protected by WRP, it is not erased and the WRPERR bit is set.

Table 10. Page erase overview

| HDP1_ACCDIS [7:0] | WRP | Comment | WRPERR | CPU bus error |
|-------------------------------|-----|---|--------|---------------|
| 0xA3 | No | Page is erased | No | No |
| | Yes | Page erase aborted (no page erase started) | Yes | |
| Any value different from 0xA3 | X | Protected pages only | No | Yes |

To erase a page (2 Kbytes), follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY1 bit of the [FLASH status register \(FLASH_SR\)](#).
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Check that the CFGBSY bit of [FLASH status register \(FLASH_SR\)](#) is cleared.
4. Set the PER bit and select the page to erase (PNB) in the [FLASH control register \(FLASH_CR\)](#).
5. Set the STRT bit of the [FLASH control register \(FLASH_CR\)](#).
6. Wait until the CFGBSY bit of the [FLASH status register \(FLASH_SR\)](#) is cleared again.

Note:

The HSI16 internal oscillator is automatically enabled when the STRT bit is set. It is automatically disabled when the STRT bit is cleared, except if previously enabled with the HSION bit of the RCC_CR register.

Flash memory bank or mass erase

When WRP is enabled, the flash memory mass erase is aborted, no erase starts, and the WRPERR bit is set.

Table 11. Mass erase overview

| HDP1_ACCDIS [7:0] | WRP | Comment | WRPERR | CPU bus error |
|-------------------------------|-----|----------------------------------|--------|---------------|
| 0xA3 | No | Memory is erased | No | No |
| | Yes | Erase aborted (no erase started) | Yes | |
| Any value different from 0xA3 | X | Erase aborted (no erase started) | No | Yes |

To perform a mass erase, respect the following procedure:

1. Check that no flash memory operation is ongoing by checking the BSY1 bit of the *FLASH status register (FLASH_SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Check that the CFGBSY bit of *FLASH status register (FLASH_SR)* is cleared.
4. Set the MER1 bit of the *FLASH control register (FLASH_CR)*.
5. Set the STRT bit of the *FLASH control register (FLASH_CR)*.
6. Wait until the CFGBSY bit of the *FLASH status register (FLASH_SR)* is cleared again.

Note:

The HSI16 internal oscillator is automatically enabled when the STRT bit is set. It is automatically disabled when the STRT bit is cleared, except if previously enabled with the HSION bit of the RCC_CR register.

3.3.8 FLASH main memory programming sequences

The flash memory is programmed 72 bits (64 bits plus 8-bit ECC).

Programming a previously programmed address with a nonzero data is not allowed. Any such attempt sets PROGERR flag of the *FLASH status register (FLASH_SR)*.

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 of the *FLASH status register (FLASH_SR)*.
- Any attempt to write a double word that is not aligned with a double word address sets PGAERR flag of the *FLASH status register (FLASH_SR)*.

Standard programming

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

1. Check that no main flash memory operation is ongoing by checking the BSY1 bit of the *FLASH status register (FLASH_SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Check that the CFGBSY bit of *FLASH status register (FLASH_SR)* is cleared.
4. Set the PG bit of the *FLASH control register (FLASH_CR)*.
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 the CFGBSY bit of the *FLASH status register (FLASH_SR)* is cleared again.
7. Check that the EOP flag in the *FLASH status register (FLASH_SR)* is set (programming operation succeeded), and clear it by software.
8. Clear the PG bit of the *FLASH control register (FLASH_CR)* if there no more programming request anymore.

Note: When the flash memory interface has received a good sequence (a double word), programming is automatically launched, and the BSY1 bit set.

The HSI16 internal oscillator is automatically enabled when the PG bit is set. It is automatically disabled when the PG bit is cleared, except if previously enabled with the HSION bit of the RCC_CR register.

Fast programming

The main purpose of this mode is to reduce the page programming time. It is achieved by eliminating the need for verifying the flash memory locations before they are programmed, thus saving the time of high voltage ramping and falling for each double word.

This mode allows programming a row (32 double words = 256 bytes).

During fast programming, the flash memory clock (HCLK) frequency must be at least 8 MHz.

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

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

1. Perform a mass or page erase. If not, PGSER is set.
2. Check that no main flash memory operation is ongoing by checking the BSY1 bit of the *FLASH status register (FLASH_SR)*.
3. Check and clear all error programming flag due to a previous programming.
4. Check that the CFGBSY bit of *FLASH status register (FLASH_SR)* is cleared.
5. Set the FSTPG bit in *FLASH control register (FLASH_CR)*.
6. Write 16 double-words to program a row (128 bytes).
7. Wait until the CFGBSY bit of the *FLASH status register (FLASH_SR)* is cleared again.
8. Check that the EOP flag in the *FLASH status register (FLASH_SR)* is set (programming operation succeeded), and clear it by software.
9. Clear the FSTPG bit of the *FLASH status register (FLASH_SR)* if there are no more programming requests anymore.

Note: When attempting to write in Fast programming mode while a read operation is ongoing, 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 BSY1 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 HSI48 internal oscillator (with a divide by three to provide 16 MHz) is automatically enabled when the FSTPG bit is set. It is automatically disabled when the FSTPG bit is cleared, except if previously enabled with the HSION bit of the RCC_CR register.

The 16 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 16 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 the next double word to programmed is not programmed. Anyway, all previous double words have been properly programmed.

Programming errors

Several kinds 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 is not previously erased (except if the value to program is full zero and the target address is in the main memory).

- **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:

- 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: a data is written when PG and FSTPG are cleared.
- In the standard programming sequence or the fast programming sequence: MER1 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 MER1 is set.
- In the page erase sequence: PG, FSTPG, and MER1 are not cleared when PER is set.
- PGSERR is also set if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR, or PGSERR is set due to a previous programming error.

- **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).
- Attempt to perform a mass erase when one page or more is protected by WRP.
- The debug features are connected or the boot is executed from SRAM or from system flash memory 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.

- **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 8 ms, which generates a time-out detection
- when the row fast programming has been interrupted by a MISSERR, PGAERR, WRPERR, or SIZERR

- **OPERR:** Generic error related to program/erase operations.

This flag can be set only if the error interrupt enable flag is set (ERRIE). In this case, it is set on any error condition (SEQERR, PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, PGSERR or FASTERR). Once set, it can only be cleared by programming it to 1.

- **OEMOPTWERR:** Key setting error

This flag is set when an invalid configuration is programmed to the OEM keys. It is set when attempting:

- to modify the OEM1KEYWxR registers when the RDP level 0 is disabled and the OEM1LOCK bit of the FLASH_SR is set.
- to modify the OEM2KEYWxR registers when the RDP level 0 is disabled and the OEM2LOCK bit of the FLASH_SR is set.
- to perform a regression from RDP level 1 to RDP level 0, while the OEM1LOCK bit of FLASH_SR is set and a wrong OEM1KEY is transmitted through JTAG or SWD.

- **HDOPTWERR:** HDP setting error

This flag is set when an invalid configuration is programmed to the HDP option bytes. It is set when attempting:

- to modify the HDP1EN[7:0] and/or the HDP1_PEND[6:0] bitfield of the FLASH_SECR register, when the HDP1_ACCDIS[7:0] and the HDP1EXT_ACCDIS[7:0] bitfields of the FLASH_HDPCR register are not equal to their reset value (0xA3).
- to set the HDP1_ACCDIS[7:0] and/or the HDP1EXT_ACCDIS[7:0] bitfields to a value different from their reset value (0xA3), while the HDP1EN[7:0] and/or the HDP1_PEND[6:0] bitfields have been changed, and no option byte modification and loading (OPTSTR and OBL_LAUNCH bits set in the FLASH_CR registers) has been done in between.

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

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

In this case, if the error interrupt enable bit ERRIE of the *FLASH control register (FLASH_CR)* is set, an interrupt is generated and the operation error flag OPERR of the *FLASH status register (FLASH_SR)* is set.

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 and cache

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

Note: The cache should be flushed only when it is disabled (ICEN = 0).

3.4 FLASH option bytes

3.4.1 FLASH option byte 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: FLASH option byte programming](#)).

A double word is split up in option bytes as indicated in [Table 12](#).

Table 12. Option byte 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 |

[Table 13](#) shows the organization of the option bytes (the lower word only) in the flash memory information block. The software can read the option bytes from these flash memory locations or from their corresponding option registers referenced in the table. Refer to sections [FLASH option register \(FLASH_OPTR\)](#) to [FLASH security register \(FLASH_SECR\)](#) for the description of the option register bitfields, also applicable to the option byte bitfields.

Table 13. Organization of option bytes

| Corresponding option register (section) | 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 |
|---|---------------|-------|----------------|--------|-----------------|-----------|--------------------------|-----------------|------------------|----------|---------|---------|------------------|-----------|---------|-----------|------------|-----------|----------|----------|----------|----|---|---|---|---|---|---|---|---|---|---|
| | Option byte 3 | | | | Option byte 2 | | | | Option byte 1 | | | | Option byte 0 | | | | | | | | | | | | | | | | | | | |
| FLASH_OPTR (Section 3.7.7) | Reserved | IRHEN | NRST_MODE[1:0] | NBOOT0 | NBOOT1 | NBOOT_SEL | BKPSRAM_HW_ERASE_DISABLE | RAM_PARTY_CHECK | NOT_VBAT_VDD_OPT | Reserved | WWDG_SW | IWDG_SW | IWDG_STBY | IWDG_STOP | IWDG_SW | NRST_SHDW | NRST_STDBY | NRST_STOP | Reserved | RRP[7:0] | RDP[7:0] | | | | | | | | | | | |
| Factory value | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | X | X | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | | | |
| FLASH_WRP1AR (Section 3.7.8) | Reserved | | | | WRP1A_END [6:0] | | | | Reserved | | | | WRP1A_STRT [6:0] | | | | | | | | | | | | | | | | | | | |
| Factory value | X | X | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | | | | |

Table 13. Organization of option bytes (continued)

| Corresponding option register (section) | 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 |
|--|-----------------|----|----|----|-----------------|----|----|----|---------------|----------|----|----|------------------|-----------------|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| | Option byte 3 | | | | Option byte 2 | | | | Option byte 1 | | | | Option byte 0 | | | | | | | | | | | | | | | | | | | |
| FLASH_WRP1BR (<i>Section 3.7.9</i>) | Reserved | | | | WRP1B_END [6:0] | | | | Reserved | | | | WRP1B_STRT [6:0] | | | | | | | | | | | | | | | | | | | |
| Factory value | X | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| FLASH_SECR (<i>Section 3.7.10</i>) | HDP1EN[7:0] | | | | Reserved. | | | | BOOT_LOCK | Reserved | | | | HDP1_PEND [6:0] | | | | | | | | | | | | | | | | | | |
| Factory value | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | | X | X | X | X | X | X | 0 | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FLASH_OEM1KEYR1 | OEM1KEY[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Factory 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 | | |
| FLASH_OEM1KEYR2 | OEM1KEY[63:32] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Factory 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 | | |
| FLASH_OEM1KEYR3 | OEM1KEY[95:64] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Factory 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 | | |
| FLASH_OEM1KEYR4 | OEM1KEY[127:96] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Factory 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 | | |
| FLASH_OEM2KEYR1 | OEM2KEY[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Factory 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 | | |
| FLASH_OEM2KEYR2 | OEM2KEY[63:32] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Factory 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 | | |
| FLASH_OEM2KEYR3 | OEM2KEY[95:64] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Factory 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 | | |
| FLASH_OEM2KEYR4 | OEM2KEY[127:96] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Factory 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 | | |

3.4.2 FLASH option byte programming

After reset, the option related bits of the [*FLASH control register \(FLASH_CR\)*](#) are write-protected. To run any operation on the option byte page, the option lock bit OPTLOCK of the [*FLASH 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 = 0x0819 2A3B of the [*FLASH option key register \(FLASH_OPTKEYR\)*](#).
3. Write OPTKEY2 = 0x4C5D 6E7F of the [*FLASH 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.

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 values in the [*FLASH* option registers](#).
3. Check that no flash memory operation is ongoing, by checking the BSY1 bit of the [*FLASH status register \(FLASH_SR\)*](#).
4. Set the options start bit OPTSTRT of the [*FLASH control register \(FLASH_CR\)*](#).
5. Wait for the BSY1 bit to be cleared.

Note:

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

The complementary values are automatically computed and written into the complemented option bytes upon setting the OPTSTRT bit.

Caution:

Upon an option byte programming failure (for any reason, such as loss of power or a reset during the option byte change sequence), the mismatch values of the option bytes are loaded after reset. Those mismatch values force a secure configuration that might permanently lock the device. To prevent this, only program option bytes in a safe environment-safe supply, no pending watchdog, and clean reset line.

Option byte loading

After the BSY1 bit is cleared, all new options are updated into the flash memory, but not applied to the system. A read from the option registers still returns the last loaded option byte values. The new options has effect on the system only after they are loaded.

Option byte loading is performed in two cases:

- when OBL_LAUNCH bit of the *FLASH control register (FLASH_CR)* is set
- 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 that the loading has correctly taken place.

During option byte loading, the options are read by double word.

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 1 for all option bits.
- 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 BOOT_LOCK, the value of mismatch is “boot forced from main flash memory”.

Upon system reset, the option bytes are copied into the following option registers that can be read and written by software:

- FLASH_OPTR
- FLASH_WRP1xR (x = A or B)
- FLASH_SECR

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 [Section : Modifying user options](#) for more details.

3.5 Flash memory protection

The main flash 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 2 Kbytes. Apart from the RDP and WRP, the flash memory also features an HDP securable area.

3.5.1 FLASH 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 main flash memory, the option bytes, and the backup registers.

Note: *If the read protection is set while the debugger is still connected through JTAG/SWD (or has been connected since the last power-on), apply a POR (power-on reset) instead of a system reset. If the read protection is programmed by software, do not set the OBL_LAUNCH bit of the [FLASH control register \(FLASH_CR\)](#). Instead perform a POR to reload the option byte. This can be done by a switch 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 14](#).

Table 14. Flash memory read protection status

| RDP byte value | RDP complement byte value | Read protection level |
|--|---------------------------|-----------------------|
| 0xAA | 0x55 | Level 0 |
| Any values except the combinations [0xAA, 0x55] and [0xCC, 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 within the main flash memory area are possible. The option bytes are also accessible by all operations.

Level 1: Read protection

Level 1 read protection is set when the RDP byte and the RDP complemented byte contain any value combinations other than [0xAA, 0x55] and [0xCC, 0x33]. Level 1 is the default protection level when RDP option byte is erased.

- **User mode:** Code executing in user mode (boot from user flash memory) can access main flash memory and option bytes with all operations.
- **Debug, boot from SRAM, and boot from system memory modes:** In debug mode or when code boots from SRAM or system memory, the main flash memory is totally inaccessible. In these modes, a read or write access to the flash memory generates a bus error and a Hard Fault interrupt.

Level 2: No debug

In this level, the protection level 1 is guaranteed. In addition, the CPU debug port, the boot from RAM (boot RAM mode) and the boot from system memory (bootloader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the main flash memory.

Note: *The CPU debug port is also disabled under reset.*

STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set.

Changing the read protection level

The read protection level can change:

- from level 0 to level 1, upon changing the value of the RDP byte to any value except 0xCC
- from level 0 or level 1 to level 2, upon changing the value of the RDP byte to 0xCC
- from level 1 to level 0, upon changing the value of the RDP byte to 0xAA

Once in level 2, it is no more possible to modify the read protection level.

The OTP area is not affected by a mass erase and remains unchanged.

Note: *Mass erase (full or partial) is only triggered by the RDP regression from level 1 to level 0. RDP level increase (level 0 to level 1, 1 to 2, or 0 to 2) does not cause any mass erase.*

To validate the protection level change, the option bytes must be reloaded by setting the OBL_LAUNCH bit of the [FLASH control register \(FLASH_CR\)](#).

Figure 3. Changing read protection (RDP) level

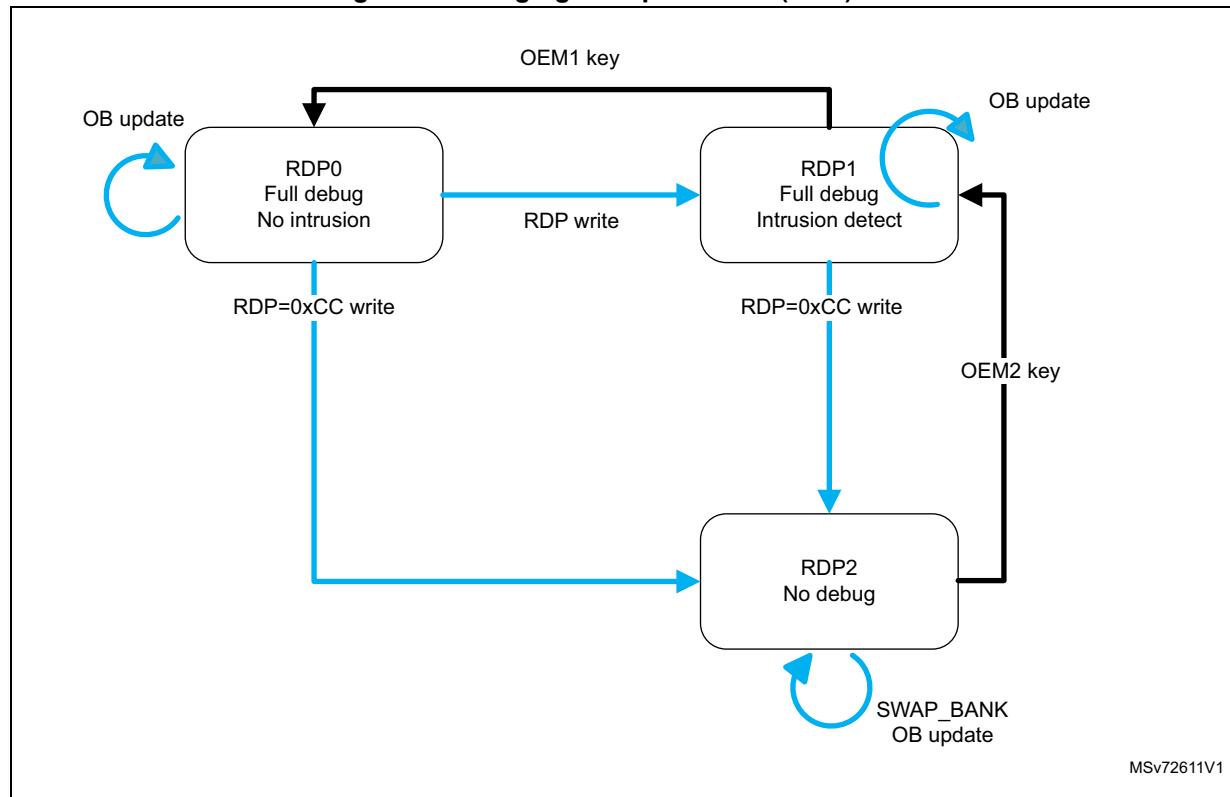


Table 15. Access status versus protection level and execution modes

| Area | Protection level | User execution (Boot from flash) | | | Debug/ Boot from RAM/ Boot from loader | | |
|----------------------------------|------------------|----------------------------------|--------------------|-------|--|--------------------|--------------------|
| | | Read | Write | Erase | Read | Write | Erase |
| Main flash memory | 1 | Yes | Yes | Yes | No | No | No ⁽³⁾ |
| | 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 | No | No | N/A ⁽¹⁾ | N/A ⁽¹⁾ | N/A ⁽¹⁾ |
| OTP, SRAM2, and backup registers | 1 | Yes | Yes | N/A | No | No | N/A |
| | 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 main memory is erased when the RDP option byte is programmed with all level of protections disabled (0xAA).

OEM1/2 lock activation

Two 128-bit keys (OEM1KEY and OEM2KEY) can be defined and used to lock the RDP regression. Each 128-bit key is coded on four registers: FLASH_OEM1KEYW0R (respectively FLASH_OEM2KEYW0R) to FLASH_OEM1KEYW3R (respectively FLASH_OEM2KEYW3R).

OEM1KEY and OEM2KEY cannot be read through these registers: they are always read as zero. However, to verify the keys, their CRC can be accessed through OEM1KEYCRC and OEM2KEYCRC of the FLASH_OEMKEYSR register.

OEM1KEY can be modified:

- in readout protection level 0
- in readout protection level 1 if OEM1LOCK = 0 in FLASH_SR

OEM2KEY can be modified:

- in readout protection level 0
- in readout protection level 1 if OEM2LOCK = 0 in FLASH_SR

When attempting to modify FLASH_OEMxKEYWyR ($x = 1$ or 2 , $y = 0$ to 3) without respecting the above rules, the user option modification is not performed, and the OEMOPTWERR bit is set.

Follow the steps below to activate the OEM1/2 lock mechanism:

1. Check that the OEM1LOCK/OEM2LOCK bit is not set, or that the readout protection is at level 0.
2. Write a 128-bit key to FLASH_OEM1KEYWxR/FLASH_OEM2KEYWxR ($y = 0$ to 3).
3. Launch the option modification by setting the OPTSTRT bit, and check that OEMOPTWERR is not set.
4. Set the OBL_LAUNCH or perform a power-on reset.
5. Check that OEM1LOCK/OEM2LOCK is set.

Note: *The OEM1KEY and OEM2KEY must not contain only 1 or 0.*

OEM1 RDP lock mechanism

The OEM1 RDP lock mechanism is active when the OEM1LOCK bit is set in the FLASH_SR register. It blocks the RDP level 1 to RDP level 0 regression.

Apply the following unlock sequence to perform an RDP level 1 to RDP level 0 regression:

1. Program the 128-bit OEM1 key in the DBGMCU_DBG_AUTH_HOST register. This operation must be done using the debug interface under reset.
2. If this key matches the OEM1KEY value, the RDP regression can be launched by setting the OPTSTRT bit of the FLASH_CR register.
3. If the key does not match the OEM1KEY value, the RDP regression and any access to the flash memory are blocked until the next power-on reset.

Attempting to perform a regression from RDP level 1 to RDP level 0 without following the above sequence sets the OEMOPTWERR option bit, and the option bytes remain unchanged.

When the lock mechanism is not activated (OEM1LOCK = 0), the regression from RDP level 1 to RDP level 0 is always granted.

OEM2 RDP lock mechanism

The OEM2 RDP lock mechanism is active when the OEM2LOCK bit is set in the FLASH_CR register. It allows the RDP level 2 to RDP level 1 regression.

The following unlock sequence must be applied:

1. Program the 128-bit OEM2 key in the DBGMCU_DBG_AUTH_HOST register. This operation must be done using the debug interface under reset.
2. If this key matches the OEM2KEY value, the RDP regression is launched automatically by hardware as soon as the reset signal goes high.
3. If the key does not match the OEM2KEY value, the device boots normally (as if no key has been provided).

3.5.2 FLASH 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 (2-Kbyte) 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 WRP area A address register \(FLASH_WRP1AR\)](#) and [FLASH WRP area B address register \(FLASH_WRP1BR\)](#).

The WRP *x* area (*x* = A, B) is defined from the address

*flash memory Base address + [WRP1*x*_START x 0x0800]* (included)

to the address

*flash memory Base address + [(WRP1*x*_END+1) x 0x0800]* (excluded).

The minimum WRP area size is one WRP page (2 Kbytes):

*WRP1*x*_END = WRP1*x*_START.*

For example, to protect the flash memory by WRP from the address 0x0800 1000 (included) to the address 0x0800 3FFF (included):

If boot in flash memory is selected, FLASH_WRP1AR register must be programmed with:

- WRP1A_STRT = 0x02.
- WRP1A_END = 0x07.

WRP1B_STRT and WRP1B_END in FLASH_WRP1BR can be used instead (area B in the 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 (WRPPER) of the FLASH_SR register is set. 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

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 (single wire) or boot code is being executed from SRAM or system flash memory, even if WRP is not activated. Any attempt generates a hard fault (BusFault).

Table 16: WRP protection

| WRP registers values (x = A or B) | WRP-protected area |
|--------------------------------------|------------------------------------|
| WRP1x_STRT = WRP1x_END | Page WRP1x |
| WRP1x_STRT > WRP1x_END | None (unprotected) |
| WRP1x_STRT < WRP1x_END | Pages from WRP1x_STRT to WRP1x_END |

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

3.5.3 Securable memory area (HDP)

The main purpose of the securable memory area is to protect a specific part of flash memory against undesired access. After system reset, the code in the securable memory area can only be executed until the securable area becomes secured and never again until the next system reset. This allows implementing software security services such as secure key storage or safe boot.

The securable memory area is located in the first pages of the main flash memory. When not secured, the securable memory behaves like the rest of main flash memory. When secured (the HDP1_ACCDIS[7:0] value in the FLASH_HDPCR register is different from 0xA3, and the HDP1EN[7:0] value in FLASH_SECR register is different from 0xB4), any write access (programming, erase) to the securable memory area is rejected and generates a bus error. Fetch and read accesses return zero (read-as-zero RAZ). The securable area can only be unsecured by a system reset.

The last page of the securable memory area is defined by the HDP1_PEND[6:0] bitfield of the FLASH_SECR register. The protection is enabled by the HDP1EN[7:0] bitfield in the

same register. After updating the FLASH_SECR register, OPTSTRT and OBL_LAUNCH are required before activating the protection through HDP1_ACCDIS[7:0] or HDP1EXT_ACCDIS[7:0].

Note: *The securable memory area start address is 0x0800 0000. Before activating the securable memory area, move the vector table outside the page 0 if necessary.*

3.5.4 Securable memory area extension (HDP extension)

The HDP1 area can be extended through HDP1_EXT[6:0] of the [FLASH HDP extension register \(FLASH_HDPEXTR\)](#). HDP1_EXT[6:0] indicates the number of pages added to the HDP area.

The start page offset of the HDP1 extension area depends on the presence of an HDP1 area. If there is an HDP1 area, the HDP1 extension area starts from HDP1_PEND + 1. Otherwise, it starts from the user flash memory base address.

The behavior of the extension is controlled by the HDP1EXT_ACCDIS[7:0] value (refer to [Table 17](#)).

Table 17. HDP extension protection

| HDP1EXT_ACCDIS [7:0] state | Write to HDP1EXT_ACCDIS [7:0] | Write to HDP1_EXT[6:0] | Access to HDP memory area |
|----------------------------|--------------------------------------|--------------------------|---------------------------|
| 0xA3 | OK | OK | Allowed |
| 0x5C | OK, if the value written is not 0xA3 | OK to write larger value | Denied |
| Others | Write ignored | Write ignored | Denied |

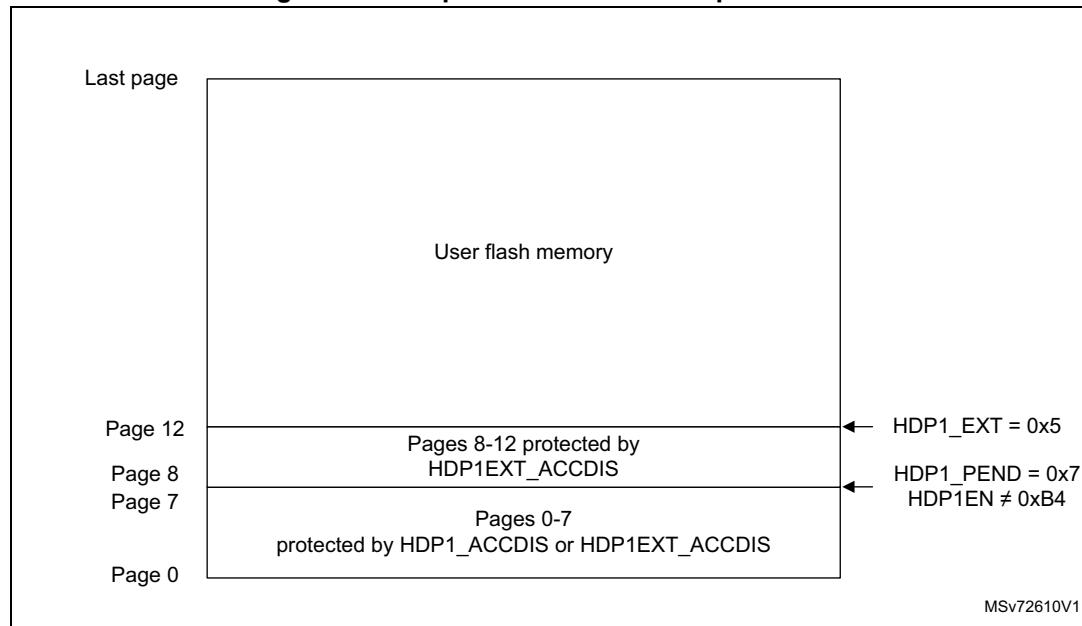
There are two determining HDP1EXT_ACCDIS[7:0] values:

- HDP1EXT_ACCDIS[7:0] = 0xA3 (reset value): the HDP area protection is not active and can be reconfigured freely.
- HDP1EXT_ACCDIS[7:0] = 0x5C: the HDP protection is active, but the extension can be expanded if needed.
- HDP1EXT_ACCDIS[7:0] = any other value: the HDP protection is active, and there is no way to modify the settings until the next reset.

When active, the extension also prevents any modification of the HDP1 option bytes, HDP1EN[7:0] and HDP1_PEND[6:0].

The access to the securable memory area extension is disabled if HDP1_ACCDIS[7:0] is set to 0xA3 (access granted), and the HDP1EXT_ACCDIS[7:0] is set to a value different from 0xA3 (access disabled).

Refer to [Figure 4](#) for an example:

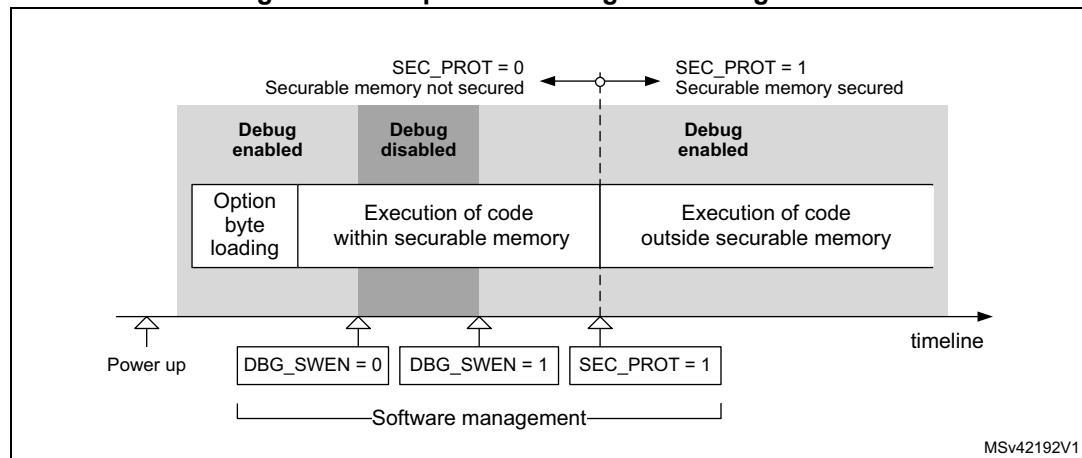
Figure 4. Example of HDP extension protection

As in case of HDP area configured by option bytes, any attempt to program protected pages is treated as a bus error, and fetch and read operations return zero (read-as-zero RAZ).

3.5.5 Disabling core debug access

For executing sensitive code or manipulating sensitive data in securable memory area, the debug access to the core can temporarily be disabled.

Figure 5 gives an example of managing DBG_SWEN and SEC_PROT bits.

Figure 5. Example of disabling core debug access

3.5.6 Forcing boot from main flash memory

To increase the security and establish a chain of trust, the BOOT_LOCK option bit of the FLASH_SECR register allows forcing the system to boot from the main flash memory regardless the other boot options. It is always possible to set the BOOT_LOCK bit. However, it is possible to reset it only when:

- RDP is set to level 0, or
- RDP is set to level 1, while level 0 is requested and a full mass-erase is performed.

Caution: If BOOT_LOCK is set in association with RDP Level 1, the debug capabilities of the device are stopped, and the reset value of the DBG_SWEN bit of the FLASH_ACR register becomes zero. If the DBG_SWEN bit is cleared by the application code after reset, there is no way to recover from this situation.

3.6 FLASH interrupts

Table 18. FLASH 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 |
| 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 |

1. EOP is set only if EOPIE is set.

2. OPERR is set only if ERRIE is set.

3.7 FLASH registers

3.7.1 FLASH access control register (FLASH_ACR)

Address offset: 0x000

Reset value: 0x0004 0600

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|----------|------|-------|
| Res. | DBG_SWEN | Res. | EMPTY |
| | | | | | | | | | | | | | rw | | rw |

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

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

Bit 18 **DBG_SWEN**: Debug access software enable

Software may use this bit to enable/disable the debugger read access.

0: Debugger disabled

1: Debugger enabled

Bit 17 Reserved, must be kept at reset value.

Bit 16 **EMPTY**: Main flash memory area empty

This bit indicates whether the first location of the main flash memory area is erased or has a programmed value.

0: Main flash memory area programmed

1: Main flash memory area empty

The bit can be set and reset by software.

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

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

0: CPU Instruction cache is not reset

1: CPU Instruction cache is reset

This bit is write-only. It can be written only when the instruction cache is disabled. It is always read as zero.

Bit 10 Reserved, must be kept at reset value.

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

0: CPU Instruction cache is disabled

1: CPU Instruction cache is enabled

Bit 8 **PRFTEN**: CPU Prefetch enable

0: CPU Prefetch disabled

1: CPU Prefetch enabled

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

Bits 2:0 **LATENCY[2:0]**: Flash memory access latency

The value in this bitfield represents the number of CPU wait states when accessing the flash memory.

000: Zero wait states

001: One wait state

Other: Reserved

A new write into the bitfield becomes effective when it returns the same value upon read.

3.7.2 FLASH 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 key

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

KEY1: 0x4567 0123

KEY2: 0xCDEF 89AB

3.7.3 FLASH 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

3.7.4 FLASH 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. | OEM2 LOCK | OEM1 LOCK | Res. | CFGBSY | Res. | BSY1 |
| | | | | | | | | | | r | r | | r | | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OPTV ERR | Res. | Res. | OEMOP TWERR | HDP OPT WERR | 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 | | rc_w1 | rc_w1 |

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OEM2LOCK:** OEM2 lock flag

This bit indicates that the OEM2 RDP key, read during the option byte loading, is not virgin.
When set, the OEM2 RDP lock mechanism is active.

Bit 20 **OEM1LOCK:** OEM1 lock flag

This bit indicates that the OEM1 RDP key, read during the option byte loading, is not virgin.
When set, the OEM1 RDP lock mechanism is active.

Bit 19 Reserved, must be kept at reset value.

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

This flag is set and cleared by hardware. It is set when the first word is sent for program or when setting the STRT bit of [FLASH control register \(FLASH_CR\)](#) for erase. It is cleared when the flash memory program or erase operation completes or ends with an error.
When set, launching any other operation through the [FLASH control register \(FLASH_CR\)](#) is impossible, and must be postponed (a programming or erase operation is ongoing).
When cleared, the program and erase settings in the [FLASH control register \(FLASH_CR\)](#) can be modified.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **BSY1:** Busy flag

This flag indicates that a flash memory operation requested by [FLASH control register \(FLASH_CR\)](#) is in progress. This bit is set at the beginning of the flash memory operation, and cleared when the operation finishes or when an error occurs.

Bit 15 **OPTVERR:** Option and engineering bits loading validity error flag

This flag is 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 haven't 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: FLASH option byte programming](#).
It is cleared by programming it to 1.

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

Bit 12 **OEMOPTWERR:** OEM options byte write error

This flag is set by hardware when the requested modifications of the OEM keys are not allowed.

Bit 11 **HDPOPTWERR:** HDP option bytes write error

This flag is set by hardware when the requested HDP related option modifications are not allowed. It is cleared by programming it to 1.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **FASTERR:** Fast programming error flag

This flag is 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.

It is cleared by programming it to 1.

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

In Fast programming mode, 16 double words (128 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.
It is cleared by programming it to 1.

Bit 7 PGSERR: Programming sequence error flag

This flag is 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.

It is cleared by programming it to 1.

Bit 6 SIZERR: Size error flag

This flag is 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).

It is cleared by programming it to 1.

Bit 5 PGAERR: Programming alignment error flag

This flag is 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.

It is cleared by programming it to 1.

Bit 4 WRPERR: Write protection error flag

This flag is set by hardware when an address to be erased/programmed belongs to a write-protected part (by WRP or RDP level 1) of the flash memory.

It is cleared by programming it to 1.

Bit 3 PROGERR: Programming error flag

This flag is 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.

It is cleared by programming it to 1.

Bit 2 Reserved, must be kept at reset value.**Bit 1 OPERR:** Operation error flag

This flag is set by hardware when a flash memory operation (program / erase) completes unsuccessfully.

This bit is set only if error interrupts are enabled (ERRIE = 1).

It is cleared by programming it to 1.

Bit 0 EOP: End of operation flag

This flag is 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).

It is cleared by programming it to 1.

3.7.5 FLASH control register (FLASH_CR)

Address offset: 0x014

Reset value: 0xC000 0000

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

This register must not be modified when the CFGBSY bit of the *FLASH status register (FLASH_SR)* is set. Otherwise, this would result in:

- a bus error, when the CFGBSY bit is set due to a double-word programming sequence, and the write to the *FLASH control register (FLASH_CR)* is performed after the first word sent for programming and before the second.
- a bus stall, when the CFGBSY bit is set for any other reason.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
|------|----------|------|------|------------|------|-------|-------|----------|------|------|------|------|-------|----------|------|----|
| LOCK | OPT LOCK | Res. | Res. | OBL_LAUNCH | Res. | ERRIE | EOPIE | Res. | Res. | Res. | Res. | Res. | FSTPG | OPT STRT | STRT | |
| rs | rs | | | rc_w1 | | 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. | Res. | | | PNB[6:0] | | | | | | MER1 | 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 user option 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**: Option byte load launch

When set, this bit triggers the load of option bytes into option registers. It is automatically cleared upon the completion of the load. The high state of the bit indicates pending option byte load.

The bit cannot be cleared by software. It cannot be written as long as OPTLOCK is set.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **ERRIE**: Error interrupt enable

This bit enables the interrupt generation upon setting the OPERR flag in the FLASH_SR register.

0: Disable

1: Enable

Bit 24 **EOPIE**: End-of-operation interrupt enable

This bit enables the interrupt generation upon setting the EOP flag in the FLASH_SR register.

0: Disable

1: Enable

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

Bit 31 **ECCD**: ECC detection

This flag is set by hardware when two ECC errors have been detected. When this bit is set, a NMI is generated.

It is cleared by programming it to 1.

Bit 30 **ECCC**: ECC correction

This flag is set by hardware when one ECC error has been detected and corrected. An interrupt is generated if ECCIE is set.

It is cleared by programming it to 1.

Bits 29:25 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 19:14 Reserved, must be kept at reset value.

Bits 13:0 **ADDR_ECC[13:0]**: ECC fail double-word address offset

In case of ECC error or ECC correction detected, this bitfield contains double-word offset (multiple of 64 bits) to main flash memory.

3.7.7 FLASH option register (FLASH_OPTR)

Address offset: 0x020

Reset value: 0xFFFF XXXX (The option bits are loaded with values from flash memory at power-on 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 |
|-----------|------------|-----------|-----------------|----------|-------------|-----------|--------------------------|------------------|-------|------|---------|------------|-----------|---------|----|
| Res. | Res. | IRHEN | NRST_MODE [1:0] | N BOOT 0 | N BOOT 1 | NBOOT_SEL | BKPSRAM_HW_ERASE_DISABLE | RAM_PARITY_CHECK | BDRST | Res. | WWDG_SW | IWDG_STDBY | IWDG_STOP | IWDG_SW | |
| | | 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 |
| NRST_SHDW | NRST_STDBY | NRST_STOP | Res. | Res. | BORLEV[2:0] | | | RDP[7:0] | | | | | | | |
| 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 **IRHEN**: Internal reset holder enable bit

0: Internal resets are propagated as simple pulse on NRST pin

1: Internal resets drives NRST pin low until it is seen as low level

- Bits 28:27 **NRST_MODE[1:0]**: NRST pin configuration
00: Reserved
01: Reset input only: a low level on the NRST pin generates system reset; internal RESET is not propagated to the NRST pin.
10: Standard GPIO: only internal RESET is possible
11: Bidirectional reset: the NRST pin is configured in reset input/output (legacy) mode
- Bit 26 **NBOOT0**: NBOOT0 option bit
0: NBOOT0 = 0
1: NBOOT0 = 1
- Bit 25 **NBOOT1**: Boot configuration
Together with the BOOT0 pin or option bit NBOOT0 (depending on NBOOT_SEL option bit configuration), this bit selects boot mode from the main flash memory, SRAM or the system memory. Refer to [Section 2.5: Boot configuration](#).
- Bit 24 **NBOOT_SEL**: BOOT0 signal source selection
This option bit defines the source of the BOOT0 signal.
0: BOOT0 pin (legacy mode)
1: NBOOT0 option bit
- Bit 23 **BKPSRAM_HW_ERASE_DISABLE**: Backup SRAM erase prevention
0: Backup SRAM is erased on system reset
1: Backup SRAM content is kept when a system reset occurs
The bit has no influence on other Backup SRAM erase requests, such as tamper or RDP regression.
- Bit 22 **RAM_PARITY_CHECK**: SRAM parity check control enable/disable
0: Enable
1: Disable
- Bit 21 **BDRST**: Backup domain reset
0: Backup domain not reset on shutdown exit
1: Reset of backup domain (RTC registers and backup registers) forced on shutdown exit
- Bit 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 **NRST_SHDW**: Reset generated when entering Shutdown mode
0: Reset generated when entering the Shutdown mode
1: No reset generated when entering the Shutdown mode

- Bit 14 **NRST_STDBY**: Reset generated when entering Standby mode
 0: Reset generated when entering the Standby mode
 1: No reset generate when entering the Standby mode
- Bit 13 **NRST_STOP**: Reset generated when entering Stop mode
 0: Reset generated when entering the Stop mode
 1: No reset generated when entering the Stop mode
- Bits 12:11 Reserved, must be kept at reset value.
- Bits 10:8 **BOR_LEV[2:0]**: BOR reset level
 This bitfield contains the V_{DD} supply level threshold that activates/releases the reset.
 000: BOR level 0, reset level threshold around 1.7 V
 001: BOR level 1, reset level threshold around 2.0 V
 010: BOR level 2, reset level threshold around 2.2 V
 011: BOR level 3, reset level threshold around 2.5 V
 100: BOR level 4, reset level threshold around 2.8 V
- 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

3.7.8 FLASH WRP area A address register (FLASH_WRP1AR)

Address offset: 0x02C

Reset value: 0x000X 000X (the option bits are loaded with values from flash memory at power-on 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 | | | | | | | |
|------|------|------|------|------|------|------|------|------|-----------------|----|----|----|----|----|----|--|--|--|--|--|--|--|
| Res. | WRP1A_END[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. | WRP1A_STRT[6:0] | | | | | | | | | | | | | |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw | | | | | | | |

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP1A_END[6:0]**: WRP area A end offset

This bitfield contains the offset of the last page of the WRP area A.

Note: The number of effective bits depends on the size of the flash memory in the device.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **WRP1A_STRT[6:0]**: WRP area A start offset

This bitfield contains the offset of the first page of the WRP area A.

Note: The number of effective bits depends on the size of the flash memory in the device.

3.7.9 FLASH WRP area B address register (FLASH_WRP1BR)

Address offset: 0x030

Reset value: 0x000X 000X (the option bits are loaded with values from flash memory at power-on 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 | | | | | | | |
|------|------|------|------|------|------|------|------|------|-----------------|----|----|----|----|----|----|--|--|--|--|--|--|--|
| Res. | WRP1B_END[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. | WRP1B_STRT[6:0] | | | | | | | | | | | | | |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw | | | | | | | |

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP1B_END[6:0]**: WRP area B end offset

This bitfield contains the offset of the last page of the WRP area B.

Note: The number of effective bits depends on the size of the flash memory in the device.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **WRP1B_STRT[6:0]**: WRP area B start offset

This bitfield contains the offset of the first page of the WRP area B.

Note: The number of effective bits depends on the size of the flash memory in the device.

3.7.10 FLASH security register (FLASH_SECR)

Address offset: 0x080

Reset value: 0bXXXX XXXX 0000 000X 0000 0000 0XXX XXXX (the option bits are loaded with values from flash memory at power-on reset release.)

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | | | | | | |
|-------------|------|------|------|------|------|------|------|------|----------------|------|------|------|------|------|-----------|--|--|--|--|--|--|--|
| HDP1EN[7:0] | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BOOT_LOCK | | | | | | | |
| 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. | HDP1_PEND[6:0] | | | | | | | | | | | | | |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw | | | | | | | |

Bits 31:24 **HDP1EN[7:0]**: Hide protection area enable

0xB4: no HDP area

Others: HDP area enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **BOOT_LOCK**: Forced boot from user area

0: Boot based on the pad/option bit configuration

1: Boot forced from main flash memory

Caution: If BOOT_LOCK is set in association with RDP Level 1, the debug capabilities of the device are stopped, and the reset value of the DBG_SWEN bit of the FLASH_ACR register becomes zero. If the DBG_SWEN bit is cleared by the application code after reset, there is no way to recover from this situation.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **HDP1_PEND[6:0]**: Last page of the first hide protection area

This bitfield contains the last page of the HDP area.

3.7.11 FLASH OEM1 key register 1 (FLASH_OEM1KEYR1)

Address offset: 0x088

Reset value: 0xFFFF FFFF

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OEM1KEY[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 |
| OEM1KEY[15:0] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **OEM1KEY[31:0]**: Bits 31 to 0 of the OEM1 key

All bits are write-only and always read as zero.

3.7.12 FLASH OEM1 key register 2 (FLASH_OEM1KEYR2)

Address offset: 0x08C

Reset value: 0xFFFF FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OEM1KEY[63:48] | | | | | | | | | | | | | | | |
| 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 |
| OEM1KEY[47:32] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **OEM1KEY[63:32]**: Bits 63 to 32 of the OEM1 key
All bits are write-only and always read as zero.

3.7.13 FLASH OEM1 key register 3 (FLASH_OEM1KEYR3)

Address offset: 0x090

Reset value: 0xFFFF FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OEM1KEY[95:80] | | | | | | | | | | | | | | | |
| 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 |
| OEM1KEY[79:64] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **OEM1KEY[95:64]**: Bits 95 to 64 of the OEM1 key
All bits are write-only and always read as zero.

3.7.14 FLASH OEM1 key register 4 (FLASH_OEM1KEYR4)

Address offset: 0x094

Reset value: 0xFFFF FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OEM1KEY[127:112] | | | | | | | | | | | | | | | |
| 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 |
| OEM1KEY[111:96] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **OEM1KEY[127:96]**: Bits 127 to 96 of the OEM1 key
All bits are write-only and always read as zero.

3.7.15 FLASH OEM2 key register 1 (FLASH_OEM2KEYR1)

Address offset: 0x098

Reset value: 0xFFFF FFFF

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OEM2KEY[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 |
|---------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| OEM2KEY[15:0] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **OEM2KEY[31:0]**: Bits 31 to 0 of the OEM2 key

All bits are write-only and always read as zero.

3.7.16 FLASH OEM2 key register 2 (FLASH_OEM2KEYR2)

Address offset: 0x09C

Reset value: 0xFFFF FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OEM2KEY[63:48] | | | | | | | | | | | | | | | |
| 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 |
| OEM2KEY[47:32] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **OEM2KEY[63:32]**: Bits 63 to 32 of the OEM2 key

All bits are write-only and always read as zero.

3.7.17 FLASH OEM2 key register 3 (FLASH_OEM2KEYR3)

Address offset: 0x0A0

Reset value: 0xFFFF FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OEM2KEY[95:80] | | | | | | | | | | | | | | | |
| 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 |
| OEM2KEY[79:64] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **OEM2KEY[95:64]**: Bits 95 to 64 of the OEM2 key

All bits are write-only and always read as zero.

3.7.18 FLASH OEM2 key register 4 (FLASH_OEM2KEYR4)

Address offset: 0x0A4

Reset value: 0xFFFF FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OEM2KEY[127:112] | | | | | | | | | | | | | | | |
| 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 |
| OEM2KEY[111:96] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:0 **OEM2KEY[127:96]**: Bits 127 to 96 of the OEM2 key

All bits are write-only and always read as zero.

3.7.19 FLASH OEM key status register (FLASH_OEMKEYSR)

Address offset: 0x0A8

Reset value: 0x00XX 00XX (where XX represent the CRC value calculated from the current OEM keys)

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | | | | | |
|------|------|------|------|------|------|------|------|-----------------|----|----|----|----|----|----|----|--|--|--|--|--|--|
| Res. | OEM2KEYCRC[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. | OEM1KEYCRC[7:0] | | | | | | | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r | | | | | | |

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

Bits 23:16 **OEM2KEYCRC[7:0]**: 8-bit CRC value calculated from the OEM2 key

This bitfield contains the read-only 8-bit CRC value calculated from the OEM2 128-bit key.

This value is calculated using the following CRC parameters:

Polynomial = 0x7

Initial value = 0x0 (128 bits)

Final XoR = 0x5

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

Bits 7:0 **OEM1KEYCRC[7:0]**: 8-bit CRC value calculated from the OEM1 key

This bitfield contains the read-only 8-bit CRC value calculated from the OEM1 128-bit key.

This value is calculated using the following CRC parameters:

Polynomial = 0x7

Initial value = 0x0 (128 bits)

Final XoR = 0x5

3.7.20 FLASH HDP control register (FLASH_HDPCR)

Address offset: 0x0AC

Reset value: 0x00A3 00A3

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | | | | | |
|------|------|------|------|------|------|------|------|---------------------|----|----|----|----|----|----|----|--|--|--|--|--|--|
| Res. | HDP1EXT_ACCDIS[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. | HDP1_ACCDIS[7:0] | | | | | | | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | | | | | | |

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

Bits 23:16 **HDP1EXT_ACCDIS[7:0]**: HDP1 extension area access disable

The access to this register depends on its current value. Refer to [Section 3.5.4: Securable memory area extension \(HDP extension\)](#) for the full conditions.

0xA3: Access to HDP1 extension area is granted

0x5C: Access to HDP1 extension area is denied, but incrementing HDP1_EXT[6:0] of FLASH_HDPEXTR register is allowed at any time (modifying HDP1EN[7:0]/HDP1_PEND[7:0] option bytes is not allowed)

Others: Access to HDP1 extension area is denied, and updating the size of HDP1 extension area is not possible since writing to HDP1_EXT[6:0] is denied (modifying HDP1EN[7:0]/HDP1_PEND[7:0] option bytes is not allowed).

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

Bits 7:0 **HDP1_ACCDIS[7:0]**: HDP1 area access disable

When set to a value different from 0xA3, this bitfield cannot be modified except by a system reset. In this case, any write access to HDP1_ACCDIS[7:0] is ignored.

0xA3: Access to HDP1 area granted

Others: Access to HDP1 area denied (modifying HDP1EN[7:0]/HDP1_PEND[7:0] option bytes is not allowed).

3.7.21 FLASH HDP extension register (FLASH_HDPEXTR)

Address offset: 0x0B0

Reset value: 0x0000 0000

Access: no wait state when no flash memory operation is ongoing; 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. | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| Res. | HDP1_EXT[6:0] | | | | | | | | | | | | | |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw | | | | | | | |

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

Bits 6:0 **HDP1_EXT[6:0]**: HDP1 extension area size

Size of the HDP1 extension area expressed in number of 2-Kbyte pages. The extension starts at page offset HDP1_PEND[7:0] + 1.

Register access depends on the value of HDP1EXT_ACCDIS[7:0] in the FLASH_HDPCR register.

3.7.22 FLASH register map

Table 19. FLASH register map and reset values

Table 19. FLASH 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 | |
|--------|-----------------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|---------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------------------|-----------------|---|---|--|
| 0x080 | FLASH_SECR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | HDP1EN[7:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 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. | | | |
| 0x088 | FLASH_OEM1KEYR1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 0x08C | FLASH_OEM1KEYR2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 0x090 | FLASH_OEM1KEYR3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 0x094 | FLASH_OEM1KEYR4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 0x098 | FLASH_OEM2KEYR1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 0x09C | FLASH_OEM2KEYR2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 0x0A0 | FLASH_OEM2KEYR3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 0x0A4 | FLASH_OEM2KEYR4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 0x0A8 | FLASH_KEYSR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OEM2KEYCRC[7:0] | Res. | OEM1KEYCRC[7:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | |
| 0x0AC | FLASH_HDPCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | HDP1EXT_ACCDIS[7:0] | Res. | HDP1_ACCDIS[7:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | |
| 0x0B0 | FLASH_HDPEXTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | HDP1_EXT[6:0] | Res. | HDP1_EXT[6:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

4 Power control (PWR)

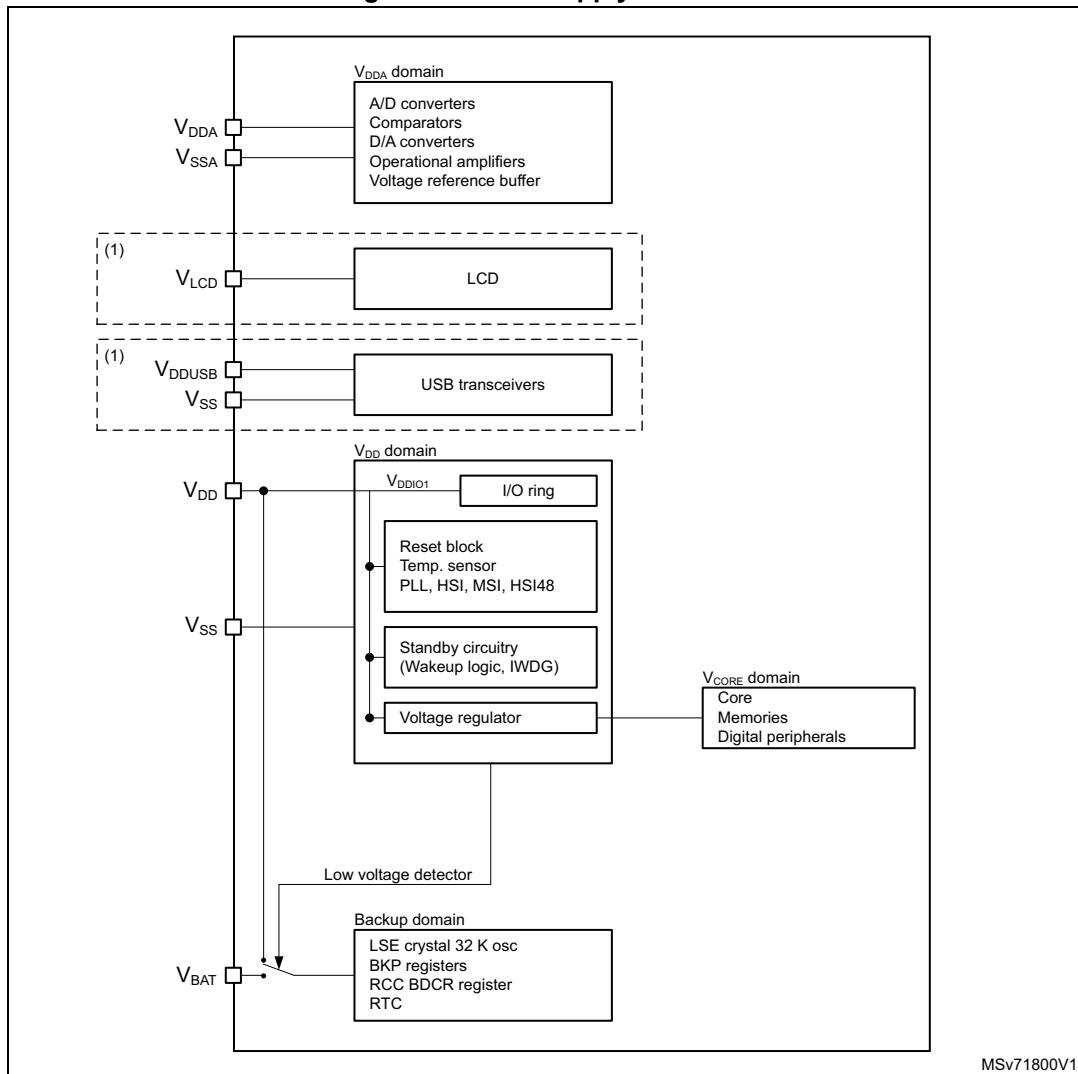
4.1 Power supplies

The STM32U0 series devices require a 1.71 V to 3.6 V operating supply voltage (V_{DD}). Several peripherals are supplied through independent power domains: V_{DDA} , V_{DDUSB} , V_{LCD} . Those supplies must not be provided without a valid operating supply on the V_{DD} pin.

- $V_{DD} = 1.71$ V to 3.6 V
 V_{DD} is the external power supply for the I/Os, the internal regulator and the system analog such as reset, power management and internal clocks. It is provided externally through VDD pins.
- $V_{DDA} = 1.62$ V (ADCs/COMPs) / 1.86 V (DACs/OPAMPS) / 2.4 V (VREFBUF) to 3.6 V.
 V_{DDA} is the external analog power supply for A/D converters, D/A converters, voltage reference buffer, operational amplifiers and comparators. The V_{DDA} voltage level is independent from the V_{DD} voltage. V_{DDA} should be preferably connected to V_{DD} when these peripherals are not used.
- $V_{DDUSB} = 3.0$ V to 3.6 V (available on STM32U0x3xx devices only)
 V_{DDUSB} is the external independent power supply for USB transceivers. The V_{DDUSB} voltage level is independent from the V_{DD} voltage. V_{DDUSB} should be preferably connected to V_{DD} when the USB is not used.
- $V_{LCD} = 2.5$ V to 3.6 V (available on STM32U0x3xx devices only)
The LCD controller can be powered either externally through VLCD pin, or internally from an internal voltage generated by the embedded step-up converter. VLCD is multiplexed with PC3 which can be used as GPIO when the LCD is not used.
- $V_{BAT} = 1.55$ 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.
 V_{BAT} is internally bonded to V_{DD} for small packages without dedicated pin.
- V_{REF-} , V_{REF+}
 V_{REF+} is the input reference voltage for ADCs and DACs. It is also the output of the internal voltage reference buffer when enabled.
When $V_{DDA} < 2$ V, V_{REF+} must be equal to V_{DDA} .
When $V_{DDA} \geq 2$ V, V_{REF+} must be between 2 V and V_{DDA} .
 V_{REF+} can be grounded when ADC and DAC are not active.
The internal voltage reference buffer supports two output voltages, which are configured with VRS bit in the VREFBUF_CSR register:
 - V_{REF+} around 2.048 V. This requires V_{DDA} equal to or higher than 2.4 V.
 - V_{REF+} around 2.5 V. This requires V_{DDA} equal to or higher than 2.8 V. V_{REF-} and V_{REF+} pins are not available on all packages. When not available on the package, they are bonded to VSSA and V_{DDA} , respectively.
When the V_{REF+} is double-bonded with V_{DDA} in a package, the internal voltage reference buffer is not available and must be kept disabled (refer to related device datasheet for packages pinout description).
 V_{REF-} must always be equal to V_{SSA} .

An embedded linear voltage regulator is used to supply the internal digital power V_{CORE} . V_{CORE} is the power supply for digital peripherals and memories.

Figure 6. Power supply overview



MSv71800V1

- Available on STM32U0x3xx devices only.

4.1.1 Independent analog peripherals supply

To improve ADC and DAC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply which 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, D/A converter, comparators, operational amplifier, voltage reference buffer).

The V_{DDA} supply can be monitored by the Peripheral Voltage Monitoring, and compared with two thresholds (1.65 V for PVM3 or 1.86 V for PVM4), refer to [Section 4.2.3: Peripheral Voltage Monitoring \(PVM\)](#) for more details.

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 and DAC reference voltage

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect to V_{REF+} a separate reference voltage lower than V_{DDA} . V_{REF+} is the highest voltage, represented by the full scale value, for an analog input (ADC) or output (DAC) signal.

V_{REF+} can be provided either by an external reference or by an internal buffered voltage reference (VREFBUF), except for packages on which VREF+ is bonded with VDDA.

The internal voltage reference is enabled by setting the ENVR bit in the [Section 16.3.1: VREFBUF control and status register \(VREFBUF_CSR\)](#). The voltage reference is set to 2.5 V when the VRS bit is set and to 2.048 V when the VRS bit is cleared. The internal voltage reference can also provide the voltage to external components through V_{REF+} pin. Refer to the device datasheet and to [Section 16: Voltage reference buffer \(VREFBUF\)](#) for further information.

4.1.2 Independent USB transceivers supply

This section is applicable to STM32U0x3xx devices only.

The USB transceivers are supplied from a separate V_{DDUSB} power supply pin. V_{DDUSB} range is from 3.0 V to 3.6 V and is completely independent from V_{DD} or V_{DDA} .

After reset, the USB features supplied by V_{DDUSB} are logically and electrically isolated and therefore are not available. The isolation must be removed before using the USB FS peripheral, by setting the USV bit in the PWR_CR2 register, once the V_{DDUSB} supply is present.

The V_{DDUSB} supply is monitored by the Peripheral Voltage Monitoring (PVM1) and compared with the internal reference voltage (V_{REFINT} , around 1.2 V), refer to [Section 4.2.3: Peripheral Voltage Monitoring \(PVM\)](#) for more details.

4.1.3 Independent LCD supply

This section is applicable to STM32U03xx devices only.

The VLCD pin is provided to control the contrast of the glass LCD. This pin can be used in two ways:

- It can receive from an external circuitry the desired maximum voltage that is provided on segment and common lines to the glass LCD by the microcontroller.
- It can also be used to connect an external capacitor that is used by the microcontroller for its voltage step-up converter. This step-up converter is controlled by software to provide the desired voltage to segment and common lines of the glass LCD.

The voltage provided to segment and common lines defines the contrast of the glass LCD pixels. This contrast can be reduced when you configure the dead time between frames.

- When an external power supply is provided to the VLCD pin, it should range from 2.5 V to 3.6 V. It does not depend on V_{DD} .
- When the LCD is based on the internal step-up converter, the VLCD pin should be

connected to a capacitor (see the product datasheet for further information).

4.1.4 Battery backup domain

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

VBAT pin is not available on low pin-count packages, V_{BAT} is internally connected to V_{DD}.

The VBAT pin powers the RTC unit, the LSE oscillator and the PC13 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 the value of $t_{RSTTEMPO}$) 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 strongly 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} externally to V_{DD} with a 100 nF external ceramic decoupling capacitor.

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

- PC13, PC14 and PC15, which can be used as GPIO pins
- PC13, PC14 and PC15, which can be configured by RTC or LSE (refer to [Section 30.3: RTC functional description on page 900](#))
- PA0/RTC_TAMP2 and PE6/RTC_TAMP3 when they are configured by the RTC as tamper pins

Note: Due to the fact that the analog switch can transfer only a limited amount of current (3 mA), the use of GPIO PC13 to PC15 in output mode is restricted: the speed has to be limited to 2 MHz with a maximum load of 30 pF and these I/Os must not be used as a current source (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:

- PC13, PC14 and PC15 can be controlled only by RTC or LSE (refer to [Section 30.3: RTC functional description on page 900](#))
- PA0/RTC_TAMP2 and PE6/RTC_TAMP3 when they are configured by the RTC as tamper pins

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. Enable the power interface clock by setting the PWREN bits in the [Section 5.2.18: Peripheral clock enable registers](#)
2. Set the DBP bit in the [Power control register 1 \(PWR_CR1\)](#) to enable access to the backup domain
3. Select the RTC clock source in the [RTC domain control register \(RCC_BDCR\)](#).
4. Enable the RTC clock by setting the RTCEN [15] bit in the [RTC domain control register \(RCC_BDCR\)](#).

VBAT battery charging

When VDD 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 kOhm resistor or through a 1.5 kOhm resistor depending on the VBRS bit value in the PWR_CR4 register.

The battery charging is enabled by setting VBE bit in the PWR_CR4 register. It is automatically disabled in VBAT mode.

4.1.5 Voltage regulator

Two embedded linear voltage regulators supply all the digital circuitries, except for the Standby circuitry and the backup domain. The main regulator output voltage (V_{CORE}) can be programmed by software to two different power ranges (Range 1 and Range 2) in order to optimize the consumption depending on the system's maximum operating frequency (refer to [Section 5.2.9: Clock source frequency versus voltage scaling](#) and to [Section 3.3.4: FLASH read access latency](#)).

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 Stop 0 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, SRAM1 and SRAM2.
- In Stop 1 and Stop 2 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, SRAM1 and SRAM2.
- In Standby mode with SRAM2 content preserved (RRS bit is set in the PWR_CR3 register), the main regulator (MR) is off and the low-power regulator (LPR) provides the supply to SRAM2 only. The core, digital peripherals (except Standby circuitry and backup domain) and SRAM1 are powered off.
- In Standby mode, both regulators are powered off. The contents of the registers, 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,

SRAM1 and SRAM2 is lost, except for the backup domain.

4.1.6 Dynamic voltage scaling management

The dynamic voltage scaling is a power management technique which consists in increasing or decreasing the voltage used for the digital peripherals (V_{CORE}), according to the application performance and power consumption needs.

Dynamic voltage scaling to increase V_{CORE} is known as over-volting. It allows to improve the device performance.

Dynamic voltage scaling to decrease V_{CORE} is known as under-volting. It is performed to save power, particularly in laptop and other mobile devices where the energy comes from a battery and is thus limited.

- Range 1: High-performance range.

The main regulator provides a typical output voltage at 1.2 V. The system clock frequency can be up to 56 MHz. The flash access time for read access is minimum, write and erase operations are possible.

- Range 2: Low-power range.

The main regulator provides a typical output voltage at 1.0 V. The system clock frequency can be up to 16 MHz when running from MSI, or up to 18 MHz when running from PLL or HSE. The flash access time for a read access is increased as compared to Range 1; write and erase operations are possible.

Voltage scaling is selected through the VOS bit in the PWR_CR1 register.

The sequence to go from Range 1 to Range 2 is:

1. Reduce the system frequency to a value lower than the maximum value allowed for Range 2.
2. Adjust the number of wait states according to the new frequency target in Range 2 (LATENCY bits in the FLASH_ACR).
3. Program the VOS bits to "10" in the PWR_CR1 register.

The sequence to go from Range 2 to Range 1 is:

1. Program the VOS bits to "01" in the PWR_CR1 register.
2. Wait until the VOSF flag is cleared in the PWR_SR2 register.
3. Adjust number of wait states according new frequency target in Range 1 (LATENCY bits in the FLASH_ACR).
4. Increase the system frequency.

4.2 Power supply supervisor

4.2.1 Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR)

The device has an integrated power-on reset (POR) / power-down reset (PDR), coupled with a brown-out reset (BOR) circuitry. The BOR is active in all power modes except Shutdown mode, and cannot be disabled.

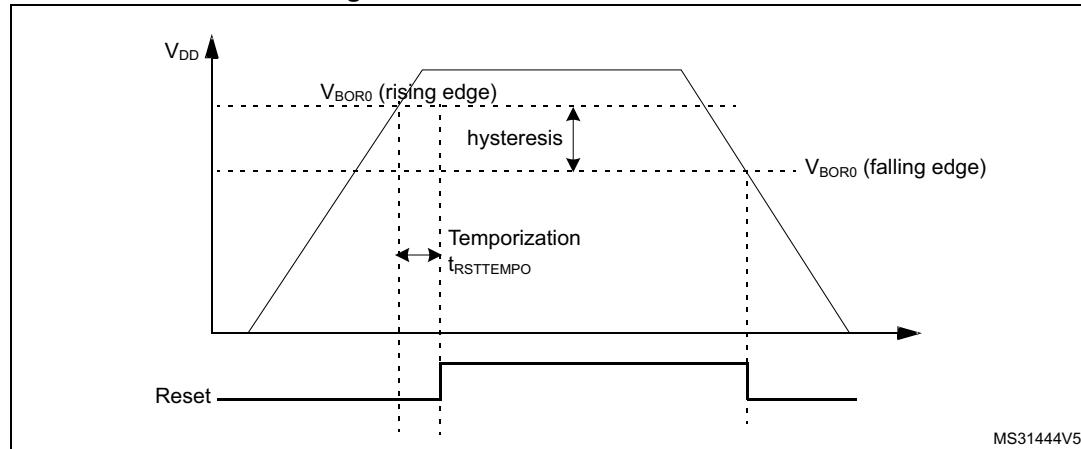
Five BOR thresholds can be selected through option bytes.

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.

On STM32U0 devices, continuous monitoring of the power supply might be changed to periodical sampling to reduce power consumption in Stop 2 and Standby modes by setting ENULP bit in [Power control register 3 \(PWR_CR3\)](#). When sampling mode is selected, fast supply drop between two samples is not detected.

Figure 7. Brown-out reset waveform



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

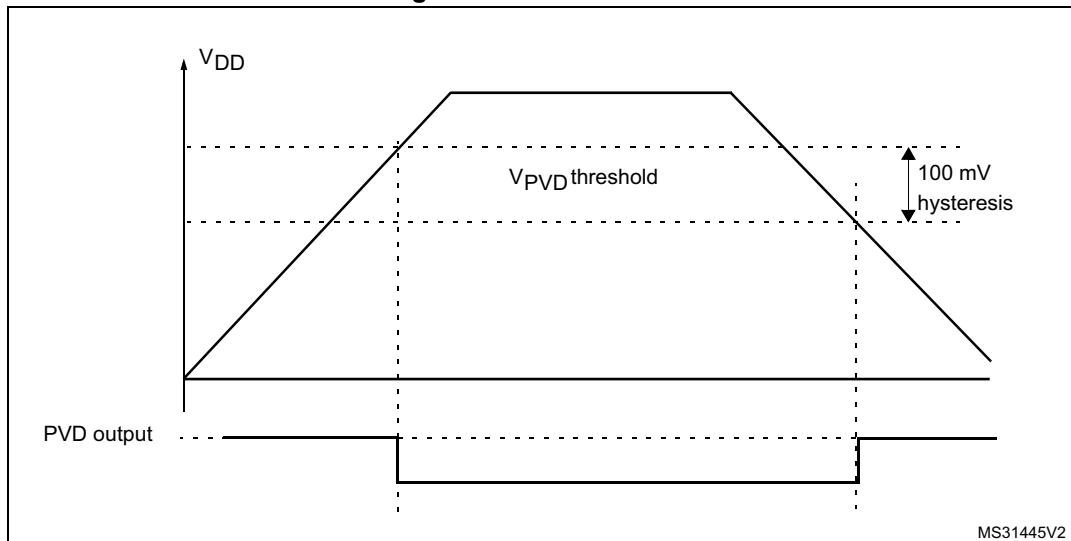
4.2.2 Programmable voltage detector (PVD)

You can use the PVD to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the [Power control register 2 \(PWR_CR2\)](#).

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the [Power status register 2 \(PWR_SR2\)](#), to indicate if V_{DD} 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 rising/falling edge sensitivity of the EXTI Line16 should be configured according to PVD output behavior i.e. if the EXTI line 16 is configured to rising edge sensitivity, the interrupt is generated when V_{DD} drops below the PVD threshold. As an example the service routine could perform emergency shutdown tasks.

Figure 8. PVD thresholds



4.2.3 Peripheral Voltage Monitoring (PVM)

Only V_{DD} is monitored by default, as it is the only supply required for all system-related functions. The other supplies (V_{DDA} and V_{DDUSB}) can be independent from V_{DD} and can be monitored with four Peripheral Voltage Monitoring (PVM).

Each of the three PVM_x ($x=1, 3, 4$) is a comparator between a fixed threshold V_{PVM_x} and the selected power supply. $PVMO_x$ flags indicate if the independent power supply is higher or lower than the PVM_x threshold: $PVMO_x$ flag is cleared when the supply voltage is above the PVM_x threshold, and is set when the supply voltage is below the PVM_x threshold.

Each PVM output is connected to an EXTI line and can generate an interrupt if enabled through the EXTI registers. The PVM_x output interrupt is generated when the independent power supply drops below the PVM_x threshold and/or when it rises above the PVM_x threshold, depending on EXTI line rising/falling edge configuration.

Each PVM can remain active in Stop 0, Stop 1 and Stop 2 modes, and the PVM interrupt can wake up from the Stop mode.

Table 20. PVM features

| PVM | Power supply | PVM threshold | EXTI line |
|---------------------|--------------|----------------------------|-----------|
| PVM1 ⁽¹⁾ | V_{DDUSB} | V_{PVM1} (around 1.2 V) | 19 |
| PVM3 | V_{DDA} | V_{PVM3} (around 1.65 V) | 20 |
| PVM4 | V_{DDA} | V_{PVM4} (around 1.86 V) | 21 |

1. Available on STM32 U0x3xx devices only.

The independent supplies (V_{DDA} and V_{DDUSB}) are not considered as present by default, and a logical and electrical isolation is applied to ignore any information coming from the peripherals supplied by these dedicated supplies.

- If these supplies are shorted externally to V_{DD} , the application should assume they are available without enabling any Peripheral Voltage Monitoring.
- If these supplies are independent from V_{DD} , the Peripheral Voltage Monitoring (PVM)

can be enabled to confirm whether the supply is present or not.

The following sequence must be done before using the USB FS peripheral on STM32U0x3xx devices:

1. If V_{DDUSB} is independent from V_{DD} :
 - a) Enable the PVM1 by setting PVME1 bit in the *Power control register 2 (PWR_CR2)*.
 - b) Wait for the PVM1 wake-up time
 - c) Wait until PVMO1 bit is cleared in the *Power status register 2 (PWR_SR2)*.
 - d) Optional: Disable the PVM1 for consumption saving.
2. Set the USV bit in the *Power control register 2 (PWR_CR2)* to remove the V_{DDUSB} power isolation.

The following sequence must be done before using any of these analog peripherals: analog to digital converters, digital to analog converters, comparators, operational amplifiers, voltage reference buffer:

1. If V_{DDA} is independent from V_{DD} :
 - a) Enable the PVM3 (or PVM4) by setting PVME3 (or PVME4) bit in the *Power control register 2 (PWR_CR2)*.
 - b) Wait for the PVM3 (or PVM4) wake-up time
 - c) Wait until PVMO3 (or PVMO4) bit is cleared in the *Power status register 2 (PWR_SR2)*.
 - d) Optional: Disable the PVM3 (or PVM4) for consumption saving.
2. Enable the analog peripheral, which automatically removes the V_{DDA} isolation.

4.3 Low-power modes

By default, the microcontroller is in Run mode after a system or a power Reset. Several low-power modes are available to save power when the CPU does not need to be kept running, for example when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time and available wake-up sources.

The device features seven low-power modes:

- Sleep mode: CPU clock off, all peripherals including Cortex®-M0+ core peripherals such as NVIC, SysTick, etc. can run and wake up the CPU when an interrupt or an event occurs. Refer to [Section 4.3.4: Sleep mode](#).
- Low-power run mode: This mode is achieved when the system clock frequency is reduced below 2 MHz. The code is executed from the SRAM or the flash memory. The regulator is in low-power mode to minimize the regulator's operating current. Refer to [Section 4.3.2: Low-power run mode \(LP run\)](#).
- Low-power sleep mode: This mode is entered from the Low-power run mode: Cortex®-M0+ is off. Refer to [Section 4.3.5: Low-power sleep mode \(LP sleep\)](#).
- Stop 0, Stop 1 and Stop 2 modes: SRAM1, SRAM2 and all registers content are 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 the

Stop mode to detect their wake-up condition.

In Stop 2 mode, most of the V_{CORE} domain is put in a lower leakage mode.

Stop 1 offers the largest number of active peripherals and wake-up sources, a smaller wake-up time but a higher consumption than Stop 2. In Stop 0 mode, the main regulator remains ON, which allows the fastest wake-up time but with much higher consumption. The active peripherals and wake-up sources are the same as in Stop 1 mode.

The system clock, when exiting from Stop 0, Stop 1 or Stop 2 mode, can be either MSI up to 48 MHz or HSI16, depending on the software configuration.

Refer to [Section 4.3.6: Stop 0 mode](#) and [Section 4.3.8: Stop 2 mode](#).

- Standby mode: V_{CORE} domain is powered off. However, it is possible to preserve the SRAM2 contents:
 - Standby mode with SRAM2 retention when the bit RRS is set in PWR_CR3 register. In this case, SRAM2 is supplied by the low-power regulator.
 - Standby mode when the bit RRS is cleared in PWR_CR3 register. 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 MSI from 1 MHz up to 8 MHz.

Refer to [Section 4.3.9: Standby mode](#).

- 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 the 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. Refer to [Section 4.3.10: Shutdown mode](#).

In addition, the power consumption in Run mode can be reduced by one of the following means:

- Slowing down the system clocks
- Gating the clocks to the APB and AHB peripherals when they are unused.

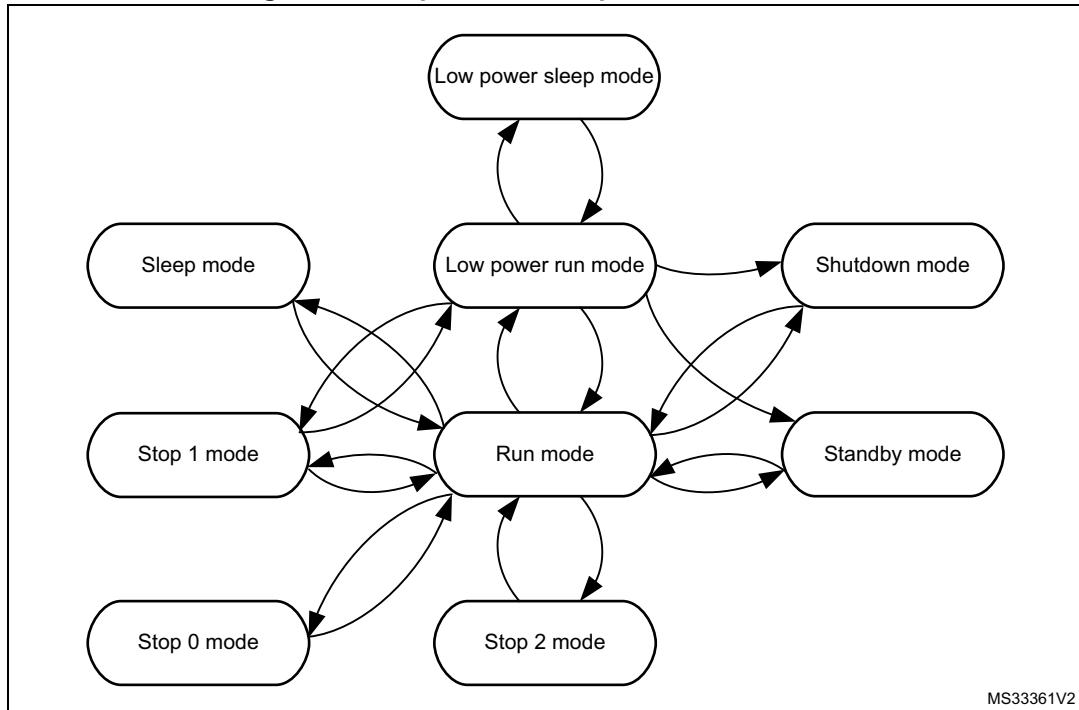
Figure 9. Low-power modes possible transitions

Table 21. Low-power mode summary

| Mode name | Entry | Wake-up source ⁽¹⁾ | Wake-up system clock | Effect on clocks | Voltage regulators | |
|---------------------------------------|--|---|--|---|--------------------|-----|
| | | | | | 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 no effect on other clocks or analog clock sources | ON | ON |
| | WFE | Wake-up event | | | | |
| Low-power run | Set LPR bit | Clear LPR bit | Same as Low-power run clock | None | OFF | ON |
| Low-power sleep | Set LPR bit + WFI or Return from ISR | Any interrupt | Same as before entering Low-power sleep mode | CPU clock OFF no effect on other clocks or analog clock sources | OFF | ON |
| | Set LPR bit + WFE | Wake-up event | | | OFF | ON |
| Stop 0 | LPMS="000" + SLEEPDEEP bit + WFI or Return from ISR or WFE | Any EXTI line (configured in the EXTI registers) Specific peripherals events | HSI16 when STOPWUCK=1 in RCC_CFGR MSI with the frequency before entering the Stop mode when STOPWUCK=0. | All clocks OFF except LSI and LSE | ON | ON |
| Stop 1 | LPMS="001" + SLEEPDEEP bit + WFI or Return from ISR or WFE | | | | | |
| Stop 2 | LPMS="010" + SLEEPDEEP bit + WFI or Return from ISR or WFE | | | | OFF | |
| Standby with SRAM2 | LPMS="011" + Set RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE | WKUP pin edge, RTC event, external reset in NRST pin, IWDG reset | MSI from 1 MHz up to 8 MHz | All clocks OFF except LSI and LSE | | |
| Standby | LPMS="011" + Clear RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE | WKUP pin edge, RTC event, external reset in NRST pin, IWDG reset | | | OFF | 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 22: Functionalities depending on the working mode](#).

Table 22. Functionalities depending on the working mode

Legend: Y = Yes (Enable). O = Optional (Disable by default. Can be enabled by software). - = Not available

| Peripheral | Run | Sleep | Low-power run | Low-power sleep | Stop 0/1 | | Stop 2 | | Standby | | Shutdown | | VBAT |
|--|------------------|------------------|------------------|------------------|------------------|--------------------|------------------|--------------------|------------------|--------------------|----------|--------------------|------|
| | | | | | - | Wake-up capability | - | Wake-up capability | - | Wake-up capability | - | Wake-up capability | |
| CPU | Y | - | Y | - | - | - | - | - | - | - | - | - | - |
| Flash memory | O ⁽¹⁾ | O ⁽¹⁾ | O ⁽¹⁾ | O ⁽¹⁾ | - | - | - | - | - | - | - | - | - |
| SRAM1 | Y | Y ⁽²⁾ | Y | Y ⁽²⁾ | Y | - | Y | - | - | - | - | - | - |
| SRAM2 | Y | Y ⁽²⁾ | Y | Y ⁽²⁾ | Y | - | Y | - | O ⁽³⁾ | - | - | - | - |
| Backup Registers | Y | Y | Y | Y | Y | - | Y | - | Y | - | Y | - | Y |
| Brown-out reset (BOR) | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | - | - | - |
| Programmable Voltage Detector (PVD) | O | O | O | O | O | O | O | O | - | - | - | - | - |
| Peripheral Voltage Monitor (PVMx; x=1,3,4) | O | O | O | O | O | O | O | O | - | - | - | - | - |
| DMA | O | O | O | O | - | - | - | - | - | - | - | - | - |
| Oscillator HSI16 | O | O | O | O | (4) | - | (4) | - | - | - | - | - | - |
| Oscillator HSI48 | O | O | - | - | - | - | - | - | - | - | - | - | - |
| High Speed External (HSE) | O | O | O | O | - | - | - | - | - | - | - | - | - |
| Low Speed Internal (LSI) | O | O | O | O | O | - | O | - | O | - | - | - | - |
| Low Speed External (LSE) | 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 | - | - | - |
| RTC / Auto wake-up | O | O | O | O | O | O | O | O | O | O | O | O | O |
| Number of RTC Tamper pins | 2 | 2 | 2 | 2 | 2 | O | 2 | O | 2 | O | 2 | O | 2 |
| USARTx (x=1,2,3,4) | O | O | O | O | O ⁽⁵⁾ | O ⁽⁵⁾ | - | - | - | - | - | - | - |
| LUARTx (x = 1, 2, 3) | O | O | O | O | O ⁽⁵⁾ | O ⁽⁵⁾ | O ⁽⁵⁾ | O ⁽⁵⁾ | - | - | - | - | - |
| I2Cx (x=2,4) | O | O | O | O | O ⁽⁶⁾ | O ⁽⁶⁾ | - | - | - | - | - | - | - |

Table 22. Functionalities depending on the working mode (continued)

Legend: Y = Yes (Enable). O = Optional (Disable by default. Can be enabled by software). - = Not available

| Peripheral | Run | Sleep | Low-power run | Low-power sleep | Stop 0/1 | | Stop 2 | | Standby | | Shutdown | | VBAT |
|--------------------------------|------------------|------------------|---------------|-----------------|----------|--------------------|----------|--------------------|----------------------|--------------------------------|----------|--------------------|------|
| | | | | | - | Wake-up capability | - | Wake-up capability | - | Wake-up capability | - | Wake-up capability | |
| I2Cx (x=1,3) | O | O | O | O | O (6) | O (6) | O (6) | O (6) | - | - | - | - | - |
| SPIx (x=1,2,3) | O | O | O | O | - | - | - | - | - | - | - | - | - |
| ADC1 | O | O | O | O | - | - | - | - | - | - | - | - | - |
| DAC1 | O | O | O | O | O | - | - | - | - | - | - | - | - |
| OPAMP1 | O | O | O | O | O | - | - | - | - | - | - | - | - |
| COMPx (x=1,2) | O | O | O | O | O | O | O | O | - | - | - | - | - |
| Temperature sensor | O | O | O | O | - | - | - | - | - | - | - | - | - |
| Timers (TIMx) | O | O | O | O | - | - | - | - | - | - | - | - | - |
| LPTIMx (x = 1, 2, 3) | O | O | O | O | O | O | O | O | - | - | - | - | - |
| Independent watchdog (IWDG) | O | O | O | O | O | O | O | O | O | O | - | - | - |
| Window watchdog (WWDG) | O | O | O | O | - | - | - | - | - | - | - | - | - |
| SysTick timer | O | O | O | O | - | - | - | - | - | - | - | - | - |
| Touch sensing controller (TSC) | O | O | O | O | - | - | - | - | - | - | - | - | - |
| Random number generator (RNG) | O ⁽⁷⁾ | O ⁽⁷⁾ | - | - | - | - | - | - | - | - | - | - | - |
| AES hardware accelerator | O | O | O | O | - | - | - | - | - | - | - | - | - |
| CRC calculation unit | O | O | O | O | - | - | - | - | - | - | - | - | - |
| GPIOs | O | O | O | O | O | O | O | O | (8) 5 pins (9) | (10) up to 5 pins (9) | - | - | - |

1. The flash memory can be configured in power-down mode. By default, it is not in power-down mode.
2. The SRAM clock can be gated on or off.
3. SRAM2 content is preserved when the bit RRS is set in PWR_CR3 register.
4. 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 which requested it. HSI16 is automatically put off when the peripheral does not need it anymore.
5. UART and LPUART reception is functional in Stop mode, and generates a wake-up interrupt on Start, address match or received frame event.
6. I2C address detection is functional in Stop mode, and generates a wake-up interrupt in case of address match.

7. Voltage scaling Range 1 only.
8. I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
9. I/Os with wake-up from Standby/Shutdown capability are the following: PA0, PC13, PE6, PA2, PC5.
10. I/Os can be configured with internal pull-up, pull-down or floating in Shutdown mode but the configuration is lost when exiting the Shutdown mode.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop 0, Stop 1, Stop 2, Standby or Shutdown mode while the debug features are used. This is due to the fact that the Cortex®-M0+ 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 37.2.6: Debug in low-power modes](#).

4.3.1 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 the Sleep mode.

For more details, refer to [Section 5.4.3: 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 the 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.

4.3.2 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 should not exceed 2 MHz.

Please 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 the 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 by setting the RUN_PD bit in the *FLASH access control register (FLASH_ACR)*.
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_CR1 register.

Refer to [Table 23: Low-power run](#) on how to enter the Low-power run mode.

Exiting the Low-power run mode

To exit the Low-power run mode, proceed as follows:

1. Force the regulator in main mode by clearing the LPR bit in the PWR_CR1 register.
2. Wait until REGLPF bit is cleared in the PWR_SR2 register.
3. Increase the system clock frequency.

Refer to [Table 23: Low-power run](#) on how to exit the Low-power run mode.

Table 23. Low-power run

| Low-power run mode | Description |
|--------------------|---|
| Mode entry | Decrease the system clock frequency below 2 MHz LPR = 1 |
| Mode exit | LPR = 0 Wait until REGLPF = 0 Increase the system clock frequency |
| Wake-up latency | Regulator wake-up time from low-power mode |

4.3.3 Low-power modes

Entering low-power mode

Low-power modes are entered by the MCU by executing the WFI (Wait For Interrupt), or WFE (Wait for Event) instructions, or when the SLEEPONEXIT bit in the Cortex®-M0+ 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.

Exiting low-power mode

From Sleep modes, and Stop modes the MCU exit low-power mode depending on the way the low-power 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:
 - NVIC IRQ interrupt.
 - When SEVONPEND = 0 in the Cortex®-M0+ System Control register. By 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 wakes up and interrupt the MCU.

- When SEVONPEND = 1 in the Cortex®-M0+ System Control register.

By 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.

- Event

Configuring a 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 modes, and Shutdown modes the MCU exit low-power mode through an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC event occurs (see [Figure 297: RTC block diagrams](#)).

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.).

4.3.4 Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

Entering the Sleep mode

The Sleep mode is entered according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is clear.

Refer to [Table 24: Sleep](#) for details on how to enter the Sleep mode.

Exiting the Sleep mode

The Sleep mode is exit according [Section : Exiting low-power mode](#).

Refer to [Table 24: Sleep](#) for more details on how to exit the Sleep mode.

Table 24. Sleep

| Sleep-now mode | Description |
|-----------------|--|
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) is pending Refer to the Cortex®-M0+ System Control register. |
| | On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex®-M0+ System Control register. |
| Mode exit | If WFI or return from ISR was used for entry Interrupt: refer to Table 54: Vector table If WFE was used for entry and SEVONPEND = 0: Wake-up event: refer to Section 11: Nested vectored interrupt controller (NVIC) If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to Table 54: Vector table or Wake-up event: refer to Section 11: Nested vectored interrupt controller (NVIC) |
| Wake-up latency | None |

4.3.5 Low-power sleep mode (LP sleep)

Please 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 the Low-power sleep mode

The Low-power sleep mode is entered from Low-power run mode according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is clear.

Refer to [Table 25: Low-power sleep](#) for details on how to enter the Low-power sleep mode.

Exiting the Low-power sleep mode

The low-power Sleep mode is exit according [Section : Exiting low-power mode](#). When exiting the Low-power sleep mode by issuing an interrupt or an event, the MCU is in Low-power run mode.

Refer to [Table 25: Low-power sleep](#) for details on how to exit the Low-power sleep mode.

Table 25. Low-power sleep

| Low-power sleep-now mode | Description |
|--------------------------|--|
| Mode entry | Low-power sleep mode is entered from the Low-power run mode. WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) is pending Refer to the Cortex®-M0+ System Control register. |
| | Low-power sleep mode is entered from the Low-power run mode. On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex®-M0+ System Control register. |
| Mode exit | If WFI or Return from ISR was used for entry Interrupt: refer to Table 54: Vector table If WFE was used for entry and SEVONPEND = 0: Wake-up event: refer to Section 11: Nested vectored interrupt controller (NVIC) If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to Table 54: Vector table Wake-up event: refer to Section 11: Nested vectored interrupt controller (NVIC) After exiting the Low-power sleep mode, the MCU is in Low-power run mode. |
| Wake-up latency | None |

4.3.6 Stop 0 mode

The Stop 0 mode is based on the Cortex®-M0+ deepsleep mode combined with the peripheral clock gating. The voltage regulator is configured in main regulator mode. In Stop 0 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 the wake-up capability (I2Cx (x=1,2,3), U(S)ARTx(x=1,2,3) and LPUART) 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 Stop 0 mode. The consumption is increased when thresholds higher than V_{BOR0} are used.

I/O states in Stop 0 mode

In the Stop 0 mode, all I/O pins keep the same state as in the Run mode.

Entering the Stop 0 mode

The Stop 0 mode is entered according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is set.

Refer to [Table 26: Stop 0 mode](#) for details on how to enter the Stop 0 mode.

If flash memory programming is ongoing, the Stop 0 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop 0 mode entry is delayed until the APB access is finished.

In Stop 0 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 28.4: IWDG functional description](#).
- Real-time clock (RTC): this is configured by the RTCEN bit in the [RTC domain control register \(RCC_BDCR\)](#)
- Internal RC oscillator (LSI): this is configured by the LSION bit in the [Control/status register \(RCC_CSR\)](#)
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the [RTC domain control register \(RCC_BDCR\)](#).

Several peripherals can be used in Stop 0 mode and can add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock: LCD, LPTIM1, LPTIM2, I2Cx (x=1,2,3,4) U(S)ARTx(x=1,2,3,4), LPUART.

The DACx (x=1,2), the OPAMP and the comparators can be used in Stop 0 mode, the PVMx (x=1,3,4) and the PVD as well. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx (x=1), temperature sensor and VREFBUF buffer can consume power during the Stop 0 mode, unless they are disabled before entering this mode.

Exiting the Stop 0 mode

The Stop 0 mode is exit according [Section : Entering low-power mode](#).

Refer to [Table 26: Stop 0 mode](#) for details on how to exit Stop 0 mode.

When exiting Stop 0 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 [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 the Stop 0 mode, the MCU is either in Run mode Range 1 or Run Mode Range 2 depending on VOS bit in PWR_CR1.

Table 26. Stop 0 mode

| Stop 0 mode | Description |
|-----------------|---|
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “000” in PWR_CR1 |
| | On Return from ISR while: – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = “000” in PWR_CR1 |
| | <i>Note:</i> To enter Stop 0 mode, all EXTI Line pending bits (in EXTI rising edge pending register 1 (EXTI_RPR1) and EXTI falling edge pending register 1 (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and program execution continues. |
| Mode exit | If WFI or Return from ISR was used for entry 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: Vector table If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to Section 11: Nested vectored interrupt controller (NVIC) . If WFE was used for entry and SEVONPEND = 1: 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: Vector table . Wake-up event: refer to Section 11: Nested vectored interrupt controller (NVIC) |
| Wake-up latency | Longest wake-up time between: MSI or HSI16 wake-up time and flash wake-up time from Stop 0 mode. |

4.3.7 Stop 1 mode

The Stop 1 mode is the same as Stop 0 mode except that the main regulator is OFF, and only the low-power regulator is ON. Stop 1 mode can be entered from Run mode and from Low-power run mode.

Refer to [Table 27: Stop 1 mode](#) for details on how to enter and exit Stop 1 mode.

Table 27. Stop 1 mode

| Stop 1 mode | Description |
|-----------------|---|
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “001” in PWR_CR1 |
| | On Return from ISR while: – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = “001” in PWR_CR1 |
| | <i>Note:</i> To enter Stop 1 mode, all EXTI Line pending bits (in EXTI rising edge pending register 1 (EXTI_RPR1) and EXTI falling edge pending register 1 (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 1 mode entry procedure is ignored and program execution continues. |
| Mode exit | If WFI or Return from ISR was used for entry 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: Vector table . If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to Section 11: Nested vectored interrupt controller (NVIC) . If WFE was used for entry and SEVONPEND = 1: 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: Vector table Wake-up event: refer to Section 11: Nested vectored interrupt controller (NVIC) |
| Wake-up 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 Stop 1 mode. |

4.3.8 Stop 2 mode

The Stop 2 mode is based on the Cortex®-M0+ deepsleep mode combined with peripheral clock gating. In Stop 2 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 (I2C3 and LPUART) 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 Stop 2 mode. The consumption is increased when thresholds higher than V_{BOR0} are used.

Note: The comparators outputs, the LPUART outputs and the LPTIM1 outputs are forced to low speed ($OSPEEDy=00$) during the Stop 2 mode.

I/O states in Stop 2 mode

In the Stop 2 mode, all I/O pins keep the same state as in the Run mode.

Entering Stop 2 mode

The Stop 2 mode is entered according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is set.

Refer to [Table 28: Stop 2 mode](#) for details on how to enter the Stop 2 mode.

Stop 2 mode can only be entered from Run mode. It is not possible to enter Stop 2 mode from the Low-power run mode.

If flash memory programming is ongoing, the Stop 2 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop 2 mode entry is delayed until the APB access is finished.

In Stop 2 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 28.4: IWDG functional description](#) in [Section 28: Independent watchdog \(IWDG\)](#).
- Real-time clock (RTC): this is configured by the RTCEN bit in the [Section 5.4.22: RTC domain control register \(RCC_BDCR\)](#)
- Internal RC oscillator (LSI): this is configured by the LSION bit in the [Section 5.4.23: Control/status register \(RCC_CSR\)](#).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the [RTC domain control register \(RCC_BDCR\)](#).

Several peripherals can be used in Stop 2 mode and can add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock: LCD, LPTIM1, I2C3, LPUART.

The comparators can be used in Stop 2 mode, the PVMx (x=1,3,4) and the PVD as well. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx, OPAMPx, DACx, temperature sensor and VREFBUF buffer can consume power during Stop 2 mode, unless they are disabled before entering this mode.

All the peripherals which cannot be enabled in Stop 2 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 [AHB peripheral reset register \(RCC_AHBRSTR\)](#), [APB peripheral reset register 1 \(RCC_APBRSTR1\)](#), and [APB peripheral reset register 2 \(RCC_APBRSTR2\)](#).

Exiting Stop 2 mode

The Stop 2 mode is exit according [Section : Exiting low-power mode](#).

Refer to [Table 28: Stop 2 mode](#) for details on how to exit Stop 2 mode.

When exiting Stop 2 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 [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 the Stop 2 mode, the MCU is in Run mode (Range 1 or Range 2 depending on VOS bit in PWR_CR1).

Table 28. Stop 2 mode

| Stop 2 mode | Description |
|-----------------|--|
| Mode entry | <p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “010” in PWR_CR1 |
| | <p>On return from ISR while:</p> <ul style="list-style-type: none"> – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = “010” in PWR_CR1 |
| | <p><i>Note: To enter Stop 2 mode, all EXTI Line pending bits (in EXTI rising edge pending register 1 (EXTI_RPR1) and EXTI falling edge pending register 1 (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop mode entry procedure is ignored and program execution continues.</i></p> |
| Mode exit | <p>If WFI or Return from ISR was used for entry: 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 . If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to Section 12.3.2: EXTI direct event input wake-up.</p> <p>If WFE was used for entry and SEVONPEND = 1: 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: Vector table Any EXTI Line configured in event mode. Refer to Section 12.3.2: EXTI direct event input wake-up.</p> |
| Wake-up 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 Stop 2 mode. |

4.3.9 Standby mode

The Standby mode allows to achieve the lowest power consumption with BOR. It is based on the Cortex®-M0+ 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 6](#)). SRAM2 content can be preserved if the bit RRS is set in the PWR_CR3 register. 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 the Standby mode, the I/Os are by default in floating state. If the APC bit of PWR_CR3 register has been set, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers ($x=A,B,C,D,E,F,G,H$)), or with a pull-down (refer to PWR_PDCRx registers ($x=A,B,C,D,E,F,G,H$)), or can be kept in analog state if none of the PWR_PUCRx or PWR_PDCRx register has been set. The pull-down configuration has highest priority over pull-up configuration in case both PWR_PUCRx and PWR_PDCRx are set for the same I/O.

Some I/Os (listed in [Section 7.3.1: General-purpose I/O \(GPIO\)](#)) are used for JTAG/SW debug. They can only be configured to their respective reset pull-up or pull-down state during Standby mode by setting their respective bit in the PWR_PUCRx or PWR_PDCRx registers, or are configured to floating state if the bit is kept cleared.

The RTC outputs on PC13 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. Five wake-up pins (WKUP x , $x=1,2\dots 5$) and the 3 RTC tampers are available.

Entering Standby mode

The Standby mode is entered according [Section : Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is set.

Refer to [Table 29: Standby mode](#) 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 28.4: IWDG functional description](#) in [Section 28: Independent watchdog \(IWDG\)](#).
- 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 LSION 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

The Standby mode is exited according [Section : Entering low-power mode](#). The SBF status flag in the [Power control register 3 \(PWR_CR3\)](#) indicates that the MCU was in Standby mode. All registers are reset after wake-up from Standby except for [Power control register 3 \(PWR_CR3\)](#).

Refer to [Table 29: Standby mode](#) for more details on how to exit Standby mode.

When exiting Standby mode, I/Os that were configured with pull-up or pull-down during Standby through the PWR_PUCRx or PWR_PDCRx registers keep this configuration upon exiting Standby mode until the bit APC of PWR_CR3 register has been cleared. Once the bit APC is cleared, they are either configured to their reset values or to the pull-up/pull-down state according the GPIOx_PUPDR registers. The content of the PWR_PUCRx or PWR_PDCRx registers however is not lost and can be re-used for a sub-sequent entering into Standby mode.

Some I/Os (listed in [Section 7.3.1: General-purpose I/O \(GPIO\)](#)) are used for JTAG/SW debug, and have internal pull-up or pull-down activated after reset, so are configured at this reset value as well when exiting Standby mode.

For I/Os with a pull-up or pull-down pre-defined after reset (some JTAG/SW I/Os) or with GPIOx_PUPDR programming done after exiting from Standby, when this programming is different from the PWR_PUCRx or PWR_PDCRx programmed value during Standby, both a pull-down and pull-up is applied until the bit APC is cleared, releasing the PWR_PUCRx or PWR_PDCRx programmed value.

Table 29. Standby mode

| Standby mode | Description |
|-----------------|---|
| | WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “011” in PWR_CR1 – WUFx bits are cleared in power status register 1 (PWR_SR1) |
| Mode entry | On return from ISR while: <ul style="list-style-type: none"> – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – SLEEPONEXIT = 1 – No interrupt is pending – LPMS = “011” in PWR_CR1 and – WUFx bits are cleared in power status register 1 (PWR_SR1) – 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, IWDG Reset, BOR reset |
| Wake-up latency | Reset phase |

4.3.10 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, are by default in floating state. If the APC bit of PWR_CR3 register has been set, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers ($x = A, B, C, D, E, F$), or with a pull-down (refer to PWR_PDCRx registers ($x = A, B, C, D, E, F$)), or can be kept in analog state if none of the PWR_PUCRx or PWR_PDCRx register has been set. The pull-down configuration has highest priority over pull-up configuration in case both PWR_PUCRx and PWR_PDCRx are set for the same I/O. However this configuration is lost when exiting the Shutdown mode due to the power-on reset.

Some I/Os (listed in [Section 7.3.1: General-purpose I/O \(GPIO\)](#)) are used for JTAG/SW debug. They can only be configured to their respective reset pull-up or pull-down state during Standby mode by setting their respective bit in the PWR_PUCRx or PWR_PDCRx registers, or configured to floating state if the bit is kept cleared.

The RTC outputs on PC13 are functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. Up to six wake-up pins (WKUP x , $x = 1,2,\dots,5$) and the 3 RTC tampers are available.

Entering Shutdown mode

The Shutdown mode is entered according [Entering low-power mode](#), when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is set.

Refer to [Table 30: Shutdown mode](#) 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 Backup domain control register (RCC_BDCR). Caution: in case of VDD power-down the RTC content is lost.
- external 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)

Exiting Shutdown mode

The Shutdown mode is exit according [Section : 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 30: Shutdown mode](#) for more details on how to exit Shutdown mode.

When exiting Shutdown mode, I/Os that were configured with pull-up or pull-down during Shutdown through the PWR_PUCRx or PWR_PDCRx registers lose their configuration and

are configured in floating state or to their pull-up/pull-down reset value (for some I/Os listed in [Section 7.3.1: General-purpose I/O \(GPIO\)](#)).

Table 30. Shutdown mode

| Shutdown mode | Description |
|-----------------|---|
| | WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “1XX” in PWR_CR1 – WUFx bits are cleared in power status register 1 (PWR_SR1) |
| Mode entry | On return from ISR while: – SLEEPDEEP bit is set in Cortex®-M0+ System Control register – SLEEPONEXT = 1 – No interrupt is pending – LPMS = “1XX” in PWR_CR1 and – WUFx bits are cleared in power status register 1 (PWR_SR1) – 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 |
| Wake-up latency | Reset phase |

4.3.11 Auto-wakeup 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-wakeup 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 [RTC 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 clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC 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.

The LCD Start of frame interrupt can also be used as a periodic wake-up from Stop (0, 1 or 2) mode. The LCD is not available in Standby mode.

The LCD clock is derived from the RTC clock selected by RTCSEL[1:0].

4.4 PWR registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

4.4.1 Power control register 1 (PWR_CR1)

Address offset: 0x00

Reset value: 0x0000 0208

This register is reset after wake-up from 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. | LPR | Res. | Res. | Res. | VOS[1:0] | DBP | Res. | Res. | FPD_L_PSLP | FPD_L_PRUN | FPD_STOP | LPMS[2:0] | | | |
| | rw | | | | rw | rw | rw | | 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: Stop 2 mode cannot be entered when LPR bit is set. Stop 1 is entered instead.

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

Bits 10:9 **VOS[1:0]**: Voltage scaling range selection

- 00: Cannot be written (forbidden by hardware)
- 01: Range 1
- 10: Range 2
- 11: Cannot be written (forbidden by hardware)

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 **FPD_LPSLP**: Flash memory powered down during Low-power sleep mode.

This bit determines whether the flash memory is put in power-down mode or remains in idle mode when the device enters Low-power sleep mode.

- 0: Flash memory idle
- 1: Flash memory powered down

Bit 4 **FPD_LPRUN**: Flash memory powered down during Low-power run mode.

This bit determines whether the flash memory is put in power-down mode or remains in idle mode when the device enters Low-power sleep mode.

- 0: Flash memory idle
- 1: Flash memory powered down

Bit 3 **FPD_STOP**: Flash memory powered down during Stop mode.

This bit determines whether the flash memory is put in power-down mode or remains in idle mode when the device enters Stop mode.

- 0: Flash memory idle
- 1: Flash memory powered down

Bits 2:0 **LPMS[2:0]**: Low-power mode selection

These bits select the low-power mode entered when CPU enters the deepsleep mode.

- 000: Stop 0 mode
- 001: Stop 1 mode
- 010: Stop 2 mode
- 011: Standby mode
- 1xx: Shutdown mode

Note: If LPR bit is set, Stop 2 mode cannot be selected and Stop 1 mode shall be entered instead of Stop 2.

In Standby mode, SRAM2 can be preserved or not, depending on RRS bit configuration in PWR_CR3.

4.4.2 Power control register 2 (PWR_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

This register is reset when exiting the 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. | USV ⁽¹⁾ | Res. | Res. | Res. | PVME4 | PVME3 | PVME1 (1) | PLS[2:0] | | | PVDE |
| | | | | | rw | | | | rw | rw | rw | rw | rw | rw | rw |

1. Available on STM32U0x3xx devices only.

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

Bit 10 **USV⁽¹⁾**: V_{DDUSB} USB supply valid

This bit is used to validate the V_{DDUSB} supply for electrical and logical isolation purpose.

Setting this bit is mandatory to use the USB FS peripheral. If V_{DDUSB} is not always present in the application, the PVM can be used to determine whether this supply is ready or not.

0: V_{DDUSB} is not present. Logical and electrical isolation is applied to ignore this supply.

1: V_{DDUSB} is valid.

Bits 9:7 Reserved, must be kept at reset value.

- Bit 6 **PVME4**: Peripheral voltage monitoring 4 enable: V_{DDA} vs. 1.86 V
 0: PVM4 (V_{DDA} monitoring vs. 1.86 V threshold) disable.
 1: PVM4 (V_{DDA} monitoring vs. 1.86 V threshold) enable.
- Bit 5 **PVME3**: Peripheral voltage monitoring 3 enable: V_{DDA} vs. 1.62 V
 0: PVM3 (V_{DDA} monitoring vs. 1.62 V threshold) disable.
 1: PVM3 (V_{DDA} monitoring vs. 1.62 V threshold) enable.
- Bit 4 **PVME1⁽¹⁾**: Peripheral voltage monitoring 1 enable: V_{DDUSB} vs. 1.2 V
 0: PVM1 (V_{DDUSB} monitoring vs. 1.2 V threshold) disable.
 1: PVM1 (V_{DDUSB} monitoring vs. 1.2 V threshold) enable.
- 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} around 2.0 V
 001: V_{PVD1} around 2.2 V
 010: V_{PVD2} around 2.4 V
 011: V_{PVD3} around 2.5 V
 100: V_{PVD4} around 2.6 V
 101: V_{PVD5} around 2.8 V
 110: V_{PVD6} around 2.9 V
 111: External input analog voltage PVD_IN (compared internally to VREFINT)
Note: These bits are write-protected when the bit PVDL (PVD Lock) is set in the SYSCFG_CBR register.
These bits are reset only by a system reset.
- Bit 0 **PVDE**: Programmable voltage detector enable
 0: Programmable voltage detector disable.
 1: Programmable voltage detector enable.
Note: This bit is write-protected when the bit PVDL (PVD Lock) is set in the SYSCFG_CBR register.
This bit is reset only by a system reset.

1. Available on STM32U0 devices only.

4.4.3 Power control register 3 (PWR_CR3)

Address offset: 0x08

Reset value: 0x0000 8000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 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 | Res. | Res. | Res. | Res. | APC | ENULP | RRS | Res. | EWUP 7 | Res. | EWUP 5 | EWUP 4 | EWUP 3 | EWUP 2 | EWUP 1 |
| 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

- 0: Internal wake-up line disable.
- 1: Internal wake-up line enable.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **APC**: Apply pull-up and pull-down configuration

When this bit is set, the I/O pull-up and pull-down configurations defined in the PWR_PUCRx and PWR_PDCRx registers are applied. When this bit is cleared, the PWR_PUCRx and PWR_PDCRx registers are not applied to the I/Os, instead the I/Os are in floating mode during Standby or configured according GPIO controller GPIOx_PUPDR register during RUN mode.

Bit 9 **ENULP**: Enable ULP sampling

When this bit is set, the BORL, BORH and PVD are periodically sampled instead continuous monitoring to reduce power consumption. Fast supply drop between two sample/compare phases is not detected in this mode. This bit has impact only on Stop 2 and Standby low-power modes.

Bit 8 **RSS**: SRAM2 retention in Standby mode

- 0: SRAM2 is powered off in Standby mode (SRAM2 content is lost).
- 1: SRAM2 is powered by the low-power regulator in Standby mode (SRAM2 content is kept).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **EWUP7**: Enable Wake-up pin WKUP7.

When this bit is set, the external wake-up pin WKUP7 is enabled and triggers a wake-up from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP7 bit in the PWR_CR4 register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **EWUP5**: Enable Wake-up pin WKUP5

When this bit is set, the external wake-up pin WKUP5 is enabled and triggers a wake-up from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP5 bit in the PWR_CR4 register.

Bit 3 **EWUP4**: Enable Wake-up pin WKUP4

When this bit is set, the external wake-up pin WKUP4 is enabled and triggers a wake-up from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP4 bit in the PWR_CR4 register.

Bit 2 **EWUP3**: Enable Wake-up pin WKUP3

When this bit is set, the external wake-up pin WKUP3 is enabled and triggers a wake-up from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP3 bit in the PWR_CR4 register.

Bit 1 **EWUP2**: Enable Wake-up pin WKUP2

When this bit is set, the external wake-up pin WKUP2 is enabled and triggers a wake-up from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP2 bit in the PWR_CR4 register.

Bit 0 **EWUP1**: Enable Wake-up pin WKUP1

When this bit is set, the external wake-up pin WKUP1 is enabled and triggers a wake-up from Standby or Shutdown event when a rising or a falling edge occurs. The active edge is configured via the WP1 bit in the PWR_CR4 register.

4.4.4 Power control register 4 (PWR_CR4)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | Res. | VBRS | VBE | Res. | WP7 | Res. | WP5 | WP4 | WP3 | WP2 | WP1 |
| | | | | | | rw | rw | | rw | | rw | rw | rw | rw | rw |

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

Bit 9 **VBRS**: V_{BAT} battery charging resistor selection

- 0: Charge V_{BAT} through a 5 kOhms resistor
- 1: Charge V_{BAT} through a 1.5 kOhms resistor

Bit 8 **VBE**: V_{BAT} battery charging enable

- 0: V_{BAT} battery charging disable
- 1: V_{BAT} battery charging enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **WP7**: Wake-up pin WKUP7 polarity

This bit defines the polarity used for an event detection on external wake-up pin, WKUP7
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WP5**: Wake-up pin WKUP5 polarity

This bit defines the polarity used for an event detection on external wake-up pin, WKUP5
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 3 **WP4**: Wake-up 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)

Bit 2 **WP3**: Wake-up pin WKUP3 polarity

This bit defines the polarity used for an event detection on external wake-up pin, WKUP3
 0: Detection on high level (rising edge)
 1: Detection on low level (falling edge)

Bit 1 **WP2**: Wake-up pin WKUP2 polarity

This bit defines the polarity used for an event detection on external wake-up pin, WKUP2
 0: Detection on high level (rising edge)
 1: Detection on low level (falling edge)

Bit 0 **WP1**: Wake-up 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)

4.4.5 Power status register 1 (PWR_SR1)

Address offset: 0x10

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: 2 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 | Res. | Res. | Res. | STOPF[2:0] | | | SBF | Res. | WUF7 | Res. | WUF5 | WUF4 | WUF3 | WUF2 | WUF1 |
| r | | | | r | r | r | r | | r | | r | r | r | r | r |

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

Bit 15 **WUFI**: Wake-up flag internal

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.

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

Bits 11:9 **STOPF[2:0]**: Stop Flags

These bits are set by hardware when the device enters any stop mode and are cleared by setting the CSBF bit in the PWR_SCR register, or by a power-on reset. It is not cleared by the system reset.

000: The device did not enter any Stop mode.

100: The device entered in Stop 0 mode.

101: The device entered in Stop 1 mode.

110: The device entered in Stop 2 mode.

Bit 8 **SBF**: Standby flag

This bit is set by hardware when the device enters the Standby mode and is cleared by setting the CSBF bit in the PWR_SCR register, or by a power-on reset. It is not cleared by the system reset.

0: The device did not enter the Standby mode

1: The device entered the Standby mode

Bit 7 Reserved, must be kept at reset value.

Bit 6 **WUF7**: Wake-up flag 7

This bit is set when a wake-up event is detected on wake-up pin, WKUP7. It is cleared by writing '1' in the CWUF7 bit of the PWR_SCR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WUF5**: Wake-up flag 5

This bit is set when a wake-up event is detected on wake-up pin, WKUP5. It is cleared by writing '1' in the CWUF5 bit of the PWR_SCR register.

Bit 3 **WUF4**: Wake-up 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_SCR register.

Bit 2 **WUF3**: Wake-up flag 3

This bit is set when a wake-up event is detected on wake-up pin, WKUP3. It is cleared by writing '1' in the CWUF3 bit of the PWR_SCR register.

Bit 1 **WUF2**: Wake-up flag 2

This bit is set when a wake-up event is detected on wake-up pin, WKUP2. It is cleared by writing '1' in the CWUF2 bit of the PWR_SCR register.

Bit 0 **WUF1**: Wake-up 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_SCR register.

4.4.6 Power status register 2 (PWR_SR2)

Address offset: 0x14

Reset value: 0x0000 0000

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 |
| PVMO4 | PVMO3 | Res. | PVMO1 (1) | PVDO | VOSF | REGLP F | REGLP S | FLASH _RDY | Res. |
| r | r | | r | r | r | r | r | r | | | | | | | |

- Available on STM32U0x3xx devices only.

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

Bit 15 **PVMO4**: Peripheral voltage monitoring output: V_{DDA} vs. 2.2 V

0: V_{DDA} voltage is above PVM4 threshold (around 2.2 V).

1: V_{DDA} voltage is below PVM4 threshold (around 2.2 V).

Note: PVMO4 is cleared when PVM4 is disabled (PVME4 = 0). After enabling PVM4, the PVM4 output is valid after the PVM4 wake-up time.

Bit 14 **PVMO3**: Peripheral voltage monitoring output: V_{DDA} vs. 1.62 V

0: V_{DDA} voltage is above PVM3 threshold (around 1.62 V).

1: V_{DDA} voltage is below PVM3 threshold (around 1.62 V).

Note: PVMO3 is cleared when PVM3 is disabled (PVME3 = 0). After enabling PVM3, the PVM3 output is valid after the PVM3 wake-up time.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PVMO1⁽¹⁾**: Peripheral voltage monitoring output: V_{DDUSB} vs. 1.2 V

0: V_{DDUSB} voltage is above PVM1 threshold (around 1.2 V).

1: V_{DDUSB} voltage is below PVM1 threshold (around 1.2 V).

Note: PVMO1 is cleared when PVM1 is disabled (PVME1 = 0). After enabling PVM1, the PVM1 output is valid after the PVM1 wake-up time.

Bit 11 **PVDO**: Programmable voltage detector output

0: V_{DD} is above the selected PVD threshold

1: V_{DD} is below the selected PVD threshold

Bit 10 **VOSF**: Voltage scaling flag

A delay is required for the internal regulator to be ready after the voltage scaling has been changed. VOSF indicates that the regulator reached the voltage level defined with VOS bits of the PWR_CR1 register.

0: The regulator is ready in the selected voltage range

1: The regulator output voltage is changing to the required voltage level

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

Bit 7 FLASH_RDY: Flash ready flag

This bit is set by hardware to indicate when the flash memory is ready to be accessed after wake-up from power-down. To place the flash memory in power-down, set either FPD_LPRUN, FPD_LPSLP or FPD_STP bits.

0: Flash memory in power down

1: Flash memory ready to be accessed

Note: If the system boots from SRAM, the user application must wait until the FLASH_RDY bit is set, prior to jumping to flash memory.

Bits 6:0 Reserved, must be kept at reset value.

- Available on STM32U0x3xx devices only.

4.4.7 Power status clear register (PWR_SCR)

Address offset: 0x18

Reset value: 0x0000 0000

Access: 3 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | CSBF | Res. | CWUF 7 | Res. | CWUF 5 | CWUF 4 | CWUF 3 | CWUF 2 | CWUF 1 |
| | | | | | | w | | w | | w | w | w | w | w | w |

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

Bit 8 CSBF: Clear standby flag

Setting this bit clears the SBF flag in the PWR_SR1 register.

Bit 7 Reserved, must be kept at reset value.

Bit 6 CWUF7: Clear wake-up flag 7

Setting this bit clears the WUF7 flag in the PWR_SR1 register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 CWUF5: Clear wake-up flag 5

Setting this bit clears the WUF5 flag in the PWR_SR1 register.

Bit 3 CWUF4: Clear wake-up flag 4

Setting this bit clears the WUF4 flag in the PWR_SR1 register.

Bit 2 **CWUF3**: Clear wake-up flag 3

Setting this bit clears the WUF3 flag in the PWR_SR1 register.

Bit 1 **CWUF2**: Clear wake-up flag 2

Setting this bit clears the WUF2 flag in the PWR_SR1 register.

Bit 0 **CWUF1**: Clear wake-up flag 1

Setting this bit clears the WUF1 flag in the PWR_SR1 register.

4.4.8 Power Port A pull-up control register (PWR_PUCRA)

Address offset: 0x20

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PUsy**: Port A pull-up bit y (y = 15 to 0)

When set, this bit activates the pull-up on PA[y] when APC bit is set in PWR_CR3 register. If the corresponding PDy bit is also set, the pull-up is not activated and the pull-down is activated instead with highest priority.

4.4.9 Power Port A pull-down control register (PWR_PDCRA)

Address offset: 0x24

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PDy**: Port A pull-down bit y (y = 15 to 0)

When set, this bit activates the pull-down on PA[y] when APC bit is set in PWR_CR3 register.

4.4.10 Power Port B pull-up control register (PWR_PUCRB)

Address offset: 0x28

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PUsy**: Port B pull-up bit y (y = 15 to 0)

When set, this bit activates the pull-up on PB[y] when APC bit is set in PWR_CR3 register.

4.4.11 Power Port B pull-down control register (PWR_PDCRB)

Address offset: 0x2C

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PDy**: Port B pull-down bit y (y = 15 to 0)

When set, this bit activates the pull-down on PB[y] when APC bit is set in PWR_CR3 register.

4.4.12 Power Port C pull-up control register (PWR_PUCRC)

Address offset: 0x30

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PUs**: Port C pull-up bit y (y = 15 to 0)

When set, this bit activates the pull-up on PC[y] when APC bit is set in PWR_CR3 register. If the corresponding PDy bit is also set, the pull-up is not activated and the pull-down is activated instead with highest priority.

4.4.13 Power Port C pull-down control register (PWR_PDCRC)

Address offset: 0x34

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

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

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

Bits 15:0 **PDy**: Port C pull-down bit y (y = 15 to 0)

When set, this bit activates the pull-down on PC[y] when APC bit is set in PWR_CR3 register.

4.4.14 Power Port D pull-up control register (PWR_PUCRD)

Address offset: 0x38

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | PU13 | PU12 | PU11 | PU10 | PU9 | PU8 | Res. | PU6 | PU5 | PU4 | PU3 | PU2 | PU1 | PU0 |
| | | rw | rw | rw | rw | rw | rw | | rw |

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

Bits 13:8 **PUy**: Port D pull-up bit y (y = 13 to 8)

When set, this bit activates the pull-up on PD[y] when APC bit is set in PWR_CR3 register. If the corresponding PDy bit is also set, the pull-up is not activated and the pull-down is activated instead with highest priority.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **PUy**: Port D pull-up bit y (y = 6 to 0)

When set, this bit activates the pull-up on PD[y] when APC bit is set in PWR_CR3 register. If the corresponding PDy bit is also set, the pull-up is not activated and the pull-down is activated instead with highest priority.

4.4.15 Power Port D pull-down control register (PWR_PDCRD)

Address offset: 0x3C

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | PD13 | PD12 | PD11 | PD10 | PD9 | PD8 | Res. | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| | | rw | rw | rw | rw | rw | rw | | rw |

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

Bits 13:8 **PDy**: Port D pull-down bit y (y = 13 to 8)

When set, this bit activates the pull-down on PD[y] when APC bit is set in PWR_CR3 register.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **PDy**: Port D pull-down bit y (y = 6 to 0)

When set, this bit activates the pull-down on PD[y] when APC bit is set in PWR_CR3 register.

4.4.16 Power Port E pull-up control register (PWR_PUCRE)

Address offset: 0x40

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 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 | Res. | Res. | Res. | PU3 | Res. | Res. | Res. |
| | | | | | | rw | rw | rw | | | | rw | | | |

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

Bits 9:7 **PUy**: Port E pull-up bit y (y = 9 to 7)

When set, this bit activates the pull-up on PE[y] when APC bit is set in PWR_CR3 register. If the corresponding PDy bit is also set, the pull-up is not activated and the pull-down is activated instead with highest priority.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **PU3**: Port E pull-up bit 3

When set, this bit activates the pull-up on PE[y] when APC bit is set in PWR_CR3 register. If the corresponding PDy bit is also set, the pull-up is not activated and the pull-down is activated instead with highest priority.

Bits 2:0 Reserved, must be kept at reset value.

4.4.17 Power Port E pull-down control register (PWR_PDCRE)

Address offset: 0x44

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 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 | Res. | Res. | Res. | PD3 | Res. | Res. | Res. |
| | | | | | | rw | rw | rw | | | | rw | | | |

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

Bits 9:7 **PDy**: Port E pull-down bit y (y = 9 to 7)

When set, this bit activates the pull-down on PE[y] when APC bit is set in PWR_CR3 register.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **PD3**: Port E pull-down bit 3

When set, this bit activates the pull-down on PE[y] when APC bit is set in PWR_CR3 register.

Bits 2:0 Reserved, must be kept at reset value.

4.4.18 Power Port F pull-up control register (PWR_PUCRF)

Address offset: 0x48

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | PU3 | PU2 | PU1 | PU0 |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bits 3:0 **PUy**: Port F pull-up bit y (y = 3 to 0)

When set, this bit activates the pull-up on PH[y] when APC bit is set in PWR_CR3 register. If the corresponding PDy bit is also set, the pull-up is not activated and the pull-down is activated instead with highest priority.

4.4.19 Power Port F pull-down control register (PWR_PDCRF)

Address offset: 0x4C

Reset value: 0x0000 0000

This register is neither reset when exiting Standby or Shutdown mode, nor by the PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | PD3 | PD2 | PD1 | PD0 |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bits 3:0 **PDy**: Port F pull-down bit y (y = 3 to 0)

When set, this bit activates the pull-down on PH[y] when APC bit is set in PWR_CR3 register.

4.4.20 PWR register map and reset value table

Table 31. PWR register map and reset values

Table 31. 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 |
|--------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---|
| 0x03C | PWR_PDCRD | Res | |
| | Reset value | Res | | |
| 0x040 | PWR_PUCRE | Res | | |
| | Reset value | Res | | |
| 0x044 | PWR_PDCRE | Res | | |
| | Reset value | Res | | |
| 0x048 | PWR_PUCRF | Res | | |
| | Reset value | Res | | |
| 0x04C | PWR_PDCRF | Res | | |
| | Reset value | Res | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses

5 Reset and clock control (RCC)

5.1 Reset

There are three types of reset, defined as system reset, power reset and RTC domain reset.

5.1.1 Power reset

A power reset is generated when one of the following events occurs:

- power-on reset (POR) or brown-out reset (BOR)
- exit from Standby mode
- exit from Shutdown mode

Power and brown-out reset set all registers to their reset values except the registers of the RTC 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 mode control) are not impacted.

When exiting Shutdown mode, the brown-out reset is generated, resetting all registers except those in the RTC domain.

5.1.2 System reset

A system reset sets all registers to their reset value unless otherwise specified in the register descriptions.

System reset is generated when one of the following events occurs:

- low level on the NRST pin (external reset)
- window watchdog event (WWDG reset)
- independent watchdog event (IWDG reset)
- software reset (SW reset) (see [Software reset](#))
- low-power mode security reset (see [Low-power mode security reset](#))
- option byte loader reset (see [Option byte loader reset](#))
- power-on reset

The reset source can be identified by checking the reset flags in the RCC_CSR register (see [Section 5.4.23: Control/status register \(RCC_CSR\)](#)).

NRST pin (external reset):

Through specific option bits, the NRST pin is configurable for operating as:

- **Reset input/output** (default at device delivery)

Valid reset signal on the pin is propagated to the internal logic, and each internal reset source is led to a pulse generator the output of which drives this pin. The GPIO functionality (PF2) is not available. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source to be output on the NRST pin. An internal reset holder option can be used, if enabled in the option bytes, to ensure that the pin is pulled low until its voltage meets V_{IL} threshold. This function allows the

detection of internal reset sources by external components when the line faces a significant capacitive load.

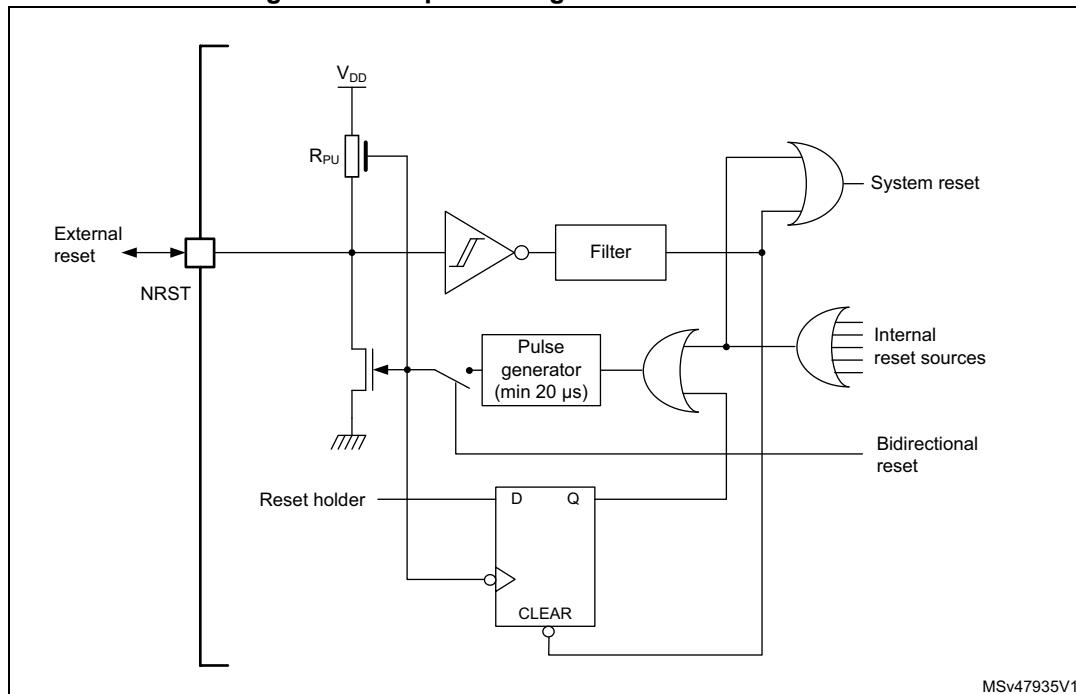
- **Reset input**

In this mode, any valid reset signal on the NRST pin is propagated to device internal logic, but resets generated internally by the device are not visible on the pin. In this configuration, GPIO functionality (PF2) is not available.

- **GPIO**

In this mode, the pin can be used as PF2 standard GPIO. The reset function of the pin is not available. Reset is only possible from device internal reset sources and it is not propagated to the pin.

Figure 10. Simplified diagram of the reset circuit



MSv47935V1

Caution: Upon power reset or wake-up from shutdown mode, the NRST pin is configured as Reset input/output and driven low by the system until it is reconfigured to the expected mode when the option bytes are loaded, in the fourth clock cycle after the end of $t_{rsttempo}$.

Software reset

The SYSRESETREQ bit in Cortex®-M0+ Application interrupt and reset control register must be set to force a software reset on the device (refer to the programming manual PM0223).

Low-power mode security reset

To prevent that critical applications mistakenly enter a low-power mode, three low-power mode security resets are available. If enabled in option bytes, the resets are generated in the following conditions:

- **Entering Standby mode**

This type of 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.

- **Entering Stop mode**

This type of 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.

- **Entering Shutdown mode**

This type of 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: FLASH option bytes](#).

Option byte loader reset

The option byte loader reset is generated when the OBL_LAUNCH bit (bit 27) is set in the FLASH_CR register. This bit is used to launch the option byte loading by software.

5.1.3 RTC domain reset

The RTC domain has three specific resets.

A RTC domain reset is generated when one of the following events occurs:

- **Software reset**, triggered by setting the BDRST bit in the [RTC domain control register \(RCC_BDCR\)](#).
- **V_{DD} or V_{BAT} power on**, if both supplies have previously been powered off.
- **Exit from Shutdown**, forced when the BDRST bit in the [FLASH option register \(FLASH_OPTR\)](#) is set.

A RTC domain reset only affects the LSE oscillator, the RTC, the backup registers and the RCC RTC domain control register.

5.2 Clocks

The device provides the following clock sources producing primary clocks:

- **HSI16 RC** - a high-speed fully-integrated RC oscillator producing **HSI16** clock (about 16 MHz)
- **HSI48 RC** - a high-speed fully-integrated RC oscillator producing **HSI48** clock for USB and RNG (about 48 MHz)
- **MSI** - (multispeed internal) RC oscillator clock
- **HSE OSC** - a high-speed oscillator with external crystal/ceramic resonator or external clock source, producing **HSE** clock (4 to 48 MHz)
- **LSI RC** - a low-speed fully-integrated RC oscillator producing **LSI** clock (about 32 kHz)
- **LSE OSC** - a low-speed oscillator with external crystal/ceramic resonator or external clock source, producing **LSE** clock (accurate 32.768 kHz or external clock up to 1 MHz)

Each oscillator can be switched on or off independently when it is not used, to optimize power consumption. Check sub-sections of this section for more functional details. For electrical characteristics of the internal and external clock sources, refer to the device datasheet.

The device produces secondary clocks by dividing or/and multiplying the primary clocks:

- **HSISYS** - a clock derived from HSI16 through division by a factor programmable from 1 to 128
- **PLLCLK, PLLQCLK and PLLRCLK** - clocks output from the PLL block
- **SYSCLK** - a clock obtained through selecting one of LSE, LSI, HSE, MSI, PLLRCLK, and HSISYS clocks
- **HCLK** - a clock derived from SYSCLK through division by a factor programmable from 1 to 512
- **HCLK8** - a clock derived from HCLK through division by eight
- **PCLK** - a clock derived from HCLK through division by a factor programmable from 1 to 16
- **LPTIMx_IN** - clock from LPTIMx_INx pins, selectable for the LPTIM peripheral

More secondary clocks are generated by fixed division of HSE, HSI16 and HCLK clocks.

The MSI in Range 6 is used as system clock source after startup from reset, delivering a 4 MHz frequency.

The HCLK clock and PCLK clock are used for clocking the AHB and the APB domains, respectively. Their maximum allowed frequency is 64 MHz.

The peripherals are clocked with the clocks from the bus they are attached to (HCLK for AHB, PCLK for APB) except:

- **TIMx**, with these clock sources to select from:
 - TIMPCLK (selectable for all timers) running at PCLK frequency
 - PLLQCLK selectable for high-speed TIM1 and TIM15 timers
- **LPTIMx**, with these clock sources to select from:
 - LSI
 - LSE
 - HSI16
 - PCLK (APB clock)
 - LPTIMx_IN selected from LPTIMx_INx pins
- The functionality in Stop mode (including wake-up) is supported only when the clock is LSI or LSE.
- **ADC**, with these clock sources to select from:
 - SYSCLK (system clock)
 - HSI16
 - PLLPCLK
- **USARTx / LPUARTx**, with these clock sources to select from:
 - SYSCLK (system clock)
 - HSI16
 - LSE
 - PCLK (APB clock)
- The wake-up from Stop mode is supported only when the clock is HSI16 or LSE.
- **I2C1/I2C3**, with these clock sources to select from:
 - SYSCLK (system clock)
 - HSI16
 - PCLK (APB clock)
- The wake-up from Stop mode is supported only when the clock is HSI16.
- **RNG**, with these clock sources to select from:
 - SYSCLK (system clock)
 - HSI16 clock divided by 8
 - PLLQCLK
- The RNG clock can additionally be divided by 2, 4 or 8, using a dedicated prescaler.

- **RTC**, with these clock sources to select from:

- LSE
- LSI
- HSE clock divided by 32

The functionality in Stop mode (including wake-up) is supported only when the clock is LSI or LSE.

- **IWDG**, always clocked with LSI clock.
- **USB**, with these clocks to select from:

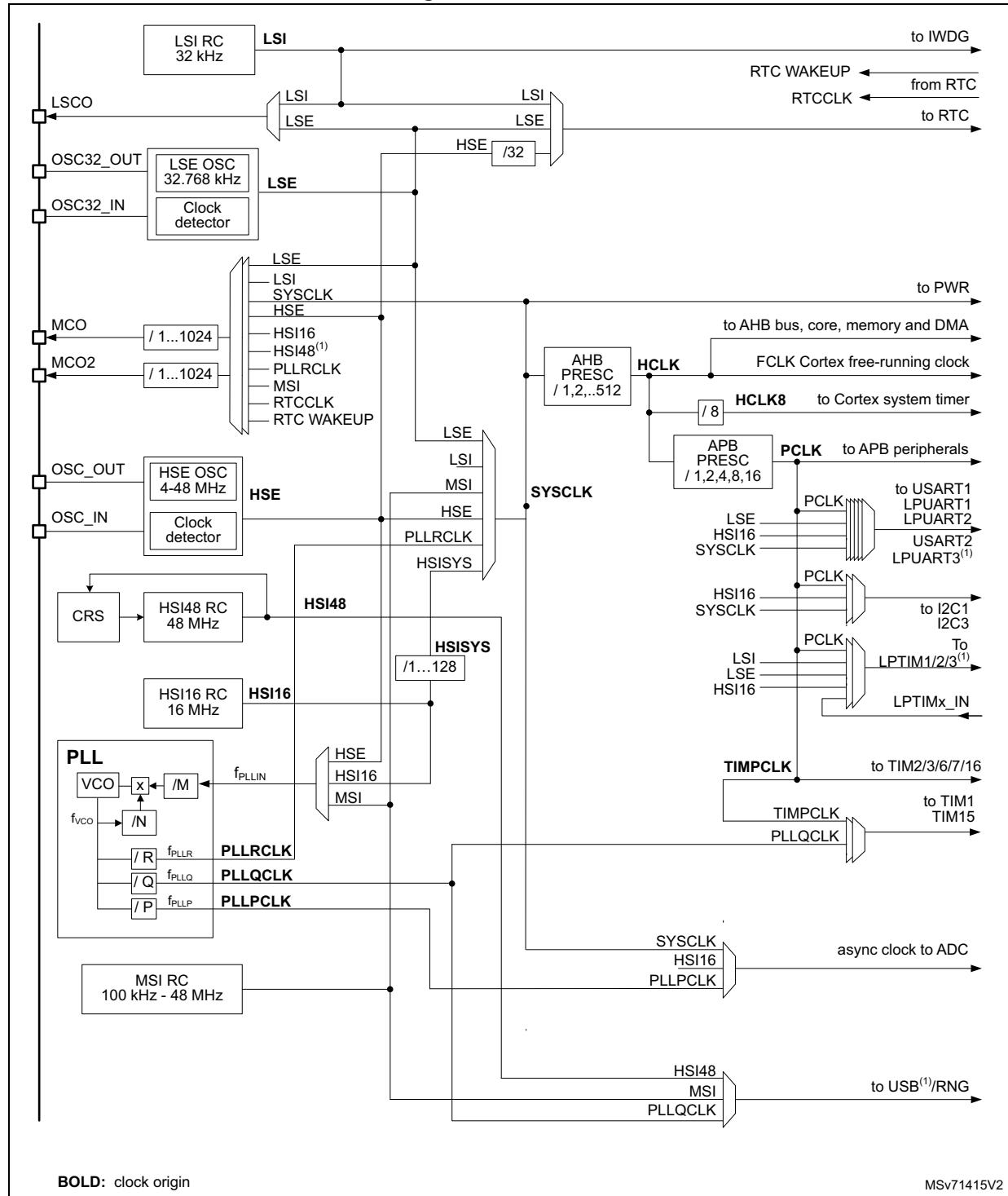
- MSI
- HSI48
- PLLQCLK

- **SysTick** (Cortex[®] core system timer), with these clock sources to select from:
 - HCLK (AHB clock)
 - HCLK clock divided by 8

The selection is done through SysTick control and status register.

HCLK is used as Cortex[®]-M0+ free-running clock (FCLK). For more details, refer to the programming manual PM0223.

Figure 11. Clock tree



- Only applies to STM32U073xx and STM32U083xx devices.

5.2.1 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources:

- HSE external crystal/ceramic resonator
- HSE user external clock

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 12. HSE/ LSE clock sources

| Clock source | Hardware configuration |
|----------------------------|------------------------|
| External clock | |
| Crystal/Ceramic resonators | |

External crystal/ceramic resonator (HSE crystal)

The 4 to 48 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in [Figure 12](#). Refer to the electrical characteristics section of the *datasheet* for more details.

The HSERDY flag in the [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 [Clock interrupt enable register \(RCC_CIER\)](#).

The HSE Crystal can be switched on and off using the HSEON bit in the [Clock control register \(RCC_CR\)](#).

External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 48 MHz. This mode is selected by setting the HSEBYP and HSEON bits in the [Clock control register \(RCC_CR\)](#). The external clock signal (square, sinus or triangle) with ~40-60 % duty cycle depending on the frequency (refer to the *datasheet*) must drive the OSC_IN pin (see [Figure 12](#)).

The OSC_OUT pin can be used as a GPIO or it can be configured as OSC_EN alternate function, to provide an enable signal to external clock synthesizer. It allows stopping the external clock source when the device enters low-power modes, as the OSC_EN output is high in Run mode and low when the device is in low-power mode

Note: *For details on pin availability, refer to the pinout section in the corresponding device datasheet.*

To minimize the consumption, it is recommended to use the square signal.

5.2.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz RC oscillator.

The HSI16 RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator. However, even after calibration, it is less accurate than an oscillator using a frequency reference such as quartz crystal or ceramic resonator.

The HSISYS clock derived from HSI16 can be selected as system clock after wake-up from Stop modes (Stop 0 or Stop 1). Refer to [Section 5.3: Low-power modes](#). It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 5.2.10: Clock security system \(CSS\)](#).

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. To compensate for this variation, each device is factory calibrated to 1 % accuracy at $T_A=25^\circ\text{C}$.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the [Internal clock sources calibration register \(RCC_ICSCR\)](#).

Voltage or temperature variations in the application may affect the HSI16 frequency of the RC oscillator. It can be trimmed using the HSITRIM[6:0] bits in the [Internal clock sources calibration register \(RCC_ICSCR\)](#).

For more details on how to measure the HSI16 frequency variation, refer to [Section 5.2.17: Internal/external clock measurement with TIM16](#).

The HSIRDY flag in the [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 [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 5.2.10: Clock security system \(CSS\) on page 163](#).

5.2.3 HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator. It can be used as clock source for the USB and RNG peripherals.

The internal 48MHz RC oscillator provides a high-precision clock to the USB peripheral thanks to the clock recovery system (CRS). CRS uses the USB SOF signal, LSE clock or an external signal as timing reference to precisely adjust the HSI48 RC oscillator frequency.

HSI48 RC oscillator is disabled as soon as the system enters in Stop or Standby mode. When the CRS is not used, the HSI48 RC oscillator runs on its free-run frequency which is subject to manufacturing process variations. The devices are factory-calibrated for ~3 % accuracy at $T_A = 25^\circ\text{C}$.

Refer to CRS section for more details on how to configure and use CRS.

The HSI48RDY flag in the RCC_CR register indicates if HSI48 is stable or not. At startup, the HSI48 clock is not released until this flag is set by hardware.

The HSI48 RC oscillator is enabled/disabled through the HSI48ON bit of the RCC_CR register. It is automatically enabled (by hardware setting the HSI48ON bit) when selected as clock source for the USB peripheral, as long as the USB peripheral is enabled.

Furthermore, it is possible to output the HSI48 clock through the MCO and MCO2 multiplexers and use it as a clock source for other application components.

5.2.4 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 [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, wake-up from Standby and Shutdown low-power modes. After restart from Reset, the MSI frequency is set to its default value 4 MHz. Refer to [Section 5.3: Low-power modes](#).

The MSI clock can be selected as system clock after a wake-up from Stop mode (Stop 0, Stop 1 or Stop 2). Refer to [Section 5.3: Low-power modes](#). It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 5.2.10: Clock security system \(CSS\)](#).

The MSI RC oscillator has the advantage of providing a low-cost (no external components) low-power clock source. In addition, when used in PLL-mode with the LSE, it provides a very accurate clock source which can be used by the USB FS device, and feed the main PLL to run the system at the maximum speed 80 MHz.

The MSIRDY flag in the [Clock control register \(RCC_CR\)](#) indicates whether 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 [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 MSIPLEN bit in the [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. At 48 MHz, the MSI in PLL-mode can be used for the USB FS device, saving the need of an external high-speed crystal.

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 an ambient temperature, TA, of 25 °C. After reset, the factory calibration value is loaded in the MSICAL[7:0] bits in the [Internal clock sources calibration register \(RCC_ICSCR\)](#). If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. You can trim the MSI frequency in the application by using the MSITRIM[7:0] bits in the RCC_ICSCR register. For more details on how to measure the MSI frequency variation please refer to [Section 5.2.17: Internal/external clock measurement with TIM16](#).

Note: *Hardware auto calibration with LSE must not be used in conjunction with software calibration.*

5.2.5 PLL

The internal PLL multiplies the frequency of HSI16- or HSE-based clock fetched on its input, to produce three independent clock outputs. The allowed input frequency range is from 2.66 to 16 MHz. The dedicated divider PLLM with division factor programmable from one to eight allows setting a frequency within the valid PLL input range. Refer to [Figure 11: Clock tree](#) and [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 [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, and PLLREN in [PLL configuration register \(RCC_PLLCFGR\)](#).

An interrupt can be generated when the PLL is ready, if enabled in the [Clock interrupt enable register \(RCC_CIER\)](#).

The enable bit of each PLL output clock (PLLPEN, PLLQEN, and PLLREN) can be modified at any time without stopping the PLL. PLLREN cannot be cleared if PLLRCLK is used as system clock.

OSC32_OUT can be configured as OSC32_EN alternate function to control the external clock source. The OSC32_EN output is high in Run mode and low when the device is in low-power mode.

5.2.6 LSE clock

The LSE crystal is a 32.768 kHz 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 LSE crystal is switched on and off using the LSEON bit in [RTC domain control register \(RCC_BDCR\)](#). The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the [RTC 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 can not be increased if LSEON=1.

The LSERDY flag in the [RTC 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 [Clock interrupt enable register \(RCC_CIER\)](#).

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. This mode is selected by setting the LSEBYP and LSEON bits in the [AHB peripheral clock enable in Sleep/Stop mode register \(RCC_AHBSMENR\)](#). 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 12](#).

5.2.7 LSI clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and RTC. The clock frequency is 32 kHz. For more details, refer to the electrical characteristics section of the datasheets.

The LSI RC can be switched on and off using the LSION bit in the [Control/status register \(RCC_CSR\)](#).

The LSIRDY flag in the [Control/status register \(RCC_CSR\)](#) indicates if the LSI 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 [Clock interrupt enable register \(RCC_CIER\)](#).

5.2.8 System clock (SYSCLK) selection

One of the following clocks can be selected as system clock (SYSCLK):

- LSI
- LSE
- MSI
- HSISYS
- HSE
- PLLRCLK

The system clock maximum frequency is 56 MHz. Upon system reset, the HSISYS clock derived from HSI16 oscillator 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 occurs when the clock source becomes ready. Status bits in the *Internal clock sources calibration register (RCC_ICSCR)* indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

5.2.9 Clock source frequency versus voltage scaling

The following table gives the different clock source frequencies depending on the product voltage range.

Table 32. Clock source frequency

| Clock | Maximum clock frequency (MHz) | |
|----------------|-------------------------------|-------------------|
| | Range 1 | Range 2 |
| HSI16 | 16 | 16 |
| HSE | 48 | 19 |
| HSI48 | 48 | N/A |
| MSI | 48 | 16 |
| PLLCLK | 122 ⁽¹⁾ | 40 ⁽²⁾ |
| PLLQCLK | 56 ⁽¹⁾ | 19 ⁽²⁾ |
| PLLRCLK | 56 ⁽¹⁾ | 19 ⁽²⁾ |
| SYSCLK | 56 ⁽¹⁾ | 19 ⁽²⁾ |
| HCLK | 56 ⁽¹⁾ | 19 ⁽²⁾ |
| PCLK | 56 ⁽¹⁾ | 19 ⁽²⁾ |

1. Maximum VCO frequency is 344 MHz.

2. Maximum VCO frequency is 128 MHz.

5.2.10 Clock security system (CSS)

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 TIM1, TIM15, TIM16 and TIM17 timers
- CSSI (clock security system interrupt) is generated

The CSSI is linked to the Cortex®-M0+ NMI (non-maskable interrupt) exception vector. It makes the software aware of a HSE clock failure to allow it to perform rescue operations.

Note: *If the CSS is enabled and the HSE clock fails, the CSSI occurs and an NMI is automatically generated. The NMI is executed infinitely unless the CSS interrupt pending bit is cleared. It*

is therefore necessary that the NMI ISR clears the CSSI by setting the CSSC bit in the [Clock interrupt clear register \(RCC_CICR\)](#).

If HSE is selected directly or indirectly (PLLRCLK selected for SYSCLK and HSE selected as PLL input) as system clock, and a failure of HSE clock is detected, the system clock switches automatically to HSISYS and the HSE oscillator is disabled. If the HSE clock (divided or not) is the clock entry of the PLL and PLLRCLK is used as system clock when the failure occurs, the PLL is disabled, too.

5.2.11 Clock security system for LSE clock (LSECSS)

A clock security system on LSE can be activated by setting the LSECSSON bit in [RTC domain control register \(RCC_BDCR\)](#). This bit can be cleared only by a hardware reset or RTC software reset, or after LSE clock failure detection. LSECSSON must be written after LSE and LSI are enabled (LSEON and LSION enabled) and ready (LSERDY and LSIRDY flags set by hardware), and after selecting the RTC clock by RTCSEL.

The LSECSS works in all modes except VBAT. It keeps working also under system reset (excluding power-on reset). If a failure is detected on the LSE oscillator, the LSE clock is no longer supplied to the RTC but its registers are not impacted.

Note: *If the LSECSS is enabled and the LSE clock fails, the LSECSSI occurs and an NMI is automatically generated. The NMI is executed infinitely unless the LSECSS interrupt pending bit is cleared. It is therefore necessary that the NMI ISR clears the LSECSSI by setting the LSECSSC bit in the [Clock interrupt clear register \(RCC_CICR\)](#).*

If LSE is used as system clock, and a failure of LSE clock is detected, the system clock switches automatically to LSI. In low-power modes, an LSE clock failure generates a wake-up. The interrupt flag must then be cleared within the RCC registers.

The software **must** then disable the LSECSSON bit, stop the defective 32 kHz oscillator (by clearing LSEON), and change the RTC clock source (no clock, LSI or HSE, with RTCSEL), or take any appropriate action to secure the application.

The frequency of the LSE oscillator must exceed 30 kHz to avoid false positive detections.

5.2.12 ADC clock

The ADC clock is derived from the system clock, or from the PLLPCLK output. It can reach 122 MHz and can be divided by the following prescalers 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 bitfields in the ADC1_CCR.

If the programmed factor is 1, the AHB prescaler must be set to 1.

5.2.13 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 [RTC domain control register \(RCC_BDCR\)](#). This selection cannot be modified without resetting the RTC domain. The system must always be configured so as to get a PCLK frequency greater than or equal to the RTCCLK frequency for a proper operation of the RTC.

The LSE clock is in the RTC 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.

5.2.14 Timer clock

The timer clock TIMPCLK is equal to the PCLK frequency.

For TIM1 and TIM15, PLLQCLK clock can also be selected, if:

- PCLK is derived from PLLRCLK, and
- PLLQCLK frequency is an integer multiplication by 2 or more of the PCLK frequency, without exceeding 56 MHz.

5.2.15 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator temporization, the clock is provided to the IWDG.

5.2.16 Clock-out capability

MCO and MCO2

The MCO and MCO2 pins output, independently of each other, the clock selected from:

- LSI
- LSE
- SYSCLK
- HSI16
- HSI48
- HSE
- MSI
- PLLRCLK
- RTCCLK
- RTC WAKEUP

The multiplexers for MCO and MCO2, respectively, are controlled by the MCSEL[3:0] and MC2SEL[3:0] bitfields of the [Clock configuration register \(RCC_CFGR\)](#). Their outputs are further divided by a factor set through the MCOPRE[2:0] and MC2PRE[2:0] bitfields of the [Clock configuration register \(RCC_CFGR\)](#).

LSCO

The LSCO pin allows outputting one of low-speed clocks:

- LSI
- LSE

The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN bit of the *RTC domain control register (RCC_BDCR)*. The configuration registers of the corresponding GPIO port must be programmed in alternate function mode.

This function remains available in Stop 0, Stop 1 and Standby modes.

5.2.17 Internal/external clock measurement with TIM16

It is possible to indirectly measure the frequency of all on-board clock sources with the TIM16 channel 1 input capture, as represented in *Figure 13*.

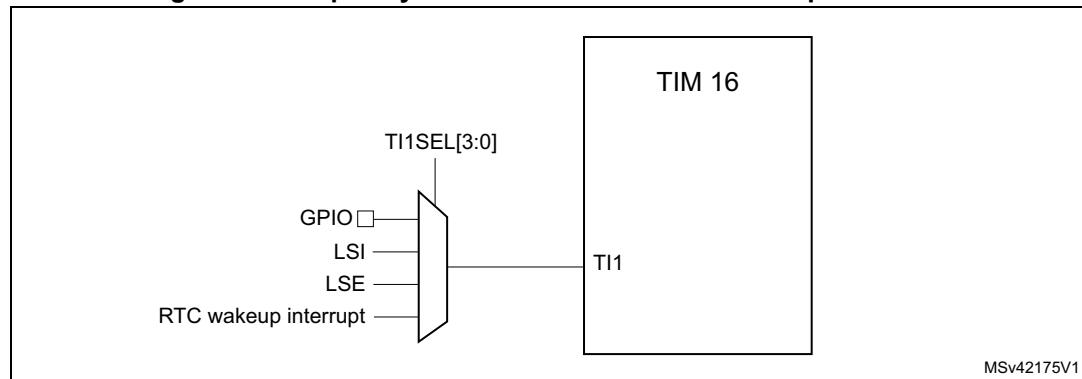
TIM16

By setting the TI1SEL[3:0] field of the TIM16_TISEL register, the clock selected for the input capture channel1 of TIM16 can be one of:

- GPIO (refer to the alternate function mapping in the device datasheets).
- LSI clock
- LSE clock
- RTC wake-up interrupt signal

The last option requires to enable the RTC interrupt.

Figure 13. Frequency measurement with TIM16 in capture mode



Calibration of the HSI16 or MSI oscillator

For TIM16, the primary purpose of connecting the LSE to the channel 1 input capture is to precisely measure HSISYS (derived from HSI16 or MSI) selected as system clock. Counting HSISYS clock pulses between consecutive edges of the LSE clock (the time reference) allows measuring the HSISYS (and HSI16 or MSI) clock period. Such measurement can determine the HSI16 or MSI oscillator frequency with nearly the same accuracy as the accuracy of the 32.768 kHz quartz crystal used with the LSE oscillator (typically a few tens of ppm). The HSI16 or MSI oscillators can then be trimmed to compensate for deviations from target frequency, due to manufacturing, process, temperature and/or voltage variation.

The HSI16 or MSI oscillators have dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (for example, the HSISYS/LSE ratio): the measurement accuracy is therefore closely related to the ratio between the two clock sources. Increasing the ratio allows improving the measurement accuracy.

Generated by the HSE oscillator, the HSE clock (divided by 32) used as time reference is the second best method for reaching a good HSI16 or MSI frequency measurement accuracy. It is recommended in absence of the LSE clock.

In order to further improve the precision of the HSI16 or MSI oscillator calibration, it is advised to employ one or a combination of the following measures to increase the frequency measurement accuracy:

- set the HSISYS divider to 1 for HSISYS frequency to be equal to HSI16 or MSI frequency
- average the results of multiple consecutive measurements
- use the input capture prescaler of the timer (one capture every up to eight periods)
- use LSE clock for the RTC and the RTC wake-up interrupt signal as time reference

The last point significantly increases the reference period for HSI16 or MSI clock pulse counting, which improves the accuracy of a single measurement. For operation, the RTC wake-up interrupt must be enabled.

Calibration of the HSI48 oscillator

The HSI48 oscillator is factory-calibrated.

Measurement of the LSI oscillator frequency

The calibration of the LSI oscillator uses the same principle as that for calibrating the HSI16 or MSI oscillator. TIM16 channel1 input capture must be used for LSI clock, and HSE selected as system clock source. The number of HSE clock pulses between consecutive edges of the LSI signal, counted by TIM16, is then representative of the LSI clock period.

5.2.18 Peripheral clock enable registers

The clocks to each peripheral can be enabled individually by the corresponding enable bit of the RCC_AHBENR register or by one of the RCC_APBENRx registers. The clocks to the I/O ports can be enabled individually through the RCC_IOPENR register.

When the clock to a peripheral or I/O port is not active, the read and write accesses to the corresponding registers are not effective.

Caution: The enable bit has a synchronization mechanism to create a glitch-free clock for the peripheral or I/O port. After the enable bit is set, there is a 2-clock-cycle delay before the clock becomes active, which the software must take into account.

5.3 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and Low-power Sleep modes stops the CPU clock. The memory interface clocks (flash memory and SRAM 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 and Stop 1) stop all the clocks in the V_{CORE} domain and disable the PLL as well as the HSI16, HSI48, MSI and HSE oscillators.

The USART1, USART2, LPUART1, LPUART2, LPUSART3, I2C1, and I2C3 peripherals can enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as clock source for one of those peripherals).

The USART1, USART2, LPUART1, LPUART2, LPUSART3, I2C1, and I2C3 peripherals can also operate with the clock from 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 set). In that case, the LSE oscillator remains active when the device enters Stop mode (these peripherals do not have the capability to turn on the LSE oscillator).

- Standby and Shutdown modes stop all clocks in the V_{CORE} domain and disable the PLL, as well as the HSI16, HSI48, and HSE oscillators.

The CPU deepsleep mode can be overridden for debugging, by setting the DBG_STOP or DBG_STANDBY bits in the DBG_CR register.

When leaving the Stop 0 or Stop 1 modes, HSISYS becomes automatically the system clock.

When leaving the Standby and Shutdown modes, HSISYS (with frequency equal to HSI16) becomes automatically the system clock. At wake-up from Standby and Shutdown mode, the user trim is lost.

If a flash memory programming operation is ongoing, Stop, Standby, and Shutdown entry is delayed until the flash memory interface access is finished. If an access to the APB domain is ongoing, the Stop, Standby, and Shutdown entry is delayed until the APB access is finished.

5.4 RCC registers

Unless otherwise specified, the RCC registers support word, half-word, and byte access, without any wait state.

5.4.1 Clock control register (RCC_CR)

This register supports only word and half-word access.

Address offset: 0x00

Power-on reset value: 0x0000 0083

Other types of reset: same as power-on reset, except HSEBYP bit that keeps its previous value.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|----------|---------|------------|-------|---------------|------|------|------|-----------|-----------|---------|--------|
| Res. | Res. | Res. | Res. | Res. | Res. | PLL RDY | PL隆ON | Res. | Res. | Res. | Res. | CSS ON | HSE BYP | HSE RDY | HSE ON |
| | | | | | | r | rw | | | | | rs | rw | r | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | HSI ASFS | HSI RDY | HSI KER ON | HSION | MSIRANGE[3:0] | | | | MSI RGSEL | MSI PLLEN | MSI RDY | MSION |
| | | | | r | r | rw | rw | rw | rw | rw | rw | rs | rw | r | rw |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **PLL RDY**: PLL clock ready flag

Set by hardware to indicate that the PLL is locked.

0: PLL unlocked

1: PLL locked

Bit 24 **PL隆ON**: 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 23:20 Reserved, must be kept at reset value.

Bit 19 **CSS ON**: Clock security system enable

Set by software to enable the clock security system. When CSS ON is set, the clock 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: Clock security system OFF (clock detector OFF)

1: Clock security system ON (Clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 18 **HSEBYP**: HSE crystal oscillator bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit set, to be used by the device. The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE crystal oscillator not bypassed

1: HSE crystal oscillator bypassed with external clock

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 6 HSE clock cycles.

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

Bit 11 **HSIASFS**: HSI16 automatic start from Stop

Set and cleared by software. When the system wake-up clock is MSI, this bit is used to wake up the HSI16 in parallel of the system wake-up.

0: HSI16 oscillator is not enabled by hardware when exiting Stop mode with MSI as wake-up clock.

1: HSI16 oscillator is enabled by hardware when exiting Stop mode with MSI as wake-up clock.

Bit 10 **HSIRDY**: HSI16 clock ready flag

Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSION is enabled by software by setting HSICON.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

Note: Once the HSICON bit is cleared, HSIRDY goes low after 6 HSI16 clock cycles.

Bit 9 **HSIKERON**: HSI16 always enable for peripheral kernels.

Set and cleared by software to force HSI16 ON even in Stop modes. The HSI16 can only feed USART1, USART2, CEC and I2C1 peripherals configured with HSI16 as kernel clock. Keeping the HSI16 ON in Stop mode allows avoiding to slow down the communication speed because of the HSI16 startup time. This bit has no effect on HSICON value.

0: No effect on HSI16 oscillator.

1: HSI16 oscillator is forced ON even in Stop mode.

Bit 8 **HSICON**: HSI16 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSI16 oscillator when entering Stop, Standby, or Shutdown mode.

Forced by hardware to keep the HSI16 oscillator ON when it is used directly or indirectly as system clock (also when leaving Stop, Standby, or Shutdown modes, or in case of failure of the HSE oscillator used for 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 MSIRGSEL is set. 12 frequency ranges are available:

- 0000: range 0 around 100 kHz
- 0001: range 1 around 200 kHz
- 0010: range 2 around 400 kHz
- 0011: range 3 around 800 kHz
- 0100: range 4 around 1M Hz
- 0101: range 5 around 2 MHz
- 0110: range 6 around 4 MHz (reset value)
- 0111: range 7 around 8 MHz
- 1000: range 8 around 16 MHz
- 1001: range 9 around 24 MHz
- 1010: range 10 around 32 MHz
- 1011: range 11 around 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 **MSIRGSEL**: MSI clock range selection

Set by software to select the MSI clock range with MSIRANGE[3:0]. Write 0 has no effect.

After a standby or a reset MSIRGSEL is at 0 and the MSI range value is provided by MSISRANGE in CSR register.

- 0: MSI Range is provided by MSISRANGE[3:0] in RCC_CSR register
- 1: MSI Range is provided by MSIRANGE[3:0] in the RCC_CR register

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

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

5.4.2 Internal clock sources calibration register (RCC_ICSCR)

Address offset: 0x04

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 | 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 when added to the HSICAL value, trim the HSI16 to 16 MHz \pm 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.

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.

5.4.3 Clock configuration register (RCC_CFGR)

One or two wait states are inserted if this register is accessed during clock source switch, and between zero and 15 wait states are inserted if during an update of APB or AHB prescaler values.

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------------|-----------|----|----|-------------|-----------|----|----|--------------|------|------|----------|--------------|----|---------|----|
| MCOPRE[3:0] | | | | MCOSEL[3:0] | | | | MCO2PRE[3:0] | | | | MCO2SEL[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 |
| STOP WUCK | PPRE[2:0] | | | | HPRE[3:0] | | | | Res. | Res. | SWS[2:0] | | | SW[2:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | | | | r | r | r | rw | rw |

Bits 31:28 MCOPRE[3:0]: Microcontroller clock output prescaler

This bitfield is controlled by software. It sets the division factor of the clock sent to the MCO output as follows:

0000: 1

0001: 2

0010: 4

...

0111: 128

1000: 256

1001: 512

1010: 1024

Others: reserved

It is highly recommended to set this field before the MCO output is enabled.

Bits 27:24 MCOSEL[3:0]: Microcontroller clock output clock selector

This bitfield is controlled by software. It sets the clock selector for MCO output as follows:

0000: no clock, MCO output disabled

0001: SYSCLK

0010: MSI

0011: HSI16

0100: HSE

0101: PLLRCLK

0110: LSI

0111: LSE

1000: HSI48

1001: RTCCCLK

1010: RTC WAKEUP

Others: Reserved

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:20 **MCO2PRE[3:0]**: Microcontroller clock output 2 prescaler

This bitfield is controlled by software. It sets the division factor of the clock sent to the MCO2 output as follows:

0000: 1

0001: 2

0010: 4

...

0111: 128

1000: 256

1001: 512

1010: 1024

Others: reserved

It is highly recommended to set this field before the MCO2 output is enabled.

Bits 19:16 **MCO2SEL[3:0]**: Microcontroller clock output 2 clock selector

This bitfield is controlled by software. It sets the clock selector for MCO2 output as follows:

0000: no clock, MCO2 output disabled

0001: SYSCLK

0010: MSI

0011: HSI16

0100: HSE

0101: PLLRCLK

0110: LSI

0111: LSE

1000: HSI48

1001: RTCCLK

1010: RTC WAKEUP

Others: Reserved

Note: This clock output may have some truncated cycles at startup or during MCO2 clock source switching.

Bit 15 **STOPWUCK**: Wake-up 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 Clock Security System is enabled by HSECSSON in 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 wake-up from stop clock and CSS backup clock.

1: HSI16 oscillator selected as wake-up from stop clock and CSS backup clock

Bits 14:12 **PPRE[2:0]**: APB prescaler

This bitfield is controlled by software. To produce PCLK clock, it sets the division factor of HCLK clock as follows:

0xx: 1

100: 2

101: 4

110: 8

111: 16

Bits 11:8 HPRE[3:0]: AHB prescaler

This bitfield is controlled by software. To produce HCLK clock, it sets the division factor of SYCLK clock as follows:

0xxx: 1
1000: 2
1001: 4
1010: 8
1011: 16
1100: 64
1101: 128
1110: 256
1111: 512

Caution: Depending on the device voltage range, 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 4.1.6: Dynamic voltage scaling management](#)). After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value has been taken into account.

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

Bits 5:3 SWS[2:0]: System clock switch status

This bitfield is controlled by hardware to indicate the clock source used as system clock:

000: MSI
001: HSI16
010: HSE
011: PLLRCLK
100: LSI
101: LSE
Others: Reserved

Bits 2:0 SW[2:0]: System clock switch

This bitfield is controlled by software and hardware. The bitfield selects the clock for SYCLK as follows:

000: MSI
001: HSI16
010: HSE
011: PLLRCLK
100: LSI
101: LSE
Others: Reserved

The setting is forced by hardware to 000 (HSISYS selected) when the MCU exits Stop, Standby, or Shutdown mode, or when the setting is 001 (HSE selected) and HSE oscillator failure is detected.

5.4.4 PLL configuration register (RCC_PLLCFGR)

Address offset: 0x0C

Reset value: 0x0000 1000

This register configures the PLL clock outputs according to the formulas:

- $f_{VCO} = f_{PLLIN} \times (N / M)$
- $f_{PLL_P} = f_{VCO} / P$
- $f_{PLL_Q} = f_{VCO} / Q$
- $f_{PLL_R} = f_{VCO} / R$

where f_{PLLIN} is the PLL input clock frequency, f_{VCO} is the PLL VCO frequency, and P, Q and R are f_{VCO} division factors and f_{PLL_P} , f_{PLL_Q} and f_{PLL_R} the clock frequencies of the PLLPCLK, PLLQCLK and PLLRCLK PLL clock outputs, respectively.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------|-----------|----|------------|-----------|----|----|----|------------|-----------|------|------------|------|------|-------------|------------|
| PLL[2:0] | | | PLL REN | PLLQ[2:0] | | | | PLL QEN | Res. | Res. | PLL_P[4:0] | | | | PLL PEN |
| 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 **PLL_R[2:0]**: PLL VCO division factor R for PLLRCLK clock output

This bitfield is controlled by software. It sets the PLL VCO division factor R as follows:

- 000: Reserved
- 001: 2
- 010: 3
- 011: 4
- 100: 5
- 101: 6
- 110: 7
- 111: 8

The bitfield can be written only when the PLL is disabled.

The PLLRCLK clock can be selected as system clock.

Caution: The software must set this bitfield so as not to exceed 54 MHz on this clock.

Bit 28 **PLLREN**: PLLRCLK clock output enable

This bit is controlled by software to enable/disable the PLLRCLK clock output of the PLL:

- 0: Disable
- 1: Enable

This bit cannot be written when PLLRCLK output of the PLL is selected for system clock.

Disabling the PLLRCLK clock output, when not used, allows saving power.

Bits 27:25 **PLLQ[2:0]**: PLL VCO division factor Q for PLLQCLK clock output

This bitfield is controlled by software. It sets the PLL VCO division factor Q as follows:

- 000: Reserved
- 001: 2
- 010: 3
- 011: 4
- 100: 5
- 101: 6
- 110: 7
- 111: 8

The bitfield can be written only when the PLL is disabled.

Caution: The software must set this bitfield so as not to exceed 54 MHz on this clock.

Bit 24 **PLLQEN**: PLLQCLK clock output enable

This bit is controlled by software to enable/disable the PLLQCLK clock output of the PLL:

- 0: Disable
- 1: Enable

Disabling the PLLQCLK clock output, when not used, allows saving power.

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

Bits 21:17 **PLLP[4:0]**: PLL VCO division factor P for PLLPCLK clock output

This bitfield is controlled by software. It sets the PLL VCO division factor P as follows:

- 00000: Reserved
- 00001: 2
- ...
- 11111: 32

The bitfield can be written only when the PLL is disabled.

Caution: The software must set this bitfield so as not to exceed 122 MHz on this clock.

Bit 16 **PLLPEN**: PLLPCLK clock output enable

This bit is controlled by software to enable/disable the PLLPCLK clock output of the PLL:

- 0: Disable
- 1: Enable

Disabling the PLLPCLK clock output, when not used, allows saving power.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLN[6:0]**: PLL frequency multiplication factor N

This bit is controlled by software to set the division factor of the f_{VCO} feedback divider (that determines the PLL multiplication ratio) as follows:

- 0000000: Invalid
- 0000001: Reserved
- ...
- 0000011: Reserved
- 0000100: 4
- 0000101: 5
- ...
- 1111110: 126
- 1111111: 127

The bitfield can be written only when the PLL is disabled.

Caution: The software must set these bits so 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 M of the PLL input clock divider

This bit is controlled by software to divide the PLL input clock before the actual phase-locked loop, as follows:

- 000: 1
- 001: 2
- 010: 3
- 011: 4
- 100: 5
- 101: 6
- 110: 7
- 111: 8

The bitfield can be written only when the PLL is disabled.

Caution: The software must set these bits so that the PLL input frequency after the /M divider is between 2.66 and 16 MHz.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSRC[1:0]**: PLL input clock source

This bit is controlled by software to select PLL clock source, as follows:

- 00: No clock
- 01: MSI
- 10: HSI16
- 11: HSE

The bitfield can be written only when the PLL is disabled.

When the PLL is not used, selecting 00 allows saving power.

5.4.5 Clock interrupt enable register (RCC_CIER)

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. | HSI48 RDYIE | LSE CSSIE | Res. | Res. | Res. | PLL RDYIE | HSE RDYIE | HSI RDYIE | MSI RDYIE | LSE RDYIE | LSI RDYIE |
| | | | | | rw | rw | | | | rw | rw | rw | rw | rw | rw |

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

Bit 10 **HSI48RDYIE**: HSI48 ready interrupt enable⁽¹⁾

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: Disable

1: Enable

Bit 4 **HSERDYIE**: HSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization:

0: Disable

1: Enable

Bit 3 **HSIRDYIE**: HSI16 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSI16 oscillator stabilization:

0: Disable

1: Enable

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 **LSEIRDYIE**: LSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization:

0: Disable

1: Enable

Bit 0 **LSIRDYIE**: LSI ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSI oscillator stabilization:

0: Disable

1: Enable

- Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.6 Clock interrupt flag register (RCC_CIFR)

Address offset: 0x1C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|---------------|-------------|------|------|------|-------------|-------------|-------------|-------------|-------------|-------------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | HSI48 RDYF | LSE CSSF | CSSF | Res. | Res. | PLL RDYF | HSE RDYF | HSI RDYF | MSI RDYF | LSE RDYF | LSI RDYF |
| | | | | | r | r | r | | | r | r | r | r | r | r |

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

Bit 10 **HSI48RDYF**: HSI48 ready interrupt flag⁽¹⁾

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 by 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 PLLRDYIE 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 HSERDYIE 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 HSIRDYIE is set in a response to setting the HSION (refer to [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 MSIRDYIE 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 LSERDYIE 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 **LSIRDYF**: LSI ready interrupt flag

Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.

Cleared by software setting the LSIRDYC bit.

0: No clock ready interrupt caused by the LSI oscillator

1: Clock ready interrupt caused by the LSI oscillator

- Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.7 Clock interrupt clear register (RCC_CICR)

Address offset: 0x20

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------------|----------|------|------|------|----------|----------|----------|----------|----------|----------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | HSI48 RDYC | LSE CSSC | CSSC | Res. | Res. | PLL RDYC | HSE RDYC | HSI RDYC | MSI RDYC | LSE RDYC | LSI RDYC |
| | | | | | w | w | w | | | w | w | w | w | w | w |

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

Bit 10 **HSI48RDYC**: HSI48 oscillator ready interrupt clear⁽¹⁾

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**: Clock security system interrupt clear

This bit is set by software to clear the HSECSSF flag.

0: No effect

1: Clear 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: Clear LSERDYF flag

Bit 0 **LSIRDYC**: LSI ready interrupt clear

This bit is set by software to clear the LSIRDYF flag.

0: No effect

1: Clear LSIRDYF flag

- Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.8 AHB peripheral reset register (RCC_AHBRSTR)

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. | TSC RST | Res. | Res. | Res. | Res. | Res. | RNG RST | Res. | AES RST |
| | | | | | | | rw | | | | | | rw | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | CRC RST | Res. | Res. | Res. | FLASH RST | Res. | Res. | Res. | Res. | Res. | DMA2 RST | DMA1 RST | |
| | | | rw | | | | rw | | | | | | rw | rw | |

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

Bit 24 **TSCRST**: Touch sensing controller reset

Set and cleared by software.

0: No effect

1: Reset TSC

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

Bit 18 **RNGRST**: Random number generator reset

Set and cleared by software.

0: No effect

1: Reset RNG

Bit 17 Reserved, must be kept at reset value.

Bit 16 **AESRST**: AES hardware accelerator reset⁽¹⁾

Set and cleared by software.

0: No effect

1: Reset AES

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

Bit 12 **CRCRST**: CRC reset

Set and cleared by software.

0: No effect

1: Reset CRC

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

Bit 8 **FLASHRST**: Flash memory interface reset

Set and cleared by software.

0: No effect

1: Reset flash memory interface

This bit can only be set when the flash memory is in power down mode.

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

Bit 1 **DMA2RST**: DMA2 and DMAMUX reset

Set and cleared by software.

0: No effect

1: Reset DMA2 and DMAMUX

Bit 0 **DMA1RST**: DMA1 and DMAMUX reset

Set and cleared by software.

0: No effect

1: Reset DMA1 and DMAMUX

- Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.9 I/O port reset register (RCC_IOPRSTR)

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | GPIOF RST | GPIOE RST | GPIOD RST | GPIOC RST | GPIOB RST | GPIOA RST |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 GPIOFRST: I/O port F reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port F

Bit 4 GPIOERST: I/O port E reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port E

Bit 3 GPIODRST: I/O port D reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port D

Bit 2 GPIOCRST: I/O port C reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port C

Bit 1 GPIOBRST: I/O port B reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port B

Bit 0 GPIOARST: I/O port A reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port A

5.4.10 APB peripheral reset register 1 (RCC_APBRSTR1)

Address offset: 0x38

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------------|------------|----------|--------------|------|------------|----------|-----------|--------------|----------|----------|--------------|-------------|-------------|-------------|----------|
| LPTIM1 RST | LPTIM2 RST | DAC1 RST | PWR RST | Res. | LPTIM3 RST | I2C4 RST | OPAMP RST | I2C3 RST | I2C2 RST | I2C1 RST | LP UART1 RST | USART 4 RST | USART 3 RST | USART 2 RST | CRS RST |
| 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 |
| SPI3 RST | SPI2 RST | USB RST | LP UART3 RST | Res. | Res. | LCD RST | Res. | LP UART2 RST | Res. | TIM7 RST | TIM6 RST | Res. | Res. | TIM3 RST | TIM2 RST |
| rw | rw | rw | rw | | | rw | | rw | | rw | rw | | | rw | rw |

Bit 31 **LPTIM1RST**: Low-power timer 1 reset

Set and cleared by software.

0: No effect

1: Reset LPTIM1

Bit 30 **LPTIM2RST**: Low-power timer 2 reset

Set and cleared by software.

0: No effect

1: Reset LPTIM2

Bit 29 **DAC1RST**: DAC1 interface reset

Set and cleared by software.

0: No effect

1: Reset DAC1 interface

Bit 28 **PWRRST**: Power interface reset

Set and cleared by software.

0: No effect

1: Reset PWR

Bit 27 Reserved, must be kept at reset value.

Bit 26 **LPTIM3RST**: LPTIM3 reset⁽¹⁾

Set and cleared by software.

0: No effect

1: Reset LPTIM3

Bit 25 **I2C4RST**: I2C4 reset⁽¹⁾

Set and cleared by software.

0: No effect

1: Reset I2C4

Bit 24 **OPAMPRST**: OPAMP reset

Set and cleared by software.

0: No effect

1: Reset the OPAMP

- Bit 23 **I2C3RST**: I2C3 reset
Set and cleared by software.
0: No effect
1: Reset I2C3
- Bit 22 **I2C2RST**: I2C2 reset
Set and cleared by software.
0: No effect
1: Reset I2C2
- Bit 21 **I2C1RST**: I2C1 reset
Set and cleared by software.
0: No effect
1: Reset I2C1
- Bit 20 **LPUART1RST**: LPUART1 reset
Set and cleared by software.
0: No effect
1: Reset LPUART1
- Bit 19 **USART4RST**: USART4 reset
Set and cleared by software.
0: No effect
1: Reset USART4
- Bit 18 **USART3RST**: USART3 reset
Set and cleared by software.
0: No effect
1: Reset USART3
- Bit 17 **USART2RST**: USART2 reset
Set and cleared by software.
0: No effect
1: Reset USART2
- Bit 16 **CRSRST**: CRS reset⁽¹⁾
Set and cleared by software.
0: No effect
1: Reset CRS
- Bit 15 **SPI3RST**: SPI3 reset⁽¹⁾
Set and cleared by software.
0: No effect
1: Reset SPI3
- Bit 14 **SPI2RST**: SPI2 reset
Set and cleared by software.
0: No effect
1: Reset SPI2
- Bit 13 **USBRST**: USB reset⁽¹⁾
Set and cleared by software.
0: No effect
1: Reset USB

Bit 12 **LPUART3RST**: LPUART3 reset⁽¹⁾

Set and cleared by software.

0: No effect

1: Reset LPUART3

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **LCDRST**: LCD reset⁽¹⁾

Set and cleared by software.

0: No effect

1: Reset LCD

Bit 8 Reserved, must be kept at reset value.

Bit 7 **LPUART2RST**: LPUART2 reset

Set and cleared by software.

0: No effect

1: Reset LPUART2

Bit 6 Reserved, must be kept at reset value.

Bit 5 **TIM7RST**: TIM7 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM7

Bit 4 **TIM6RST**: TIM6 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM6

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3RST**: TIM3 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM3

Bit 0 **TIM2RST**: TIM2 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM2

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.11 APB peripheral reset register 2 (RCC_APBRSTR2)

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. | ADC RST | Res. | Res. | TIM16 RST | TIM15 RST |
| | | | | | | | | | | | rw | | | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | USART 1 RST | Res. | SPI1 RST | TIM1 RST | Res. | Res. | Res. | SYS CFG RST | |
| | rw | | rw | rw | | | | | | | | | | | rw |

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

Bit 20 **ADCRST**: ADC reset

Set and cleared by software.

0: No effect

1: Reset ADC

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **TIM16RST**: TIM16 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM16 timer

Bit 16 **TIM15RST**: TIM15 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM15 timer

Bit 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 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRST**: SYSCFG, COMP and VREFBUF reset
Set and cleared by software.
0: No effect
1: Reset SYSCFG + COMP + VREFBUF

5.4.12 AHB peripheral clock enable register (RCC_AHBENR)

Address offset: 0x48

Reset value: 0x0000 0100

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|--------|------|------|------|----------|------|------|------|------|------|---------|---------|--------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | TSC EN | Res. | Res. | Res. | Res. | Res. | RNG EN | Res. | AES EN |
| | | | | | | | rw | | | | | | rw | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | CRC EN | Res. | Res. | Res. | FLASH EN | Res. | Res. | Res. | Res. | Res. | DMA2 EN | DMA1 EN | |
| | | | rw | | | | rw | | | | | | rw | rw | |

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

Bit 24 **TSCEN**: Touch sensing controller clock enable
Set and cleared by software.
0: TSC clock disable
1: TSC clock enable

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

Bit 18 **RNGEN**: Random number generator clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 17 Reserved, must be kept at reset value.

Bit 16 **AESEN**: AES hardware accelerator⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable

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

Bit 12 **CRCEN**: CRC clock enable
Set and cleared by software.
0: Disable
1: Enable

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

Bit 8 **FLASHEN**: Flash memory interface clock enable

Set and cleared by software.

0: Disable

1: Enable

This bit can only be cleared when the flash memory is in power down mode.

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

Bit 1 **DMA2EN**: DMA2 and DMAMUX clock enable

Set and cleared by software.

0: Disable

1: Enable

DMAMUX is enabled as long as at least one DMA peripheral is enabled.

Bit 0 **DMA1EN**: DMA1 and DMAMUX clock enable

Set and cleared by software.

0: Disable

1: Enable

DMAMUX is enabled as long as at least one DMA peripheral is enabled.

- Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.13 I/O port clock enable register (RCC_IOPENR)

Address offset: 0x4C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|-------------|-------------|-------------|-------------|-------------|-------------|
| Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | GPIOF EN | GPIOE EN | GPIOD EN | GPIOC EN | GPIOB EN | GPIOA EN |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **GPIOFEN**: I/O port F clock enable

This bit is set and cleared by software.

0: Disable

1: Enable

Bit 4 **GPIOEEN**: I/O port E clock enable⁽¹⁾

This bit is set and cleared by software.

0: Disable

1: Enable

Bit 3 **GPIODEN**: I/O port D clock enable⁽¹⁾

This bit is set and cleared by software.

0: Disable

1: Enable

Bit 2 **GPIOCEN:** I/O port C clock enable

This bit is set and cleared by software.

0: Disable

1: Enable

Bit 1 **GPIOBEN:** I/O port B clock enable

This bit is set and cleared by software.

0: Disable

1: Enable

Bit 0 **GPIOAEN:** I/O port A clock enable

This bit is set and cleared by software.

0: Disable

1: Enable

- Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.14 Debug configuration register (RCC_DBGCFGR)

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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | DBG RST | DBG EN |
| | | | | | | | | | | | | | | rw | rw |

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

Bit 1 **DBG_RST:** Debug support reset

Set and cleared by software.

0: No effect

1: Reset DBG

Bit 0 **DBG_EN:** Debug support clock enable

Set and cleared by software.

0: Disable

1: Enable

5.4.15 APB peripheral clock enable register 1 (RCC_APBENR1)

Address offset: 0x58

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------------|--------------|------------|-------------------|------------|------------------|------------|-------------|-------------------|------------|------------|-------------------|------------------|------------------|------------------|------------|
| LPTIM1 EN | LPTIM2 EN | DAC1 EN | PWR EN | Res. | LPTIM3 EN | I2C4 EN | OPAMP EN | I2C3 EN | I2C2 EN | I2C1 EN | LP UART1 EN | USART 4 EN | USART 3 EN | USART 2 EN | CRS EN |
| 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 |
| SPI3 EN | SPI2 EN | USB EN | LP UART3 EN | WWDG EN | RTC APB EN | LCD EN | Res. | LP UART2 EN | Res. | TIM7 EN | TIM6 EN | Res. | Res. | TIM3 EN | TIM2 EN |
| rw | rw | rw | rw | rw | rw | rw | | rw | | rw | rw | | | rw | rw |

Bit 31 **LPTIM1EN**: LPTIM1 clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 30 **LPTIM2EN**: LPTIM2 clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 29 **DAC1EN**: DAC1 interface clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 28 **PWREN**: Power interface clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 27 Reserved, must be kept at reset value.

Bit 26 **LPTIM3EN**: LPTIM3 clock enable⁽¹⁾

Set and cleared by software.

0: Disable

1: Enable

Bit 25 **I2C4EN**: I2C4EN clock enable⁽¹⁾

Set and cleared by software.

0: Disable

1: Enable

Bit 24 **OPAMPEN**: OPAMP clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 23 **I2C3EN**: I2C3 clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 22 **I2C2EN**: I2C2 clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 21 **I2C1EN**: I2C1 clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 20 **LPUART1EN**: LPUART1 clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 19 **USART4EN**: USART4 clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 18 **USART3EN**: USART3 clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 17 **USART2EN**: USART2 clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 16 **CRSEN**: CRS clock enable⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable

Bit 15 **SPI3EN**: SPI3 clock enable⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable

Bit 14 **SPI2EN**: SPI2 clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 13 **USBEN**: USB clock enable⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable

- Bit 12 **LPUART3EN**: LPUART3 clock enable
Set and cleared by software.
0: Disable
1: Enable
- Bit 11 **WWDGEN**: WWDG clock enable
Set by software to enable the window watchdog clock. Cleared by hardware system reset
0: Disable
1: Enable
This bit can also be set by hardware if the WWDG_SW option bit is 0.
- Bit 10 **RTCAPBEN**: RTC APB clock enable
Set and cleared by software.
0: Disable
1: Enable
- Bit 9 **LCDEN**: LCD clock enable⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable
- Bit 8 Reserved, must be kept at reset value.
- Bit 7 **LPUART2EN**: LPUART2 clock enable
Set and cleared by software.
0: Disable
1: Enable
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **TIM7EN**: TIM7 timer clock enable
Set and cleared by software.
0: Disable
1: Enable
- Bit 4 **TIM6EN**: TIM6 timer clock enable
Set and cleared by software.
0: Disable
1: Enable
- Bits 3:2 Reserved, must be kept at reset value.
- Bit 1 **TIM3EN**: TIM3 timer clock enable
Set and cleared by software.
0: Disable
1: Enable
- Bit 0 **TIM2EN**: TIM2 timer clock enable
Set and cleared by software.
0: Disable
1: Enable

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.16 APB peripheral clock enable register 2(RCC_APBENR2)

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. | ADC EN | Res. | Res. | TIM16 EN | TIM15 EN |
| | | | | | | | | | | | rw | | | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | USART 1 EN | Res. | SPI1 EN | TIM1 EN | Res. | Res. | Res. | SYS CFG EN | |
| | rw | | rw | rw | | | | | | | | | | | rw |

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

Bit 20 **ADCEN**: ADC clock enable

Set and cleared by software.

0: Disable

1: Enable

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **TIM16EN**: TIM16 timer clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 16 **TIM15EN**: TIM15 timer clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: USART1 clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN**: SPI1 clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 11 **TIM1EN**: TIM1 timer clock enable

Set and cleared by software.

0: Disable

1: Enable

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGGEN**: SYSCFG, COMP and VREFBUF clock enable

Set and cleared by software.

0: Disable

1: Enable

5.4.17 AHB peripheral clock enable in Sleep/Stop mode register (RCC_AHBSMENR)

Address offset: 0x68

Reset value: 0x0105 1303

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|----------|------|------|-----------|------------|------|------|------|------|------|-----------|-----------|----------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | TSC SMEN | Res. | Res. | Res. | Res. | Res. | RNG SMEN | Res. | AES SMEN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | CRC SMEN | Res. | Res. | SRAM SMEN | FLASH SMEN | Res. | Res. | Res. | Res. | Res. | DMA2 SMEN | DMA1 SMEN | rw |
| | | | rw | | | rw | rw | | | | | | rw | rw | rw |

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

Bit 24 **TSCSMEN**: TSC clock enable during Sleep and Stop mode

Set and cleared by software.

0: Disable

1: Enable

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

Bit 18 **RNGSMEN**: RNG clock enable during Sleep and Stop mode

Set and cleared by software.

0: Disable

1: Enable

Bit 17 Reserved, must be kept at reset value.

Bit 16 **AESSMEN**: AES hardware accelerator clock enable during Sleep mode⁽¹⁾

Set and cleared by software.

0: Disable

1: Enable

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

Bit 12 **CRCSMEN**: CRC clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAMSMEN**: SRAM clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 8 **FLASHSMEN**: Flash memory interface clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

This bit can be activated only when the flash memory is in power down mode.

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

Bit 1 **DMA2SMEN**: DMA2 and DMAMUX clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Clock to DMAMUX during Sleep mode is enabled as long as the clock in Sleep mode is enabled to at least one DMA peripheral.

Bit 0 **DMA1SMEN**: DMA1 and DMAMUX clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Clock to DMAMUX during Sleep mode is enabled as long as the clock in Sleep mode is enabled to at least one DMA peripheral.

- Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.18 I/O port in Sleep mode clock enable register (RCC_IOPSMENR)

Address offset: 0x6C

Reset value: 0x0000 003F

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|---------------|---------------|---------------|---------------|---------------|---------------|
| Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | GPIOF SMEN | GPIOE SMEN | GPIOD SMEN | GPIOC SMEN | GPIOB SMEN | GPIOA SMEN |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **GPIOFSMEN**: I/O port F clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 4 **GPIOESMEN**: I/O port E clock enable during Sleep mode⁽¹⁾

Set and cleared by software.

0: Disable

1: Enable

Bit 3 **GPIODSMEN**: I/O port D clock enable during Sleep mode⁽¹⁾

Set and cleared by software.

0: Disable

1: Enable

Bit 2 **GPIOCSMEN**: I/O port C clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 1 **GPIOBSMEN**: I/O port B clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 0 **GPIOASMEN**: I/O port A clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

- Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.19 APB peripheral clock enable in Sleep/Stop mode register 1 (RCC_APBSMENR1)

Address offset: 0x78

Reset value: 0b1111 1111 0111 1110 0100 1100 0011 0011

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------------|----------------|--------------|---------------------|--------------|--------------------|--------------|---------------|---------------------|--------------|--------------|---------------------|--------------------|--------------------|--------------------|--------------|
| LPTIM1 SMEN | LPTIM2 SMEN | DAC1 SMEN | PWR SMEN | Res. | LPTIM3 SMEN | I2C4 SMEN | OPAMP SMEN | I2C3 SMEN | I2C2 SMEN | I2C1 SMEN | LP UART1 SMEN | USART 4 SMEN | USART 3 SMEN | USART 2 SMEN | CRS SMEN |
| 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 |
| SPI3 SMEN | SPI2 SMEN | USB SMEN | LP UART3 SMEN | WWDG SMEN | RTC APB SMEN | LCD SMEN | Res. | LP UART2 SMEN | Res. | TIM7 SMEN | TIM6 SMEN | Res. | Res. | TIM3 SMEN | TIM2 SMEN |
| rw | rw | rw | rw | rw | rw | rw | | rw | | rw | rw | | | rw | rw |

- Bit 31 **LPTIM1SMEN**: Low-power timer 1 clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable
- Bit 30 **LPTIM2SMEN**: Low-power timer 2 clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable
- Bit 29 **DAC1SMEN**: DAC1 interface clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable
- Bit 28 **PWRSMEN**: Power interface clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 27 Reserved, must be kept at reset value.
- Bit 26 **LPTIM3SMEN**: Low-power timer 3 clock enable during Sleep mode⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable
- Bit 25 **I2C4SMEN**: I2C4 clock enable during Sleep mode⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable
- Bit 24 **OPAMPSMEN**: OPAMP clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable
- Bit 23 **I2C3SMEN**: I2C3 clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 22 **I2C2SMEN**: I2C2 clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 21 **I2C1SMEN**: I2C1 clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable
- Bit 20 **LPUART1SMEN**: LPUART1 clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable

- Bit 19 **USART4SMEN**: USART4 clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 18 **USART3SMEN**: USART3 clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 17 **USART2SMEN**: USART2 clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable
- Bit 16 **CRSSMEN**: CRS clock enable during Sleep and Stop modes⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable
- Bit 15 **SPI3SMEN**: SPI3 clock enable during Sleep mode⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable
- Bit 14 **SPI2SMEN**: SPI2 clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 13 **USBSMEN**: USB clock enable during Sleep mode⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable
- Bit 12 **LPUART3SMEN**: LPUART3 clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable
- Bit 11 **WWDGSMEN**: WWDG clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable
- Bit 10 **RTCAPBSMEN**: RTC APB clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 9 **LCDSMEN**: LCD clock enable during Sleep mode⁽¹⁾
Set and cleared by software.
0: Disable
1: Enable
- Bit 8 Reserved, must be kept at reset value.

Bit 7 **LPUART2SMEN**: LPUART2 clock enable during Sleep and Stop modes

Set and cleared by software.

0: Disable

1: Enable

Bit 6 Reserved, must be kept at reset value.

Bit 5 **TIM7SMEN**: TIM7 timer clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 4 **TIM6SMEN**: TIM6 timer clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3SMEN**: TIM3 timer clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 0 **TIM2SMEN**: TIM2 timer clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

- Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.20 APB peripheral clock enable in Sleep/Stop mode register 2 (RCC_APBSMENR2)

Address offset: 0x80

Reset value: 0x00017 D801

| | | | | | | | | | | | | | | | |
|------|--------------|------|-----------|-----------|------|------|------|------|------|------|----------|------|------|--------------|------------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | ADC SMEN | Res. | Res. | TIM16 SMEN | TIM15 SMEN |
| | | | | | | | | | | | rw | | | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | USART 1 SMEN | Res. | SPI1 SMEN | TIM1 SMEN | Res. | Res. | Res. | SYS CFG SMEN | |
| | rw | | rw | rw | | | | | | | | | | rw | |

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

Bit 20 **ADCSMEN**: ADC clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bits 19:18 Reserved, must be kept at reset value.

- Bit 17 **TIM16SMEN**: TIM16 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 16 **TIM15SMEN**: TIM15 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **USART1SMEN**: USART1 clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 **SPI1SMEN**: SPI1 clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bit 11 **TIM1SMEN**: TIM1 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
- Bits 10:1 Reserved, must be kept at reset value.
- Bit 0 **SYSCFGSMEN**: SYSCFG, COMP and VREFBUF clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable

5.4.21 Peripherals independent clock configuration register (RCC_CCIPR)

Address offset: 0x88

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|--------------|----|------------------|----|------------------|----------|------------------|----|----------------|------|-----------------|----|-----------------|----|
| Res. | Res. | ADCSEL[1:0] | | CLK48SEL[1:0] | | TIM15 SEL | TIM1 SEL | LPTIM3SEL[1:0] | | LPTIM2SEL[1:0] | | LPTIM1SEL[1:0] | | I2C3SEL[1:0] | |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | I2C1SEL[1:0] | | LPUART1SEL [1:0] | | LPUART2SEL [1:0] | | LPUART3SEL [1:0] | | Res. | Res. | USART2SEL [1:0] | | USART1SEL [1:0] | |
| | | rw | rw | rw | rw | rw | rw | rw | rw | | | rw | rw | rw | rw |

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

Bits 29:28 **ADCSEL[1:0]**: ADCs clock source selection

This bitfield is controlled by software to select the clock source for ADC:

- 00: System clock
- 01: PLLPCLK
- 10: HSI16
- 11: Reserved

Bits 27:26 **CLK48SEL[1:0]**: 48 MHz clock source selection

This bitfield is controlled by software to select the 48 MHz clock source used by the USB FS and the RNG:

- 00: No clock
- 01: MSI
- 10: PLLQCLK
- 11: HSI48⁽¹⁾

Bit 25 **TIM15SEL**: TIM15 clock source selection

This bit is set and cleared by software. It selects TIM15 clock source as follows:

- 0: TIMPCLK
- 1: PLLQCLK

Bit 24 **TIM1SEL**: TIM1 clock source selection

This bit is set and cleared by software. It selects TIM1 clock source as follows:

- 0: TIMPCLK
- 1: PLLQCLK

Bits 23:22 **LPTIM3SEL[1:0]**: LPTIM3 clock source selection⁽¹⁾

This bitfield is controlled by software to select LPTIM3 clock source as follows:

- 00: PCLK
- 01: LSI
- 10: HSI16
- 11: LSE

Bits 21:20 **LPTIM2SEL[1:0]**: LPTIM2 clock source selection

This bitfield is controlled by software to select LPTIM2 clock source as follows:

- 00: PCLK
- 01: LSI
- 10: HSI16
- 11: LSE

Bits 19:18 **LPTIM1SEL[1:0]**: LPTIM1 clock source selection

This bitfield is controlled by software to select LPTIM1 clock source as follows:

- 00: PCLK
- 01: LSI
- 10: HSI16
- 11: LSE

Bits 17:16 **I2C3SEL[1:0]**: I2C3 clock source selection

This bitfield is controlled by software to select I2C3 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: Reserved

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

Bits 13:12 I2C1SEL[1:0]: I2C1 clock source selection

This bitfield is controlled by software to select I2C1 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: Reserved

Bits 11:10 LPUART1SEL[1:0]: LPUART1 clock source selection

This bitfield is controlled by software to select LPUART1 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: LSE

Bits 9:8 LPUART2SEL[1:0]: LPUART2 clock source selection

This bitfield is controlled by software to select LPUART2 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: LSE

Bits 7:6 LPUART3SEL[1:0]: LPUART3 clock source selection⁽¹⁾

This bitfield is controlled by software to select LPUART3 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: LSE

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:2 USART2SEL[1:0]: USART2 clock source selection

This bitfield is controlled by software to select USART2 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: LSE

Bits 1:0 USART1SEL[1:0]: USART1 clock source selection

This bitfield is controlled by software to select USART1 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: LSE

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.22 RTC domain control register (RCC_BDCR)

Up to three wait states are inserted in case of successive accesses to this register. As this register is outside of the V_{CORE} domain, it is write-protected upon reset. The DBP bit of the [Power control register 1 \(PWR_CR1\)](#) must be set to allow their modification. Refer to [Section 4.1.4: Battery backup domain on page 108](#) for further information.

The register bits are only reset upon RTC domain reset (see [Section 5.1.3: RTC domain reset](#)), except the LSCOSEL, LSCOEN, and BDRST bits that are only reset upon RTC domain power-on reset. Any internal or external reset has no effect on these bits.

Address offset: 0x90

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. | 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 |
| RTSEN | Res. | Res. | Res. | LSE SYS RDY | Res. | RTCSEL[1:0] | | LSE SYSEN | LSE CSSD | LSE CSS ON | LSEDRV[1:0] | | LSE BYP | LSE RDY | LSEON |
| rw | | | | r | | 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 to select the low-speed output clock:

- 0: LSI
- 1: LSE

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable

Set and cleared by software.

- 0: Disable
- 1: Enable

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **BDRST**: RTC domain software reset

Set and cleared by software to reset the RTC domain:

- 0: No effect
- 1: Reset

Bit 15 **RTSEN**: RTC clock enable

Set and cleared by software. The bit enables clock to RTC and TAMP.

- 0: Disable
- 1: Enable

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

Bit 11 **LSESYRDY**: LSE clock ready for system usage

This flag is set by hardware to indicate that the LSE clock is ready for being used by the system (see LSESYSEN bit). This flag is set when LSE clock is ready (LSEON = 1 and LSERDY = 1) and two LSE clock cycles after that LSESYSEN is set.

Cleared by hardware to indicate that the LSE clock is not ready to be used by the system.

- 0: LSE clock not ready for system
- 1: LSE clock ready for system

Bit 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 as follows:

00: No clock

01: LSE

10: LSI

11: HSE divided by 32

Once the RTC clock source is selected, it cannot be changed anymore unless the RTC domain is reset, or unless a failure is detected on LSE (LSECSSD is set). The BDRST bit can be used to reset this bitfield to 00.

Bit 7 **LSESYSEN**: LSE clock enable for system usage

This bit must be set by software to enable the LSE clock for a system usage.

0: Disabled

1: Enabled, LSE distributed to peripherals including LSCO/MCO/SYSCLK.

Bit 6 **LSECSSD**: CSS on LSE failure Detection

Set by hardware to indicate when a failure is detected by the clock security system on the external 32 kHz oscillator (LSE):

0: No failure detected

1: Failure detected

Bit 5 **LSECSSON**: CSS on LSE enable

Set by software to enable the clock security system on LSE (32 kHz) oscillator as follows:

0: Disable

1: Enable

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.

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator drive capability

Set by software to select the LSE oscillator drive capability as follows:

00: low driving capability

01: medium-low driving capability

10: medium-high driving capability

11: high driving capability

Applicable when the LSE oscillator is in Xtal mode, as opposed to bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and cleared by software to bypass the LSE oscillator (in debug mode).

0: Not bypassed

1: Bypassed

This bit can be written only when the external 32 kHz oscillator is disabled (LSEON=0 and LSERDY=0).

Bit 1 **LSERDY**: LSE oscillator ready

Set and cleared by hardware to indicate when the external 32 kHz oscillator is ready (stable):

0: Not ready

1: Ready

After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.

Bit 0 **LSEON**: LSE oscillator enable

Set and cleared by software to enable LSE oscillator:

0: Disable

1: Enable

5.4.23 Control/status register (RCC_CSR)

Up to three wait states are inserted in case of successive accesses to this register. The register is reset upon system reset, except for reset flags that are only reset upon power reset.

Address offset: 0x94

Reset value: 0xXX00 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------------|--------------|--------------|-------------|----------------|-------------|-------------|------|------|------|------|------|------|-------------------|------------|-------|
| LPWR RSTF | WWDG RSTF | IWDG RSTF | SFT RSTF | PWR RSTF | PIN RSTF | OBL RSTF | Res. | RMVF | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| r | r | r | r | r | r | r | | rw | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | MSISRANGE[3:0] | | | | Res. | Res. | Res. | Res. | Res. | LSI PRE DIV | LSI RDY | LSION |
| | | | | rw | rw | rw | rw | | | | | | 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 setting the RMVF bit.

0: No illegal mode reset occurred
1: Illegal mode reset occurred

This operates only if nRST_STOP, nRST_STDBY or nRST_SHDW option bits are cleared.

Bit 30 WWDGRSTF: Window watchdog reset flag

Set by hardware when a window watchdog reset occurs.
Cleared by setting 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 setting the RMVF bit.

0: No independent watchdog reset occurred
1: Independent watchdog reset occurred

Bit 28 SFRSTF: Software reset flag

Set by hardware when a software reset occurs.
Cleared by setting the RMVF bit.

0: No software reset occurred
1: Software reset occurred

Bit 27 PWRRSTF: BOR or POR/PDR flag

Set by hardware when a BOR or POR/PDR occurs.
Cleared by setting the RMVF bit.

0: No BOR or POR occurred
1: BOR or POR occurred

Bit 26 PINRSTF: Pin reset flag

Set by hardware when a reset from the NRST pin occurs.
Cleared by setting 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 setting 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 flags

Set by software to clear the reset flags.
0: No effect
1: Clear reset flags

Bits 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MSISRANGE[3:0]**: MSI range after Standby mode

Set by software to chose the MSI frequency at startup. This range is used after exiting Standby mode until MSIRGSEL is set. After a pad or a power-on reset, the range is always 4 MHz. MSISRANGE[3:0] can be written only when MSIRGSEL = 1.

0100: Range 4 around 1 MHz

0100: Range 5 around 2 MHz

0100: Range 6 around 4 MHz

0100: Range 7 around 8 MHz

Others: Reserved

Note: Changing the MSISRANGE[3:0] does not change the current MSI frequency.

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

Bit 2 **LSIPREDIV**: Internal low-speed oscillator pre-divided by 128

Set and reset by hardware to indicate when the low-speed internal RC oscillator has to be divided by 128. The software has to switch off the LSI before changing this bit.

0: LSI RC oscillator is not divided

1: LSI RC oscillator is divided by 128

Bit 1 **LSIRDY**: LSI oscillator ready

Set and cleared by hardware to indicate when the LSI oscillator is ready (stable):

0: Not ready

1: Ready

After the LSION bit is cleared, LSIRDY goes low after 3 LSI oscillator clock cycles. This bit can be set even if LSION = 0 if the LSI is requested by the Clock Security System on LSE, by the Independent Watchdog or by the RTC.

Bit 0 **LSION**: LSI oscillator enable

Set and cleared by software to enable/disable the LSI oscillator:

0: Disable

1: Enable

5.4.24 RCC clock recovery RC register (RCC_CRRCR)

This register applies to STM32U073xx and STM32U083xx devices only. It is reserved otherwise.

Address offset: 0x98

Reset value: 0b0000 0000 0000 0000 1000 1000 0000 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 |
| HSI48CAL[8:0] | | | | | | | | | | 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⁽¹⁾

These bits are initialized at startup with the factory-programmed HSI48 calibration trim value.

Bits 6:2 Reserved, must be kept at reset value.

Bit 1 **HSI48RDY**: HSI48 clock ready flag⁽¹⁾

The flag is set when the HSI48 clock is ready for use.

Bit 0 **HSI48ON**: HSI48 RC oscillator enable⁽¹⁾

0: Disable

1: Enable

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

5.4.25 RCC register map

The following table gives the RCC register map and the reset values.

Table 33. RCC register map and reset values

Table 33. RCC register map and reset values (continued)

| Offset | Register | Reset value | 31 |
|-------------|--------------|-------------|-------------|
| 0x24 | Reserved | | Res. |
| 0x28 | RCC_AHBRSTR | | Res. |
| 0x2C | RCC_IOPRSTR | | Res. |
| 0x30 - 0x34 | Reserved | | Res. |
| 0x38 | RCC_APBRSTR1 | | 0 LPTIM1RST |
| 0x3C | Reserved | | 0 LPTIM2RST |
| 0x40 | RCC_APBRSTR2 | | 0 DAC1RST |
| 0x44 | Reserved | | 0 PWRRST |
| 0x48 | RCC_AHBENR | Res. | 0 Res. |
| 0x4C | RCC_IOPENR | Res. | 0 Res. |
| 0x50 | RCC_DBGCFGR | Res. | 0 Res. |
| 0x54 | RCC_APBENR1 | 0 LPTIM1EN | 0 Res. |
| 0x58 | Reset value | 0 LPTIM2EN | 0 Res. |
| 0x5C | Reserved | 0 DAC1EN | 0 Res. |
| | | 0 PWREN | 0 Res. |
| | | 0 Res. | 0 Res. |
| | | 0 LPTIM3EN | 0 Res. |
| | | 0 I2C4EN | 0 Res. |
| | | 0 OPAMPEN | 0 Res. |
| | | 0 TSCEN | 0 Res. |
| | | 0 I2C3EN | 0 Res. |
| | | 0 I2C2EN | 0 Res. |
| | | 0 I2C1EN | 0 Res. |
| | | 0 LPUART1EN | 0 Res. |
| | | 0 USART4EN | 0 Res. |
| | | 0 USART3EN | 0 Res. |
| | | 0 USART2EN | 0 Res. |
| | | 0 CRSEN | 0 AESEN |
| | | 0 SPI3EN | 0 Res. |
| | | 0 SPI2EN | 0 Res. |
| | | 0 USBEN | 0 Res. |
| | | 0 LPUART3EN | 0 CRCEN |
| | | 0 WWDGEN | 0 Res. |
| | | 0 RTCAPBEN | 0 Res. |
| | | 0 LCDEN | 0 Res. |
| | | 0 LPUART2EN | 0 FLASHEN |
| | | 0 Res. | 1 Res. |
| | | 0 TIM7EN | 0 GPOFFEN |
| | | 0 TIM6EN | 0 GPIOEEN |
| | | 0 Res. | 0 GPODEN |
| | | 0 Res. | 0 GPIOCN |
| | | 0 DBGRST | 0 GPIOBEN |
| | | 0 TIM3EN | 0 DMA2EN |
| | | 0 TIM2EN | 0 GPIOAEN |
| | | 0 Res. | 0 DMA1EN |
| | | 0 Res. | 0 SYSCFGRST |
| | | 0 DBGEN | 0 TIM2RST |
| | | 0 Res. | 0 DMA2RST |
| | | 0 Res. | 0 DMA1RST |

Table 33. RCC register map and reset values (continued)

| Offset | Register | Reset value |
|-------------|---------------|-------------|
| 0x60 | RCC_APBENR2 | |
| | Reset value | |
| 0x64 | Reserved | |
| | | |
| 0x68 | RCC_AHBSMENR | |
| | Reset value | |
| 0x6C | RCC_IOPSMENR | |
| | Reset value | |
| 0x70 - 0x74 | Reserved | |
| | | |
| 0x78 | RCC_APBSMENR1 | |
| | Reset value | |
| 0x7C | Reserved | |
| | | |
| 0x80 | RCC_APBSMENR2 | |
| | Reset value | |
| 0x84 | Reserved | |
| | | |
| 0x88 | RCC_CCIPR | |
| | Reset value | |
| 0x8C | Reserved | |
| | | |
| 0x90 | RCC_BDCR | |
| | Reset value | |
| 0x94 | RCC_CSR | |
| | Reset value | |

Table 33. RCC register map and reset values (continued)

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

6 Clock recovery system (CRS)

6.1 CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from the LSE oscillator output, or generated by user software.

6.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
 - External pin
 - LSE oscillator output
 - USB SOF packet reception
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
 - Expected synchronization (ESYNC)
 - Synchronization OK (SYNCOK)
 - Synchronization warning (SYNCWARN)
 - Synchronization or trimming error (ERR)

6.3 CRS implementation

Table 34. CRS features

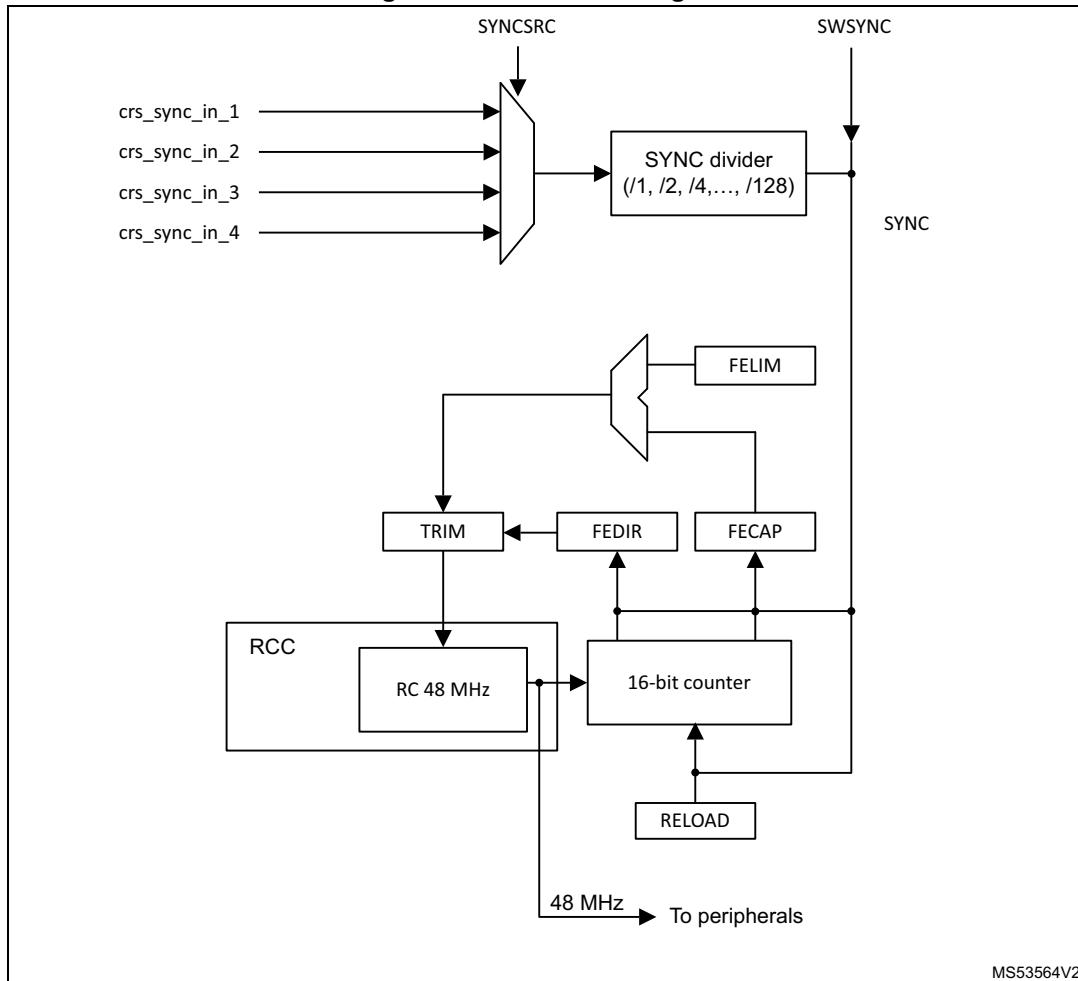
| Feature | CRS |
|------------|--------|
| TRIM width | 7 bits |

Table 35. CRS internal input/output signals

| Internal signal name | Signal type | Description |
|----------------------|-------------|--|
| crs_sync_in_1 | Input | 00: GPIO AF selected as SYNC signal source |
| crs_sync_in_2 | Input | 01: LSE selected as SYNC signal source |
| crs_sync_in_3 | Input | 10: USB SOF selected as SYNC signal source (default) |
| crs_sync_in_4 | Input | 11: Reserved |

6.4 CRS functional description

6.4.1 CRS block diagram

Figure 14. CRS block diagram

MS53564V2

6.4.2 Synchronization input

For more information on the CRS synchronization source configuration, refer to [Section 6.7.2](#).

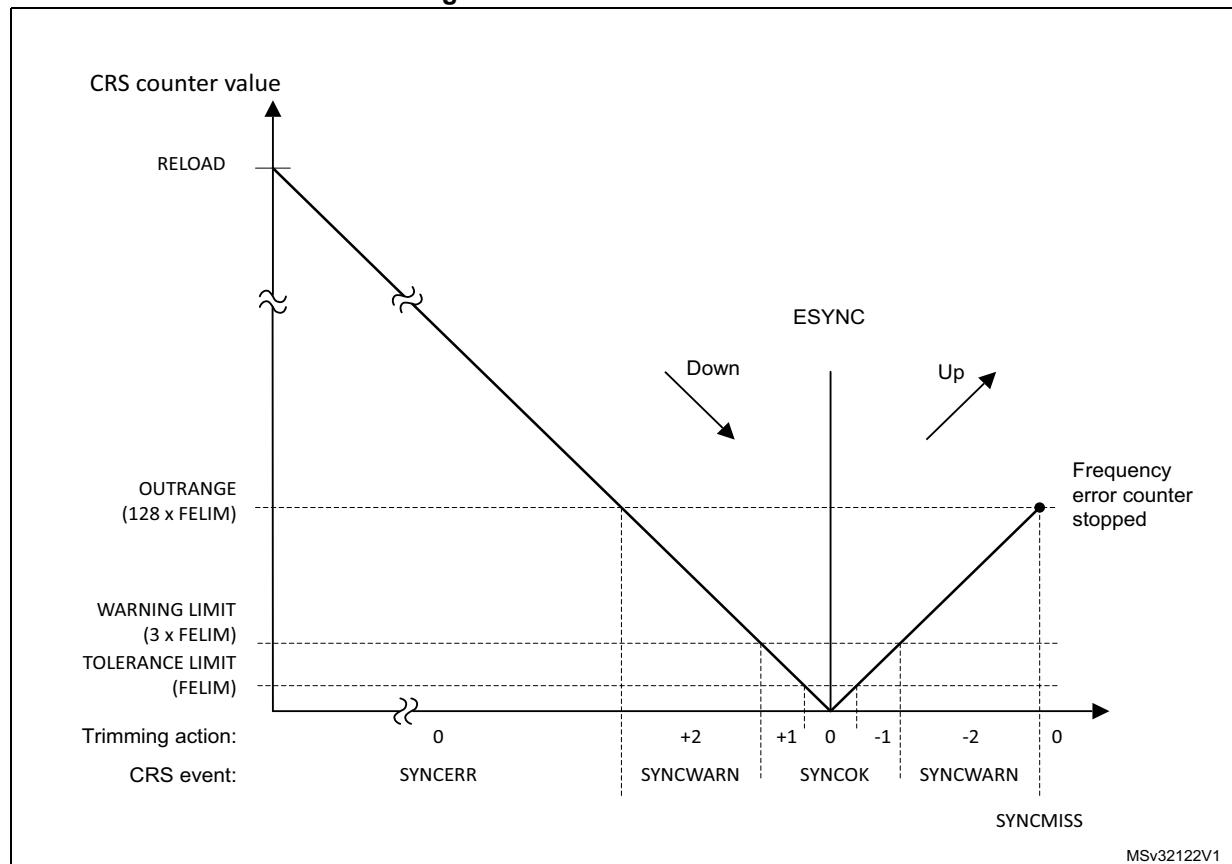
It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

6.4.3 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFG register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).

Figure 15. CRS counter behavior



6.4.4 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFG register
- WARNING LIMIT, defined as $3 \times$ FELIM value
- OUTRANGE (error limit), defined as $128 \times$ FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
 - SYNCOK status indicated
 - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
 - SYNCOK status indicated
 - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
 - SYNCWARN status indicated
 - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
 - SYNCERR or SYNCMISS status indicated
 - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

6.4.5 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

$$\text{RELOAD} = (f_{\text{TARGET}} / f_{\text{SYNC}}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

$$\text{FELIM} = (\text{f}_{\text{TARGET}} / \text{f}_{\text{SYNC}}) * \text{STEP}[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to $(\text{f}_{\text{TARGET}} / \text{f}_{\text{SYNC}}) = 48000$, and to a typical trimming step size of 0.14%.

Note: *The trimming step size depends upon the product, check the datasheet for accurate setting.*

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than $128 * \text{FELIM}$ value (OUTRANGE limit).

6.5 CRS in low-power modes

Table 36. Effect of low-power modes on CRS

| Mode | Description |
|----------|--|
| Sleep | No effect. CRS interrupts cause the device to exit the Sleep mode. |
| Stop | CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted. |
| Standby | The peripheral is powered down and must be reinitialized after exiting Standby mode. |
| Shutdown | The peripheral is powered down and must be reinitialized after exiting Shutdown mode. |

6.6 CRS interrupts

Table 37. Interrupt control bits

| Interrupt event | Event flag | Enable control bit | Clear flag bit |
|---|------------|--------------------|----------------|
| Expected synchronization | ESYNCF | ESYNCIE | ESYNCC |
| Synchronization OK | SYNCOKF | SYNCOKIE | SYNCOKC |
| Synchronization warning | SYNCWARNF | SYNCWARNIE | SYNCWARNC |
| Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR) | ERRF | ERRIE | ERRC |

6.7 CRS registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

6.7.1 CRS control register (CRS_CR)

Address offset: 0x000

Reset value: 0x0000 4000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
|------|-----------|------|------|------|------|------|------|-------|---------|-------------|------|------|----------|--------|-------------|-----------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Res. | TRIM[6:0] | | | | | | | | SW SYNC | AUTO TRIMEN | CEN | Res. | ESYNC IE | ERR IE | SYNC WARNIE | SYNC OKIE |
| | rw | rw | rw | rw | rw | rw | rw | rt_w1 | rw | rw | | rw | rw | rw | rw | rw |

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

Bits 14:8 **TRIM[6:0]**: HSI48 oscillator smooth trimming

The default value of the HSI48 oscillator smooth trimming is 64, which corresponds to the middle of the trimming interval.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to [Section 6.4.4](#) for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFG register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable

0: Synchronization or trimming error (ERRF) interrupt disabled

1: Synchronization or trimming error (ERRF) interrupt enabled

- Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable
 0: SYNC warning (SYNCWARNF) interrupt disabled
 1: SYNC warning (SYNCWARNF) interrupt enabled
- Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable
 0: SYNC event OK (SYNCOKF) interrupt disabled
 1: SYNC event OK (SYNCOKF) interrupt enabled

6.7.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------------|------|--------------|----|------|--------------|----|----|------------|----|----|----|----|----|----|----|
| SYNCPOL | Res. | SYNCSRC[1:0] | | Res. | SYNCDIV[2:0] | | | FELIM[7:0] | | | | | | | |
| 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 |
| RELOAD[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.
 0: SYNC active on rising edge (default)
 1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source (see [Table 35](#)):
 00: crs_sync_in_1 selected as SYNC signal source
 01: crs_sync_in_2 selected as SYNC signal source
 10: crs_sync_in_3 selected as SYNC signal source
 11: crs_sync_in_4 selected as SYNC signal source

Note: When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the HSI48 oscillator on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE or reference clock on the GPIOs must be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.
 000: SYNC not divided (default)
 001: SYNC divided by 2
 010: SYNC divided by 4
 011: SYNC divided by 8
 100: SYNC divided by 16
 101: SYNC divided by 32
 110: SYNC divided by 64
 111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to [Section 6.4.4](#) for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to [Section 6.4.3](#) for more details about counter behavior.

6.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------------|------|------|------|------|----------|-----------|----------|------|------|------|------|--------|------|------------|----------|
| FECAP[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 |
| FEDIR | Res. | Res. | Res. | Res. | TRIM OVF | SYNC MISS | SYNC ERR | Res. | Res. | Res. | Res. | ESYNCF | ERRF | SYNC WARNF | SYNC OKF |
| r | | | | | r | r | r | | | | | r | r | r | r |

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to [Section 6.4.4](#) for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.

- 0: Up-counting direction, the actual frequency is above the target
- 1: Down-counting direction, the actual frequency is below the target

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

- 0: No trimming error signaled
- 1: Trimming error signaled

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

- 0: No SYNC missed error signaled
- 1: SYNC missed error signaled

Bit 8 SYNCERR: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

- 0: No SYNC error signaled
- 1: SYNC error signaled

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

Bit 3 ESYNCF: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.

- 0: No expected SYNC signaled
- 1: Expected SYNC signaled

Bit 2 ERRF: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

- 0: No synchronization or trimming error signaled
- 1: Synchronization or trimming error signaled

Bit 1 SYNCWARNF: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.

- 0: No SYNC warning signaled
- 1: SYNC warning signaled

Bit 0 SYNCOKF: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.

- 0: No SYNC event OK signaled
- 1: SYNC event OK signaled

6.7.4 CRS interrupt flag clear register (CRS_ICR)

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | ESYNCC | ERRC | SYNC WARNC | SYNC OKC |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bit 3 **ESYNCC**: Expected SYNC clear flag

Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag

Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag

Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag

Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

6.7.5 CRS register map

Table 38. CRS 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 |
|--------|------------------------------|---------|------|----------------|------|----------------|------|------------|------|------|------|------|------|------|------|------|----|
| 0x00 | CRS_CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| | Reset value | | | | | | | | | | | | | | | | |
| 0x04 | CRS_CFGR | SYNCPOL | Res. | SYNC SRC [1:0] | Res. | SYNC DIV [2:0] | | FELIM[7:0] | | | | | | | | | |
| | Reset value | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | |
| 0x08 | CRS_ISR | | | | | | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x0C | CRS_ICR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| | Reset value | | | | | | | | | | | | | | | | |

Refer to [Section 2.2](#) for the register boundary addresses.

7 General-purpose I/Os (GPIO)

7.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIO_x_MODER, GPIO_x_OTYPER, GPIO_x_OSPEEDR and GPIO_x_PUPDR), two 32-bit data registers (GPIO_x_IDR and GPIO_x_ODR) and a 32-bit set/reset register (GPIO_x_BSRR). In addition, all GPIOs have a 32-bit locking register (GPIO_x_LCKR) and two 32-bit alternate function selection registers (GPIO_x_AFRH and GPIO_x_AFRL).

7.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIO_x_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 (GPIO_x_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIO_x_BSRR) for bitwise write access to GPIO_x_ODR
- Locking mechanism (GPIO_x_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

7.3 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 GPIO_x_BSRR and GPIO_x_BRR registers is to allow atomic read/modify accesses to any of the GPIO_x_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 16 and *Figure 17* show the basic structures of a standard and a 5-Volt tolerant I/O port bit, respectively. *Table 39* gives the possible port bit configurations.

Figure 16. Basic structure of an I/O port bit

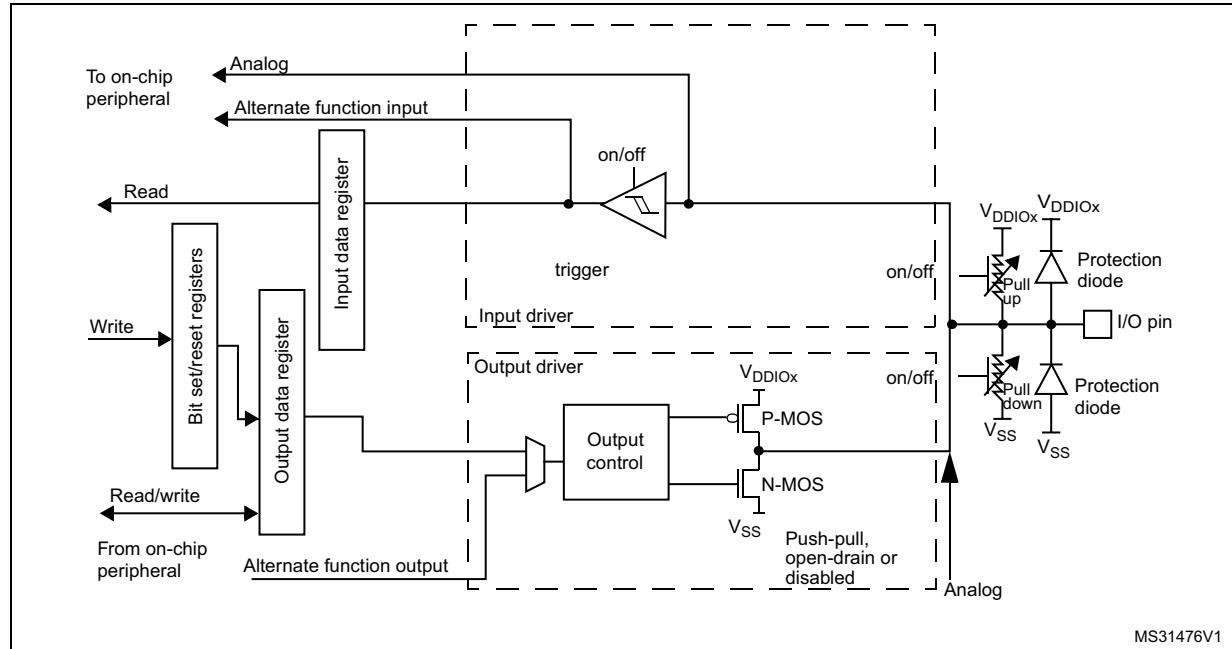
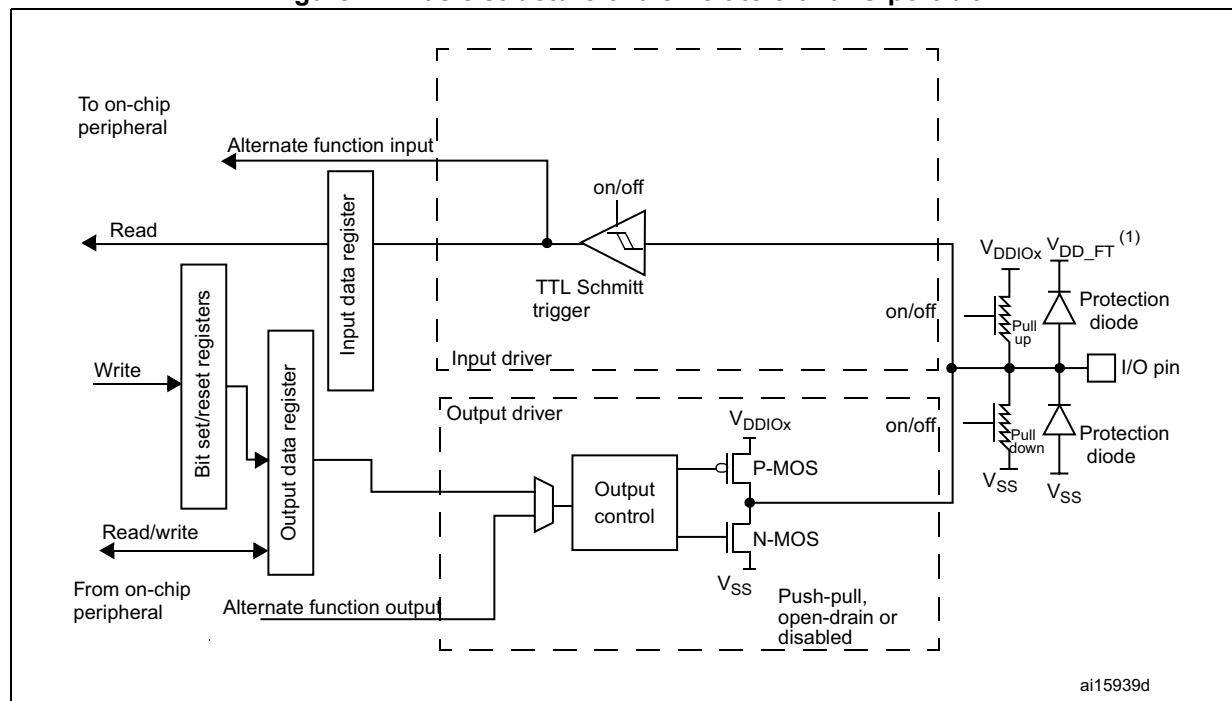


Figure 17. Basic structure of a 5-Volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD} .

Table 39. 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 |
| | x | x | x | 0 | 1 | Input |
| | x | x | x | 1 | 0 | Input |
| | x | x | x | 1 | 1 | Reserved (input floating) |
| 11 | x | x | x | 0 | 0 | Input/output |
| | 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.

7.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA14: SWCLK in pull-down.
- PA13: SWDIO in pull-up.

PF3/BOOT0 is in input mode during the reset until at least the end of the option byte loading phase. See [Section 7.3.15: Using PF3 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.

7.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin (except PF3) has a multiplexer with up to 16 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 the 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, DAC, OPAMP and COMP, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC, DAC, OPAMP, 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.

7.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIO_x_MODER, GPIO_x_OTYPER, GPIO_x_OSPEEDR, GPIO_x_PUPDR) to configure up to 16 I/Os. The GPIO_x_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIO_x_OTYPER and GPIO_x_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIO_x_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

7.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIO_x_IDR and GPIO_x_ODR). GPIO_x_ODR stores the data to be output, it is read/write accessible. The data input through the I/O is stored into the input data register (GPIO_x_IDR), a read-only register.

See [Section 7.4.5: GPIO port input data register \(GPIO_x_IDR\) \(x = A to F\)](#) and [Section 7.4.6: GPIO port output data register \(GPIO_x_ODR\) \(x = A to F\)](#) for the register descriptions.

7.3.5 I/O data bitwise handling

The bit set reset register (GPIO_x_BSRR) is a 32-bit register, which allows the application to set and reset each individual bit in the output data register (GPIO_x_ODR). The bit set reset register has twice the size of GPIO_x_ODR.

To each bit in GPIO_x_ODR, correspond two control bits in GPIO_x_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 GPIO_x_BSRR does not have an effect on the corresponding bit in GPIO_x_ODR. If there is an attempt to both set and reset a bit in GPIO_x_BSRR, the set action takes priority.

Using the GPIO_x_BSRR register to change the values of individual bits in GPIO_x_ODR is a “one-shot” effect that does not lock the GPIO_x_ODR bits. The GPIO_x_ODR bits can always be accessed directly. The GPIO_x_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIO_x_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

7.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIO_x_LCKR register. The frozen registers are GPIO_x_MODER, GPIO_x_OTYPER, GPIO_x_OSPEEDR, GPIO_x_PUPDR, GPIO_x_AFRL and GPIO_x_AFRH.

To write the GPIO_x_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 GPIO_x_LCKR bit freezes the corresponding bit in the control registers (GPIO_x_MODER, GPIO_x_OTYPER, GPIO_x_OSPEEDR, GPIO_x_PUPDR, GPIO_x_AFRL and GPIO_x_AFRH).

The LOCK sequence (refer to [Section 7.4.8: GPIO port configuration lock register \(GPIO_x_LCKR\) \(x = A to F\)](#)) can only be performed using a word (32-bit long) access to the GPIO_x_LCKR register due to the fact that GPIO_x_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details, refer to the LCKR register description in [Section 7.4.8: GPIO port configuration lock register \(GPIO_x_LCKR\) \(x = A to F\)](#).

7.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, 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 GPIO_x_AFRL and GPIO_x_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.

No alternate function is mapped on PF3.

7.3.8 External interrupt/wake-up lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output, or alternate function mode (the port must not be configured in analog mode).

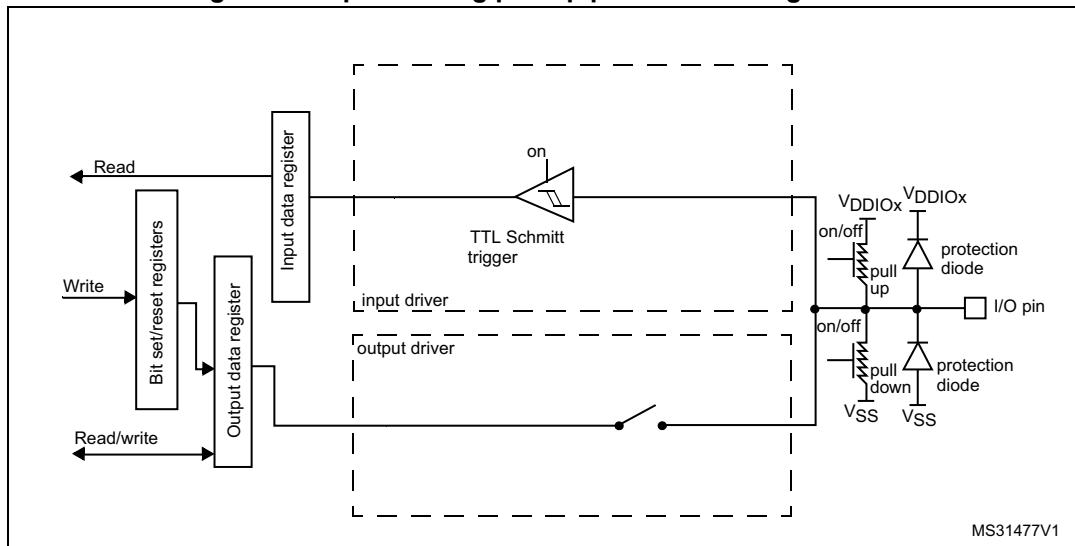
Refer to [Section 12: Extended interrupt and event controller \(EXTI\)](#) and to [Section 12.3.2: EXTI direct event input wake-up](#).

7.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIO_x_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 18](#) shows the input configuration of the I/O port bit.

Figure 18. Input floating/pull up/pull down configurations

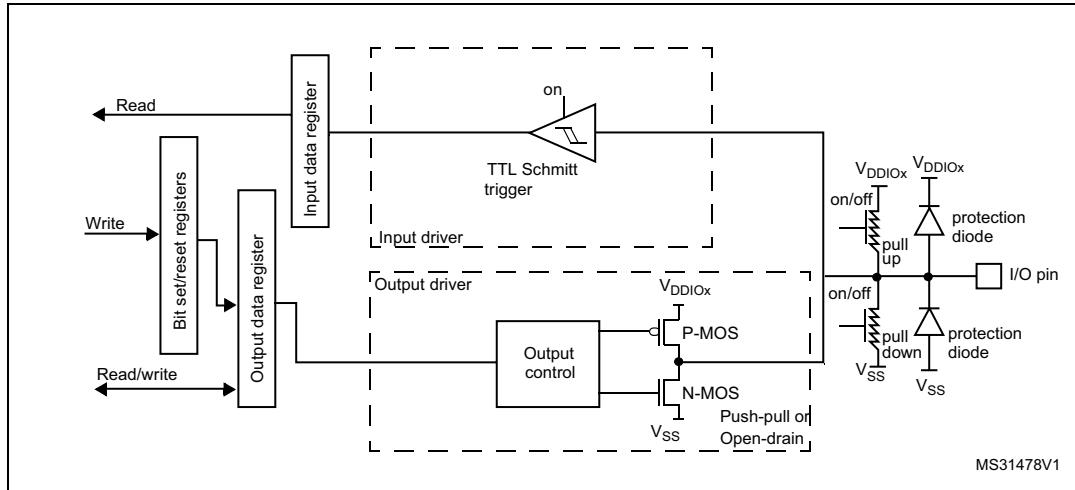
7.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
 - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in high-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 19 shows the output configuration of the I/O port bit.

Figure 19. Output configuration



7.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

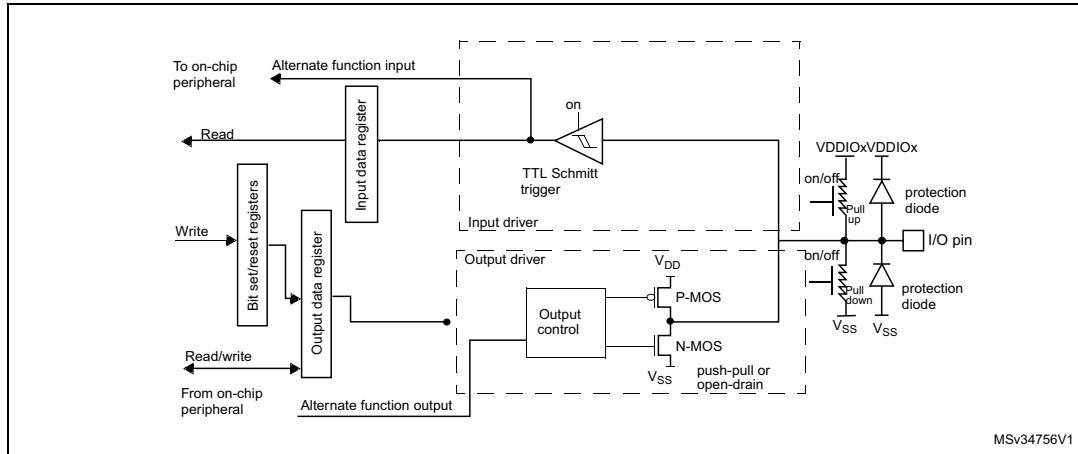
- The output buffer can be configured in open-drain or push-pull mode.
- The signals coming from the peripheral (transmitter enable and data) drive the output buffer.
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the `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.

Note:

The alternate function configuration described above is not applied when the selected alternate function is an LCD function or a SWPML_IO. In this case, the I/O, programmed as an alternate function output, is configured as described in the analog configuration.

Figure 20 shows the alternate function configuration of the I/O port bit.

Figure 20. Alternate function configuration



MSv34756V1

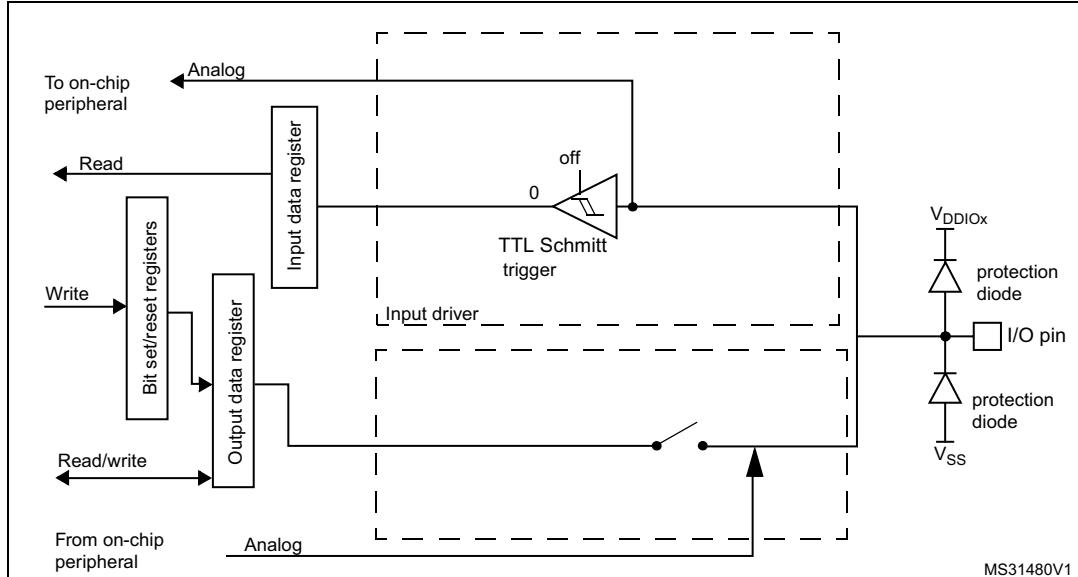
7.3.12 Analog configuration

When the I/O port is programmed as an 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).
- Hardware disables the weak pull-up and pull-down resistors.
- Read access to the input data register gets the value “0”.

Figure 21 shows the high-Z, analog-input configuration of the I/O port bits.

Figure 21. High impedance-analog configuration



MS31480V1

7.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the HSEON or 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 user external clock mode, only the pin is reserved for clock input. The OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

7.3.14 Using the GPIO pins in the RTC supply domain

The PC13/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 30.3: RTC functional description](#).

7.3.15 Using PF3 as GPIO

PF3 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.

7.4 GPIO registers

For a summary of register bits, register address offsets and reset values, refer to [Table 40](#).

The peripheral registers can be written in word, half word, or byte mode.

7.4.1 GPIO port mode register (GPIOx_MODER) (x = A to F)

Address offset: 0x00

Reset value: 0xEBFF FFFF (for port A)

Reset value: 0xFFFF FFFF (for other ports)

| 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 **MODEy[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)

7.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to F)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

7.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to F)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 0000 (for the other ports)

| 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 |
| 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 |

Bits 31:0 **OSPEEDy[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: High speed

11: Very high speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed..

7.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to F)

Address offset: 0x0C

Reset value: 0x2400 0000 (for port A)

Reset value: 0x0000 0000 (for other ports)

| 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 **PUPDy[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

7.4.5 GPIO port input data register (GPIOx_IDR) (x = A to F)

Address offset: 0x10

Reset value: 0x0000 XXXX

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

7.4.6 GPIO port output data register (GPIOx_ODR) (x = A to F)

Address offset: 0x14

Reset value: 0x0000 0000

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

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).

7.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to F)

Address offset: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| BR15 | BR14 | BR13 | BR12 | BR11 | BR10 | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BS15 | BS14 | BS13 | BS12 | BS11 | BS10 | BS9 | BS8 | BS7 | BS6 | BS5 | BS4 | BS3 | BS2 | BS1 | BS0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:16 **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

7.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to F)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the

LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|-------|-------|-------|-------|-------|------|------|------|------|------|------|------|------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | LCKK |
| | | | | | | | | | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LCK15 | LCK14 | LCK13 | LCK12 | LCK11 | LCK10 | LCK9 | LCK8 | LCK7 | LCK6 | LCK5 | LCK4 | LCK3 | LCK2 | LCK1 | LCK0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = 1 + LCKR[15:0]

WR LCKR[16] = 0 + LCKR[15:0]

WR LCKR[16] = 1 + LCKR[15:0]

RD LCKR

RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **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

7.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to F)

Address offset: 0x20

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------------|----|----|----|-------------|----|----|----|-------------|----|----|----|-------------|----|----|----|
| AFSEL7[3:0] | | | | AFSEL6[3:0] | | | | AFSEL5[3:0] | | | | AFSEL4[3:0] | | | |
| rw | rw | rw | rw |
| 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 |

Bits 31:0 **AFSELy[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

7.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to F)

Address offset: 0x24

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------------|----|----|----|--------------|----|----|----|--------------|----|----|----|--------------|----|----|----|
| AFSEL15[3:0] | | | | AFSEL14[3:0] | | | | AFSEL13[3:0] | | | | AFSEL12[3:0] | | | |
| rw | rw | rw | rw |
| 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 |

Bits 31:0 **AFSEL_y[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

7.4.11 GPIO port bit reset register (GPIO_x_BRR) (x = A to F)

Address offset: 0x28

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 OD_x bit

1: Reset the corresponding OD_x bit

7.4.12 GPIO register map

The following table gives the GPIO register map and reset values.

Table 40. GPIO 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 | GPIOx_MODER | | 1 | 1 | 1 | 1 | 0 | 1 | 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 | 1 | 1 | 1 | | |
| | Reset value (port A) | | Res. | |
| | Reset value (port B to F) | | 1 | 1 | 1 | 1 | 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 | 0 | 0 | 0 | 0 | |
| 0x04 | GPIOx_OTYPER (where x = A to F) | | Res. | | | |
| | Reset value | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| | Reset value (port A) | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | GPIOx_OSPEEDR (where x = A to F) | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Reset value (port A) | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Reset value (port B) | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | GPIOx_PUPDR (where x = A to F) | | Res. | | |
| | Reset value (Port A) | | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Reset value (Port B to F) | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | GPIOx_IDR (where x = A to F) | | Res. | | |
| | Reset value | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x14 | GPIOx_ODR (where x = A to F) | | Res. | | |
| | Reset value | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x18 | GPIOx_BSRR (where x = A to F) | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x1C | GPIOx_LCKR (where x = A to F) | | Res. | |
| | Reset value | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x20 | GPIOx_AFRL (where x = A to F) | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x24 | GPIOx_AFRH (where x = A to F) | | Res. | |
| | Reset value | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x28 | GPIOx_BRR (where x = A to F) | | Res. | |
| | Reset value | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

8 System configuration controller (SYSCFG)

The devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Enabling/disabling I²C Fast Mode Plus on some I/O ports
- Enabling/disabling the analog switch booster
- Configuring the IR modulation signal and its output polarity
- Remapping of some I/O ports
- Remapping the memory located at the beginning of the code area
- Flag pending interrupts from each interrupt line
- Managing robustness feature

8.1 SYSCFG registers

8.1.1 SYSCFG configuration register 1 (SYSCFG_CFGR1)

This register is used for specific configurations of memory and DMA requests remap and to control special I/O features.

Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the hardware BOOT selection. After reset these bits take the value selected by the actual boot mode configuration.

In the reset value, X is the memory mode selected by the actual boot mode configuration.

Address offset: 0x00

Reset value: 0x0000 000X

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|----------|--------------|-------------|------|----------|-------------|-------------|----------------|-------------|
| Res. | I2C3_FMP | I2C_PA10_FMP | I2C_PA9_FMP | Res. | Res. | I2C_PB9_FMP | I2C_PB8_FMP | I2C_PB7_FMP | I2C_PB6_FMP |
| | | | | | | | rw | rw | rw | | | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | BOOST_EN | Res. | Res. | Res. | PA12_RMP | PA11_RMP | Res. | MEM_MODE [1:0] | |
| | | | | | | | rw | | | | rw | rw | | rw | rw |

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

Bit 24 **I2C3_FMP**: Fast Mode Plus (FM+) enable for I2C3

This bit is set and cleared by software. It enables I²C FM+ driving capability on I/O ports configured as I2C3 through GPIOx_AFR registers.

0: Disable

1: Enable

With this bit in disable state, the I²C FM+ driving capability on I/O ports configured as I2C3 can be enabled through their corresponding I2Cx_FMP bit. When I²C FM+ is enabled, the speed control is ignored.

Note: This control bit is kept for legacy reasons. It is recommended to use the FMP bit of the I2Cx_CR1 register instead.

Bit 23 **I2C_PA10_FMP**: Fast Mode Plus (FM+) enable for PA10

This bit is set and cleared by software. It enables I²C FM+ driving capability on PA10 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I²C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I²C FM+ is enabled, the speed control is ignored.

Note: This control bit is kept for legacy reasons. It is recommended to use the FMP bit of the I2Cx_CR1 register instead.

Bit 22 **I2C_PA9_FMP**: Fast Mode Plus (FM+) enable for PA9

This bit is set and cleared by software. It enables I²C FM+ driving capability on PA9 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I²C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I²C FM+ is enabled, the speed control is ignored.

Note: This control bit is kept for legacy reasons. It is recommended to use the FMP bit of the I2Cx_CR1 register instead.

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

Bit 19 **I2C_PB9_FMP**: Fast Mode Plus (FM+) enable for PB9

This bit is set and cleared by software. It enables I²C FM+ driving capability on PB9 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I²C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I²C FM+ is enabled, the speed control is ignored.

Note: This control bit is kept for legacy reasons. It is recommended to use the FMP bit of the I2Cx_CR1 register instead.

Bit 18 **I2C_PB8_FMP**: Fast Mode Plus (FM+) enable for PB8

This bit is set and cleared by software. It enables I²C FM+ driving capability on PB8 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I²C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I²C FM+ is enabled, the speed control is ignored.

Note: This control bit is kept for legacy reasons. It is recommended to use the FMP bit of the I2Cx_CR1 register instead.

Bit 17 **I2C_PB7_FMP**: Fast Mode Plus (FM+) enable for PB7

This bit is set and cleared by software. It enables I²C FM+ driving capability on PB7 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I²C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I²C FM+ is enabled, the speed control is ignored.

Note: This control bit is kept for legacy reasons. It is recommended to use the FMP bit of the I2Cx_CR1 register instead.

Bit 16 **I2C_PB6_FMP**: Fast Mode Plus (FM+) enable for PB6

This bit is set and cleared by software. It enables I²C FM+ driving capability on PB6 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I²C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I²C FM+ is enabled, the speed control is ignored.

Note: This control bit is kept for legacy reasons. It is recommended to use the FMP bit of the I2Cx_CR1 register instead.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable

This bit selects the way of supplying I/O analog switches:

0: V_{DD}

1: Dedicated voltage booster (supplied by V_{DD})

When using the analog inputs, setting to 0 is recommended for high V_{DD}, setting to 1 for low V_{DD} (less than 2.4 V).

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

Bit 4 **PA12_RMP**: PA12 pin remapping

This bit is set and cleared by software. When set, it remaps the PA12 pin to operate as PA10 GPIO port, instead as PA12 GPIO port.

0: No remap (PA12)

1: Remap (PA10)

Bit 3 **PA11_RMP**: PA11 pin remapping

This bit is set and cleared by software. When set, it remaps the PA11 pin to operate as PA9 GPIO port, instead as PA11 GPIO port.

0: No remap (PA11)

1: Remap (PA9)

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **MEM_MODE[1:0]**: Memory mapping selection bits

These bits are set and cleared by software. They control the memory internal mapping at address 0x0000 0000. After reset these bits take on the value selected by the actual boot mode configuration. Refer to [Section 2.5: Boot configuration](#) for more details.

X0: Main flash memory mapped at 0x0000 0000

01: System flash memory mapped at 0x0000 0000

11: Embedded SRAM mapped at 0x0000 0000

8.1.2 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x18

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SPF | BKPF | Res. | Res. | BKPL | ECCL | PVDL | SPL | CCL |
| | | | | | | | rc_w1 | rc_w1 | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

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

Bit 8 SPF: SRAM1 parity error flag

This bit is set by hardware when an SRAM1 parity error is detected. It is cleared by software by writing 1.

- 0: No SRAM1 parity error detected
- 1: SRAM1 parity error detected

Bit 7 BKPF: Backup 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 6:5 Reserved, must be kept at reset value.

Bit 4 BKPL: Backup SRAM2 parity lock

This bit is set by software and cleared by a system reset. It can be used to enable and lock the SRAM2 parity error signal connection to TIM1/15/16 Break input.

- 0: SRAM2 parity error disconnected from TIM1/15/16 Break input
- 1: SRAM2 parity error connected to TIM1/15/16 Break input

Bit 3 ECCL: ECC error lock bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the flash memory ECC 2-bit error detection signal connection to TIM1/15/16 Break input.

- 0: ECC error disconnected from TIM1/15/16 Break input
- 1: ECC error connected to TIM1/15/16 Break input

Bit 2 **PVDL**: PVD lock enable bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the PVD connection to TIM1/15/16 Break input, as well as the PVDE and PLS[2:0] in the PWR_CR register.

0: PVD interrupt disconnected from TIM1/15/16 Break input. PVDE and PLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/15/16 Break input, PVDE and PLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM1 parity lock bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the SRAM1 parity error signal connection to TIM1/15/16 Break input.

0: SRAM1 parity error disconnected from TIM1/15/16 Break input

1: SRAM1 parity error connected to TIM1/15/16 Break input

Bit 0 **CCL**: Cortex®-M0+ LOCKUP bit enable bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the connection of Cortex®-M0+ LOCKUP (Hardfault) output to TIM1/15/16 Break input.

0: Cortex®-M0+ LOCKUP output disconnected from TIM1/15/16 Break input

1: Cortex®-M0+ LOCKUP output connected to TIM1/15/16 Break input

8.1.3 SYSCFG SRAM2 control and status register (SYSCFG_SCSR)

Address offset: 0x1C

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-----------|----------|
| Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SRAM2 BSY | SRAM2 ER |
| | | | | | | | | | | | | | | r | rw |

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

Bit 1 **SRAM2BSY**: SRAM2 busy by erase operation

0: No SRAM2 erase operation is ongoing

1: SRAM2 erase operation is ongoing

Bit 0 **SRAM2ER**: SRAM2 erase

Setting this bit starts a hardware SRAM2 erase operation. This bit is automatically cleared at the end of the SRAM2 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.

8.1.4 SYSCFG SRAM2 key register (SYSCFG_SKR)

Address offset: 0x20

System reset value: 0x0000 0000

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

8.1.5 SYSCFG TSC comparator register (SYSCFG_TSCCR)

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TSC_I_OCTRL | G7_IO1 | G6_IO1 | G4_IO3 | G2_IO3 | G2_IO1 |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **TSC_I_OCTRL**: I/O control in comparator mode

The I/O control in comparator mode can be overwritten by hardware.

0: I/O configured through the corresponding control register

1: I/O configured as analog when TSC AF is activated

Bit 4 **G7_IO1**: Comparator mode for group 7 on I/O 2

0: Disabled

1: Enable connection of PA9 to COMP1

Bit 3 **G6_IO1**: Comparator mode for group 6 on I/O 1

0: Disabled

1: Enable connection of PD10 to COMP2

Bit 2 **G4_IO3**: Comparator mode for group 4 on I/O 1

0: Disabled

1: Enable connection of PC6 to COMP1

Bit 1 **G2_IO3**: Comparator mode for group 2 on I/O 3

0: Disabled

1: Enable connection of PB6 to COMP2

Bit 0 **G2_IO1**: Comparator mode for group 2 on I/O 1

0: Disabled

1: Enable connection of PB4 to COMP2

8.1.6 SYSCFG interrupt line 0 status register (SYSCFG_ITLINE0)

A dedicated set of registers is implemented on the device to collect all pending interrupt sources associated with each interrupt line into a single register. This allows users to check by single read which peripheral requires service in case more than one source is associated to the interrupt line.

All bits in those registers are read only, set by hardware when there is corresponding interrupt request pending and cleared by resetting the interrupt source flags in the peripheral registers.

Address offset: 0x80

System reset value: 0x0000 0000

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

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

Bit 0 **WWDG**: Window watchdog interrupt pending flag

8.1.7 SYSCFG interrupt line 1 status register (SYSCFG_ITLINE1)

Address offset: 0x84

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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | PVM OUT4 | PVM OUT3 | PVM OUT1 | PVD OUT |
| | | | | | | | | | | | | r | r | r | r |

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

Bit 3 **PVMOUT4**: DAC supply monitoring interrupt request pending (EXTI line 21)

Bit 2 **PVMOUT3**: ADC supply monitoring interrupt request pending (EXTI line 20)

Bit 1 **PVMOUT1**: V_{DDUSB} supply monitoring interrupt request pending (EXTI line 19)

Bit 0 **PVDOUT**: PVD supply monitoring interrupt request pending (EXTI line 16).

8.1.8 SYSCFG interrupt line 2 status register (SYSCFG_ITLINE2)

Address offset: 0x88

System reset value: 0x0000 0000

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

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

Bit 1 **RTC**: RTC interrupt request pending (EXTI line 19)

Bit 0 **TAMP**: Tamper interrupt request pending (EXTI line 21)

8.1.9 SYSCFG interrupt line 3 status register (SYSCFG_ITLINE3)

Address offset: 0x8C

System reset value: 0x0000 0000

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

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

Bit 1 **FLASH_ITF**: Flash interface interrupt request pending

Bit 0 **FLASH_ECC**: Flash interface ECC interrupt request pending

8.1.10 SYSCFG interrupt line 4 status register (SYSCFG_ITLINE4)

Address offset: 0x90

System reset value: 0x0000 0000

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

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

Bit 1 **CRS**: CRS interrupt request pending

Bit 0 **RCC**: Reset and clock control interrupt request pending

8.1.11 SYSCFG interrupt line 5 status register (SYSCFG_ITLINE5)

Address offset: 0x94

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|
| Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | EXTI1 | EXTI0 |
| | | | | | | | | | | | | | | r | r |

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

Bit 1 **EXTI1**: EXTI line 1 interrupt request pending

Bit 0 **EXTI0**: EXTI line 0 interrupt request pending

8.1.12 SYSCFG interrupt line 6 status register (SYSCFG_ITLINE6)

Address offset: 0x98

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|
| Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | EXTI3 | EXTI2 |
| | | | | | | | | | | | | | | r | r |

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

Bit 1 **EXTI3**: EXTI line 3 interrupt request pending

Bit 0 **EXTI2**: EXTI line 2 interrupt request pending

8.1.13 SYSCFG interrupt line 7 status register (SYSCFG_ITLINE7)

Address offset: 0x9C

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. | EXTI15 | EXTI14 | EXTI13 | EXTI12 | EXTI11 | EXTI10 | EXTI9 | EXTI8 | EXTI7 | EXTI6 | EXTI5 | EXTI4 |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **EXTI15**: EXTI line 15 interrupt request pending

Bit 10 **EXTI14**: EXTI line 14 interrupt request pending

Bit 9 **EXTI13**: EXTI line 13 interrupt request pending

Bit 8 **EXTI12**: EXTI line 12 interrupt request pending

Bit 7 **EXTI11**: EXTI line 11 interrupt request pending

Bit 6 **EXTI10**: EXTI line 10 interrupt request pending

Bit 5 **EXTI9**: EXTI line 9 interrupt request pending

Bit 4 **EXTI8**: EXTI line 8 interrupt request pending

Bit 3 **EXTI7**: EXTI line 7 interrupt request pending

Bit 2 **EXTI6**: EXTI line 6 interrupt request pending

Bit 1 **EXTI5**: EXTI line 5 interrupt request pending

Bit 0 **EXTI4**: EXTI line 4 interrupt request pending

8.1.14 SYSCFG interrupt line 8 status register (SYSCFG_ITLINE8)

Address offset: 0xA0

System reset value: 0x0000 0000

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

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

Bit 0 **USB**: USB interrupt request pending

8.1.15 SYSCFG interrupt line 9 status register (SYSCFG_ITLINE9)

Address offset: 0xA4

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---------------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | DMA1_- CH1 |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **DMA1_CH1**: DMA1 channel 1 interrupt request pending

8.1.16 SYSCFG interrupt line 10 status register (SYSCFG_ITLINE10)

Address offset: 0xA8

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---------------|---------------|
| Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | DMA1_- CH3 | DMA1_- CH2 |
| | | | | | | | | | | | | | | r | r |

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

Bit 1 **DMA1_CH3**: DMA1 channel 3 interrupt request pending

Bit 0 **DMA1_CH2**: DMA1 channel 2 interrupt request pending

8.1.17 SYSCFG interrupt line 11 status register (SYSCFG_ITLINE11)

Address offset: 0xAC

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | Res. | DMA2_CH5 | DMA2_CH4 | DMA2_CH3 | DMA2_CH2 | DMA2_CH1 | DMA1_CH7 | DMA1_CH6 | DMA1_CH5 | DMA1_CH4 | DMAMUX |
| | | | | | | r | r | r | r | r | r | r | r | r | r |

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

Bit 9 **DMA2_CH5**: DMA2 channel 5 interrupt request pending

Bit 8 **DMA2_CH4**: DMA2 channel 4 interrupt request pending

Bit 7 **DMA2_CH3**: DMA2 channel 3 interrupt request pending

Bit 6 **DMA2_CH2**: DMA2 channel 2 interrupt request pending

Bit 5 **DMA2_CH1**: DMA2 channel 1 interrupt request pending

Bit 4 **DMA1_CH7**: DMA1 channel 7 interrupt request pending

Bit 3 **DMA1_CH6**: DMA1 channel 6 interrupt request pending

Bit 2 **DMA1_CH5**: DMA1 channel 5 interrupt request pending

Bit 1 **DMA1_CH4**: DMA1 channel 4 interrupt request pending

Bit 0 **DMAMUX**: DMAMUX interrupt request pending

8.1.18 SYSCFG interrupt line 12 status register (SYSCFG_ITLINE12)

Address offset: 0xB0

System reset value: 0x0000 0000

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

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

Bit 2 **COMP2**: Comparator 2 interrupt request pending (EXTI line 18)

Bit 1 **COMP1**: Comparator 1 interrupt request pending (EXTI line 17)

Bit 0 **ADC**: ADC interrupt request pending

8.1.19 SYSCFG interrupt line 13 status register (SYSCFG_ITLINE13)

Address offset: 0xB4

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TIM1_BRK | TIM1_UPD | TIM1_TRG | TIM1_CCU |
| | | | | | | | | | | | | r | r | r | r |

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

Bit 3 **TIM1_BRK**: Timer 1 break interrupt request pending

Bit 2 **TIM1_UPD**: Timer 1 update interrupt request pending

Bit 1 **TIM1_TRG**: Timer 1 trigger interrupt request pending

Bit 0 **TIM1_CCU**: Timer 1 commutation interrupt request pending

8.1.20 SYSCFG interrupt line 14 status register (SYSCFG_ITLINE14)

Address offset: 0xB8

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TIM1_CC4 | TIM1_CC3 | TIM1_CC2 | TIM1_CC1 |
| | | | | | | | | | | | | r | r | r | r |

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

Bit 3 **TIM1_CC4**: Timer 1 capture compare 4 interrupt request pending

Bit 2 **TIM1_CC3**: Timer 1 capture compare 3 interrupt request pending

Bit 1 **TIM1_CC2**: Timer 1 capture compare 2 interrupt request pending

Bit 0 **TIM1_CC1**: Timer 1 capture compare 1 interrupt request pending

8.1.21 SYSCFG interrupt line 15 status register (SYSCFG_ITLINE15)

Address offset: 0xBC

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TIM2 |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **TIM2**: Timer 2 interrupt request pending

8.1.22 SYSCFG interrupt line 16 status register (SYSCFG_ITLINE16)

Address offset: 0xC0

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TIM3 |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **TIM3**: Timer 3 interrupt request pending

8.1.23 SYSCFG interrupt line 17 status register (SYSCFG_ITLINE17)

Address offset: 0xC4

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|--------|------|------|
| Res. | Res. | Res. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | LPTIM1 | DAC | TIM6 |
| | | | | | | | | | | | | | r | r | r |

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

Bit 2 **LPTIM1**: Low-power timer 1 interrupt request pending (EXTI line 29)

Bit 1 **DAC**: DAC underrun interrupt request pending

Bit 0 **TIM6**: Timer 6 interrupt request pending

8.1.24 SYSCFG interrupt line 18 status register (SYSCFG_ITLINE18)

Address offset: 0xC8

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|--------|------|
| Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | LPTIM2 | TIM7 |
| | | | | | | | | | | | | | | r | r |

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

Bit 1 **LPTIM2**: Low-power timer 2 interrupt request pending (EXTI line 30)

Bit 0 **TIM7**: Timer 7 interrupt request pending

8.1.25 SYSCFG interrupt line 19 status register (SYSCFG_ITLINE19)

Address offset: 0xCC

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|--------|-------|
| Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | LPTIM3 | TIM15 |
| | | | | | | | | | | | | | | r | r |

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

Bit 1 **LPTIM3**: Low-power timer 3 interrupt request pending

Bit 0 **TIM15**: Timer 15 interrupt request pending

8.1.26 SYSCFG interrupt line 20 status register (SYSCFG_ITLINE20)

Address offset: 0xD0

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TIM16 |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **TIM16**: Timer 16 interrupt request pending

8.1.27 SYSCFG interrupt line 21 status register (SYSCFG_ITLINE21)

Address offset: 0xD4

System reset value: 0x0000 0000

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

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

Bit 1 **TSC_EOA**: TSC end of acquisition interrupt request pending

Bit 0 **TSC_MCE**: TSC max count error interrupt request pending

8.1.28 SYSCFG interrupt line 22 status register (SYSCFG_ITLINE22)

Address offset: 0xD8

System reset value: 0x0000 0000

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

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

Bit 0 **LCD**: LCD interrupt request pending

8.1.29 SYSCFG interrupt line 23 status register (SYSCFG_ITLINE23)

Address offset: 0xDC

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | I2C1 |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **I2C1**: I2C1 interrupt request pending (EXTI line 33)

8.1.30 SYSCFG interrupt line 24 status register (SYSCFG_ITLINE24)

Address offset: 0xE0

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | I2C3 | I2C4 | I2C2 |
| | | | | | | | | | | | | | r | r | r |

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

Bit 2 **I2C3**: I2C3 interrupt request pending (EXTI line 23)

Bit 1 **I2C4**: I2C4 interrupt request pending

Bit 0 **I2C2**: I2C2 interrupt request pending

8.1.31 SYSCFG interrupt line 25 status register (SYSCFG_ITLINE25)

Address offset: 0xE4

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SPI1 |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **SPI1**: SPI1 interrupt request pending

8.1.32 SYSCFG interrupt line 26 status register (SYSCFG_ITLINE26)

Address offset: 0xE8

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SPI3 | SPI2 |
| | | | | | | | | | | | | | | r | r |

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

Bit 1 **SPI3**: SPI3 interrupt request pending

Bit 0 **SPI2**: SPI2 interrupt request pending

8.1.33 SYSCFG interrupt line 27 status register (SYSCFG_ITLINE27)

Address offset: 0xEC

System reset value: 0x0000 0000

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

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

Bit 0 **USART1**: USART1 interrupt request pending, combined with EXTI line 25

8.1.34 SYSCFG interrupt line 28 status register (SYSCFG_ITLINE28)

Address offset: 0xF0

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|----------|---------|
| Res. | Res. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | LP UART2 | USART 2 |
| | | | | | | | | | | | | | | r | r |

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

Bit 1 **LPUART2**: LPUART2 interrupt request pending (EXTI line 31)

Bit 0 **USART2**: USART2 interrupt request pending (EXTI line 35)

8.1.35 SYSCFG interrupt line 29 status register (SYSCFG_ITLINE29)

Address offset: 0xF4

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------------|------------|
| Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | LP UART1 | USART 3 |
| | | | | | | | | | | | | | | r | r |

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

Bit 1 **LPUART1**: LPUART1 interrupt request pending (EXTI line 30)

Bit 0 **USART3**: USART3 interrupt request pending

8.1.36 SYSCFG interrupt line 30 status register (SYSCFG_ITLINE30)

Address offset: 0xF8

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------------|------------|
| Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | LP UART3 | USART 4 |
| | | | | | | | | | | | | | | r | r |

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

Bit 1 **LPUART3**: LPUART3 interrupt request pending (EXTI line 32)

Bit 0 **USART4**: USART4 interrupt request pending

8.1.37 SYSCFG interrupt line 31 status register (SYSCFG_ITLINE31)

Address offset: 0xFC

System reset value: 0x0000 0000

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

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

Bit 1 **AES**: AES interrupt request pending

Bit 0 **RNG**: RNG interrupt request pending

8.1.38 SYSCFG register map

The following table gives the SYSCFG register map and the reset values.

Table 41. SYSCFG register map and reset values

Table 41. SYSCFG register map and reset values (continued)

| Offset | Register | 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 |
|--------|------------------------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|---|---|---|
| 0x94 | SYSCFG_ITLINE5 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x98 | SYSCFG_ITLINE6 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x9C | SYSCFG_ITLINE7 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA0 | SYSCFG_ITLINE8 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA4 | SYSCFG_ITLINE9 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA8 | SYSCFG_ITLINE10 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xAC | SYSCFG_ITLINE11 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xB0 | SYSCFG_ITLINE12 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xB4 | SYSCFG_ITLINE13 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xB8 | SYSCFG_ITLINE14 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xBC | SYSCFG_ITLINE15 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xC0 | SYSCFG_ITLINE16 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xC4 | SYSCFG_ITLINE17 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xC8 | SYSCFG_ITLINE18 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 41. 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 |
|--------|-----------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|---|
| 0xCC | SYSCFG_ITLINE19 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD0 | SYSCFG_ITLINE20 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD4 | SYSCFG_ITLINE21 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD8 | SYSCFG_ITLINE22 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xDC | SYSCFG_ITLINE23 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xE0 | SYSCFG_ITLINE24 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xE4 | SYSCFG_ITLINE25 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xE8 | SYSCFG_ITLINE26 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xEC | SYSCFG_ITLINE27 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xF0 | SYSCFG_ITLINE28 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xF4 | SYSCFG_ITLINE29 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xF8 | SYSCFG_ITLINE30 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFC | SYSCFG_ITLINE31 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

9 Direct memory access controller (DMA)

9.1 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.

Refer to [Section 9.3](#) for information on DMA implementation.

Each channel is dedicated to managing memory access requests from one or more peripherals. DMA includes an arbiter for handling the priority between DMA requests.

9.2 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.

9.3 DMA implementation

9.3.1 DMA

The devices incorporate one or two DMA controller instances. The following implementation table shows the number of DMA channels for either instance. A dash indicates that the instance is not implemented.

Table 42. DMA implementation

| Number of channels | STM32U031xx | STM32U073xx STM32U083xx |
|--------------------|-------------|----------------------------|
| DMA1 | 7 | 7 |
| DMA2 | - | 5 |

9.3.2 DMA request mapping

The DMA controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX peripheral.

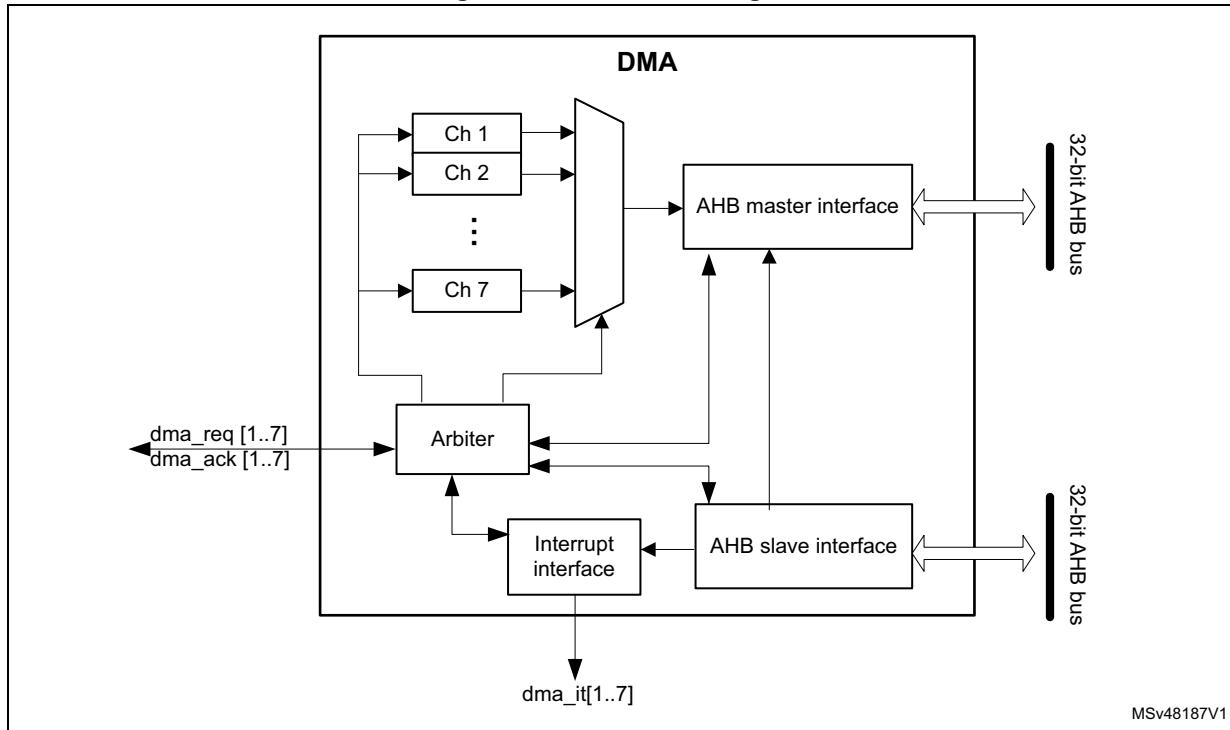
For the mapping of the different requests, refer to the DMAMUX section.

9.4 DMA functional description

9.4.1 DMA block diagram

The DMA block diagram is shown in the figure below.

Figure 22. 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.

9.4.2 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 |

9.4.3 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.

9.4.4 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.

9.4.5 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.

9.4.6 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 @0x1 / B1 @0x2 / B2 @0x3 / B3 | 1: read B0[7:0] @0x0 then write B0[7:0] @0x0 2: read B1[7:0] @0x1 then write B1[7:0] @0x1 3: read B2[7:0] @0x2 then write B2[7:0] @0x2 4: read B3[7:0] @0x3 then write B3[7:0] @0x3 | @0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3 |
| 8 | 16 | 4 | @0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3 | 1: read B0[7:0] @0x0 then write 00B0[15:0] @0x0 2: read B1[7:0] @0x1 then write 00B1[15:0] @0x2 3: read B2[7:0] @0x2 then write 00B2[15:0] @0x4 4: read B3[7:0] @0x3 then write 00B3[15:0] @0x6 | @0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3 |
| 8 | 32 | 4 | @0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3 | 1: read B0[7:0] @0x0 then write 000000B0[31:0] @0x0 2: read B1[7:0] @0x1 then write 000000B1[31:0] @0x4 3: read B2[7:0] @0x2 then write 000000B2[31:0] @0x8 4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC | @0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3 |
| 16 | 8 | 4 | @0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6 | 1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0 2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1 3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2 4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3 | @0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6 |
| 16 | 16 | 4 | @0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6 | 1: read B1B0[15:0] @0x0 then write B1B0[15:0] @0x0 2: read B3B2[15:0] @0x2 then write B3B2[15:0] @0x2 3: read B5B4[15:0] @0x4 then write B5B4[15:0] @0x4 4: read B7B6[15:0] @0x6 then write B7B6[15:0] @0x6 | @0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6 |
| 16 | 32 | 4 | @0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6 | 1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0 2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4 3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8 4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC | @0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6 |
| 32 | 8 | 4 | @0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDDBC | 1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x1 3: read BBBAB9B8[31:0] @0x8 then write B8[7:0] @0x2 4: read BFBEBDDBC[31:0] @0xC then write BC[7:0] @0x3 | @0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC |
| 32 | 16 | 4 | @0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDDBC | 1: read B3B2B1B0[31:0] @0x0 then write B1B0[15:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B5B4[15:0] @0x2 3: read BBBAB9B8[31:0] @0x8 then write B9B8[15:0] @0x4 4: read BFBEBDDBC[31:0] @0xC then write BDBC[15:0] @0x6 | @0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC |
| 32 | 32 | 4 | @0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDDBC | 1: read B3B2B1B0[31:0] @0x0 then write B3B2B1B0[31:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B7B6B5B4[31:0] @0x4 3: read BBBAB9B8[31:0] @0x8 then write BBBAB9B8[31:0] @0x8 4: read BFBEBDDBC[31:0] @0xC then write BFBEBDDBC[31:0] @0xC | @0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDDBC |

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.

9.4.7 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.

9.5 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 | - |

9.6 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).

9.6.1 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.

9.6.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

Setting the global clear bit CGIF x of the channel x in this DMA_IFCR register, causes the DMA hardware to clear the corresponding GIF x bit and any individual flag among TEIF x , HTIF x , TCIF x , in the DMA_ISR register.

Setting any individual clear bit among CTEIF x , CHTIF x , CTCIF x in this DMA_IFCR register, causes the DMA hardware to clear the corresponding individual flag and the global flag GIF x 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

9.6.3 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. | MEM2MEM | 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.

In 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.

9.6.4 DMA channel x number of data to transfer register (DMA_CNDTR x)

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. |
| | | | | | | | | | | | | | | | |
| NDT[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 |

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_CCR x 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).

9.6.5 DMA channel x peripheral address register (DMA_CPAR x)

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).

9.6.6 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).

9.6.7 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 |
|--------|---------------|------|------|------|------|--------|--------|--------|-------|--------|--------|--------|-------|--------|--------|
| 0x000 | DMA_ISR | Res. | Res. | Res. | Res. | TEIF7 | HTIF7 | TCIF7 | CGIF7 | TEIF6 | HTIF6 | TCIF6 | GIF6 | TEIF5 | HTIF5 |
| | Reset value | 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 |
| | Reset value | 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. | PL[1:0] | PSIZE[1:0] | MSIZE[1:0] | PSIZE[1:0] | MINC | PINC | CIRC | DIR | TEIE | HTIE | TClE | EN | 0 | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x00C | DMA_CNDTR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x010 | DMA_CPAR1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | | |
| 0x014 | DMA_CMAR1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 0x018 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x01C | DMA_CCR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x020 | DMA_CNDTR2 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x024 | DMA_CPAR2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x028 | DMA_CMAR2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x02C | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x030 | DMA_CCR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x034 | DMA_CNDTR3 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x038 | DMA_CPAR3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x03C | DMA_CMAR3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x040 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x044 | DMA_CCR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x048 | DMA_CNDTR4 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x04C | DMA_CPAR4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x050 | DMA_CMAR4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x054 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x058 | DMA_CCR5 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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. | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x060 | DMA_CPAR5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x064 | DMA_CMAR5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x068 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x06C | DMA_CCR6 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x070 | DMA_CNDTR6 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x074 | DMA_CPAR6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x078 | DMA_CMAR6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x07C | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x080 | DMA_CCR7 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x084 | DMA_CNDTR7 | Res. | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x088 | DMA_CPAR7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x08C | DMA_CMAR7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |

Refer to [Section 2.2](#) for the register boundary addresses.

10 DMA request multiplexer (DMAMUX)

10.1 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 controllers 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 10.3.1](#).

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 10.3.2](#).

10.2 DMAMUX main features

- Up to 12-channel programmable DMA request line multiplexer output
- 4-channel DMA request generator
- 23 trigger inputs to DMA request generator
- 23 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:
 - Up to 76 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

10.3 DMAMUX implementation

10.3.1 DMAMUX instantiation

DMAMUX instantiated with the hardware configuration parameters listed in the following table.

Table 47. DMAMUX instantiation

| Feature | DMAMUX |
|---|-------------------------------------|
| Number of DMAMUX output request channels | 12 ⁽¹⁾ /7 ⁽²⁾ |
| Number of DMAMUX request generator channels | 4 |
| Number of DMAMUX request trigger inputs | 23 |
| Number of DMAMUX synchronization inputs | 23 |
| Number of DMAMUX peripheral request inputs | Up to 76 |

1. STM32U073xx and STM32U083xx devices.
2. STM32U031xx devices.

10.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

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 | 27 | LPTIM3_IC3 | 53 | TIM2_UP |
| 2 | dmamux_req_gen1 | 28 | LPTIM3_IC4 | 54 | TIM3_CH1 |
| 3 | dmamux_req_gen2 | 29 | LPTIM3_UE | 55 | TIM3_CH2 |
| 4 | dmamux_req_gen3 | 30 | LPUART1_RX | 56 | TIM3_CH3 |
| 5 | ADC | 31 | LPUART1_TX | 57 | TIM3_CH4 |
| 6 | AES_IN | 32 | LPUART2_RX | 58 | TIM3_TRIG |
| 7 | AES_OUT | 33 | LPUART2_TX | 59 | TIM3_UP |
| 8 | DAC_Channel1 | 34 | LPUART3_RX | 60 | TIM6_UP |
| 9 | I2C1_RX | 35 | LPUART3_TX | 61 | TIM7_UP |
| 10 | I2C1_TX | 36 | SPI1_RX | 62 | TIM15_CH1 |
| 11 | I2C2_RX | 37 | SPI1_TX | 63 | TIM15_CH2 |
| 12 | I2C2_TX | 38 | SPI2_RX | 64 | TIM15_TRIG_COM |
| 13 | I2C3_RX | 39 | SPI2_TX | 65 | TIM15_UP |
| 14 | I2C3_TX | 40 | SPI3_RX | 66 | TIM16_CH1 |
| 15 | I2C4_RX | 41 | SPI3_TX | 67 | TIM16_COM |
| 16 | I2C4_TX | 42 | TIM1_CH1 | 68 | TIM16_UP |
| 17 | LPTIM1_IC1 | 43 | TIM1_CH2 | 69 | USART1_RX |
| 18 | LPTIM1_IC2 | 44 | TIM1_CH3 | 70 | USART1_TX |
| 19 | LPTIM1_IC3 | 45 | TIM1_CH4 | 71 | USART2_RX |
| 20 | LPTIM1_IC4 | 46 | TIM1_TRIG_COM | 72 | USART2_TX |
| 21 | LPTIM1_UE | 47 | TIM1_UP | 73 | USART3_RX |
| 22 | LPTIM2_IC1 | 48 | TIM2_CH1 | 74 | USART3_TX |
| 23 | LPTIM2_IC2 | 49 | TIM2_CH2 | 75 | USART4_RX |
| 24 | LPTIM2_UE | 50 | TIM2_CH3 | 76 | USART4_TX |
| 25 | LPTIM3_IC1 | 51 | TIM2_CH4 | - | - |
| 26 | LPTIM3_IC2 | 52 | TIM2_TRIG | - | - |

Table 49. DMAMUX: assignment of trigger inputs to resources

| Trigger input | Resource | Trigger input | Resource |
|---------------|------------|---------------|-------------|
| 0 | EXTI LINE0 | 12 | EXTI LINE12 |
| 1 | EXTI LINE1 | 13 | EXTI LINE13 |
| 2 | EXTI LINE2 | 14 | EXTI LINE14 |
| 3 | EXTI LINE3 | 15 | EXTI LINE15 |
| 4 | EXTI LINE4 | 16 | dmamux_evt0 |
| 5 | EXTI LINE5 | 17 | dmamux_evt1 |

Table 49. DMAMUX: assignment of trigger inputs to resources (continued)

| Trigger input | Resource | Trigger input | Resource |
|---------------|-------------|---------------|-------------|
| 6 | EXTI LINE6 | 18 | dmamux_evt2 |
| 7 | EXTI LINE7 | 19 | dmamux_evt3 |
| 8 | EXTI LINE8 | 20 | LPTIM1_OUT |
| 9 | EXTI LINE9 | 21 | LPTIM2_OUT |
| 10 | EXTI LINE10 | 22 | LPTIM3_OUT |
| 11 | EXTI LINE11 | 23 | Reserved |

Table 50. DMAMUX: assignment of synchronization inputs to resources

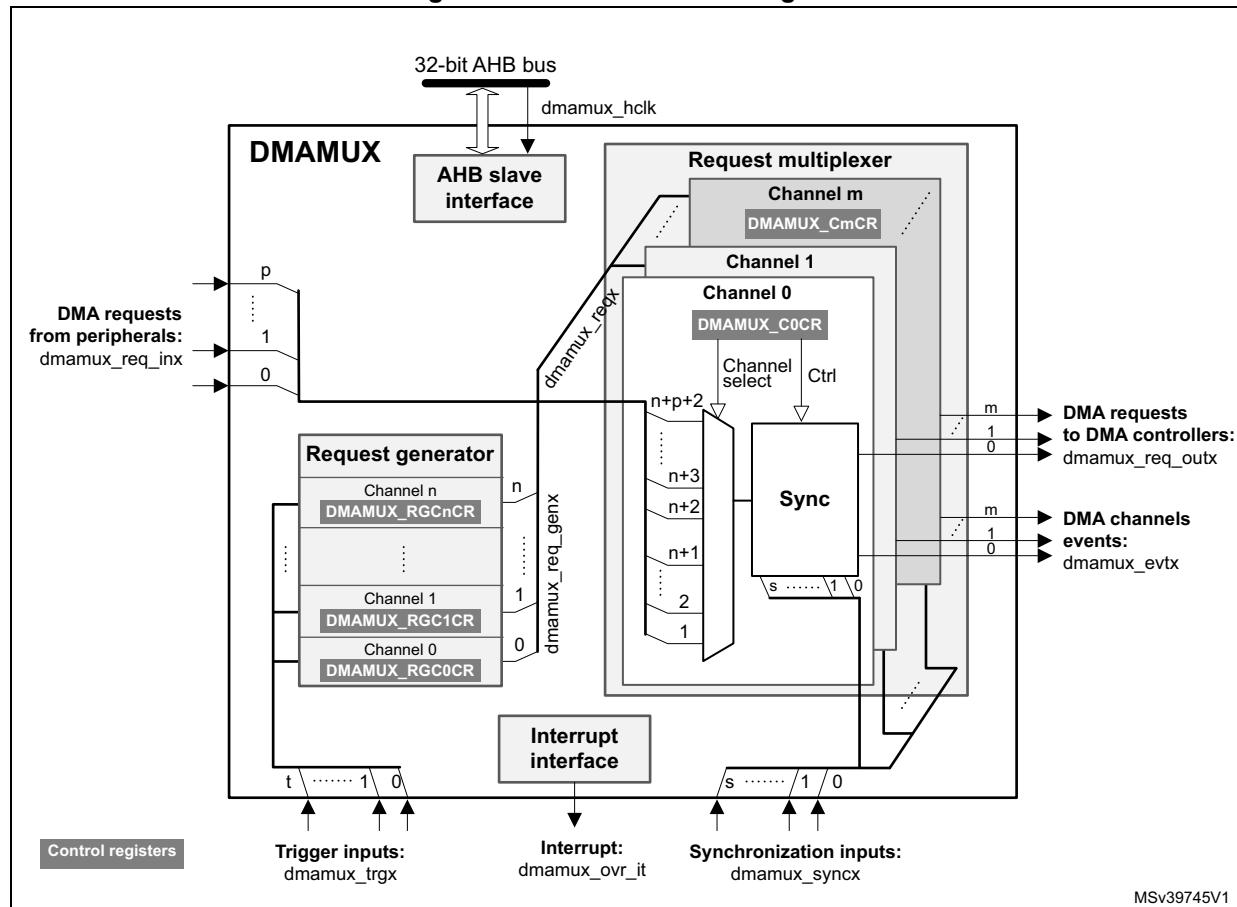
| Sync. input | Resource | Sync. input | Resource |
|-------------|-------------|-------------|-------------|
| 0 | EXTI LINE0 | 12 | EXTI LINE12 |
| 1 | EXTI LINE1 | 13 | EXTI LINE13 |
| 2 | EXTI LINE2 | 14 | EXTI LINE14 |
| 3 | EXTI LINE3 | 15 | EXTI LINE15 |
| 4 | EXTI LINE4 | 16 | dmamux_evt0 |
| 5 | EXTI LINE5 | 17 | dmamux_evt1 |
| 6 | EXTI LINE6 | 18 | dmamux_evt2 |
| 7 | EXTI LINE7 | 19 | dmamux_evt3 |
| 8 | EXTI LINE8 | 20 | LPTIM1_OUT |
| 9 | EXTI LINE9 | 21 | LPTIM2_OUT |
| 10 | EXTI LINE10 | 22 | LPTIM3_OUT |
| 11 | EXTI LINE11 | 23 | Reserved |

10.4 DMAMUX functional description

10.4.1 DMAMUX block diagram

Figure 23 shows the DMAMUX block diagram.

Figure 23. DMAMUX block diagram



MSv39745V1

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`)

10.4.2 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 controllers) |
| dmamux_evtx | DMAMUX events outputs |
| dmamux_ovr_it | DMAMUX overrun interrupts |

10.4.3 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(s).

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.

10.4.4 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.

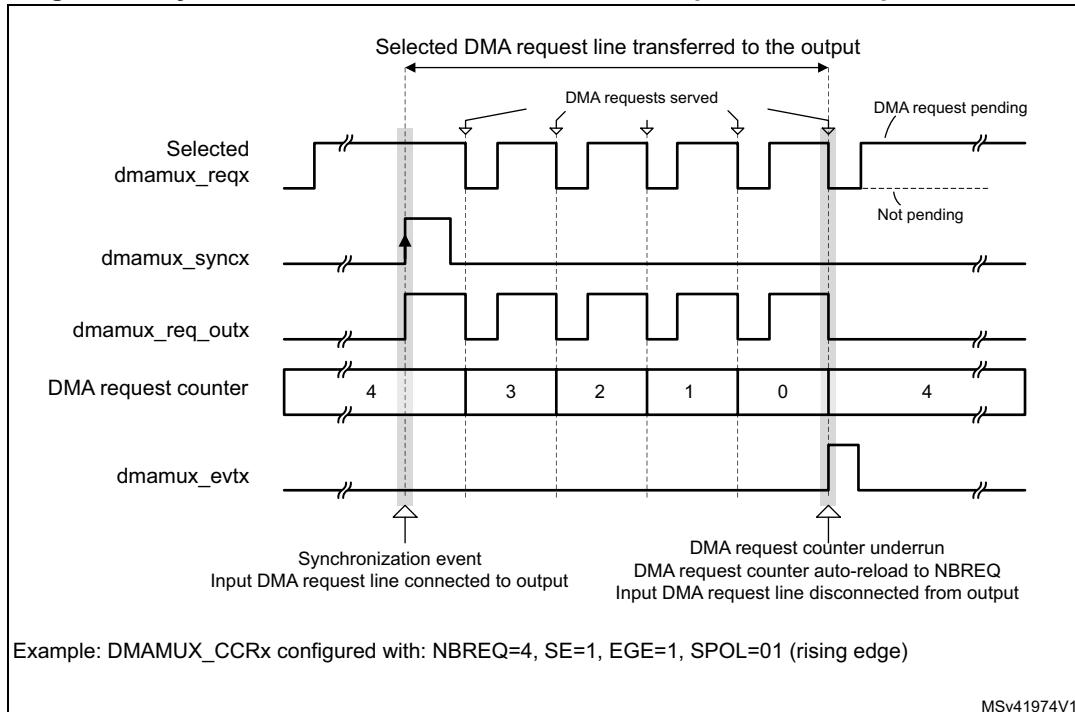
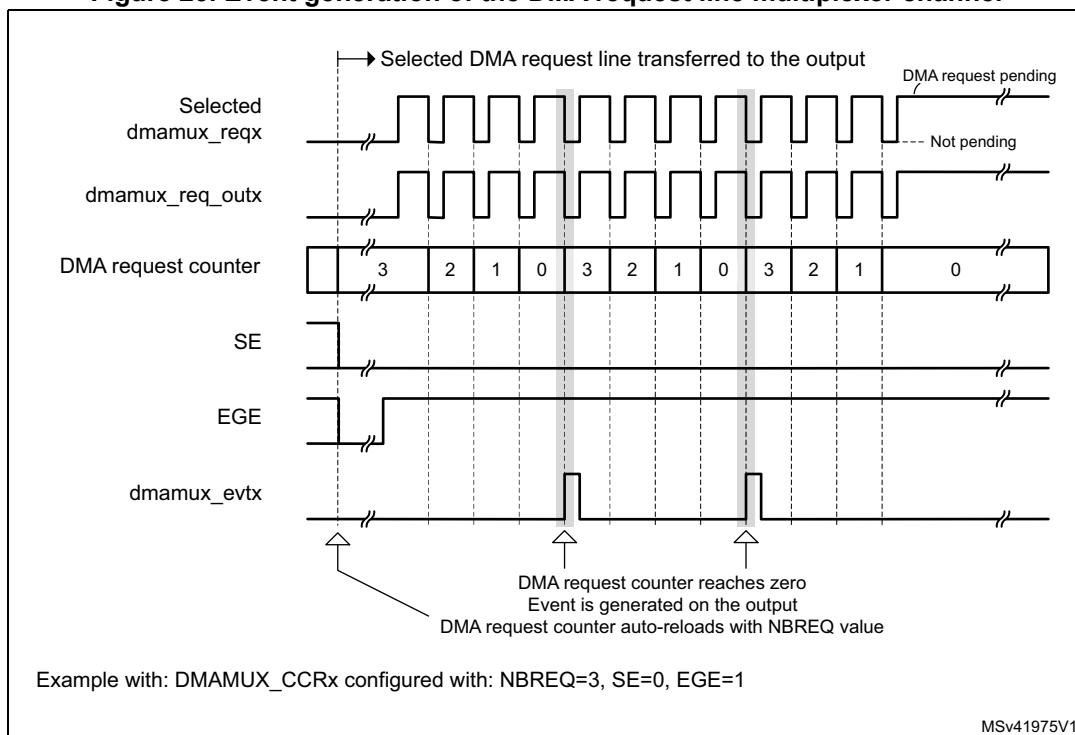
As shown in [Figure 25](#), upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

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 24. Synchronization mode of the DMAMUX request line multiplexer channel**Figure 25. 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 24](#) and [Figure 25](#).

- 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.

10.4.5 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.

10.5 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 individual 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 |

10.6 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.

10.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)

Address offset: 0x000 + 0x04 * x (x = 0 to 11)

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 | 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. | DMAREQ_ID[6:0] | | | | | | | |
| | | | | | | rw | rw | | 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

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DMAREQ_ID[6:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

10.6.2 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. | SOF11 | SOF10 | SOF9 | SOF8 | SOF7 | SOF6 | SOF5 | SOF4 | SOF3 | SOF2 | SOF1 | SOF0 |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **SOF[11:0]**: 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.

10.6.3 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. | CSOF 11 | CSOF 10 | CSOF 9 | CSOF 8 | CSOF 7 | CSOF 6 | CSOF 5 | CSOF 4 | CSOF 3 | CSOF 2 | CSOF 1 | CSOF 0 |
| | | | | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CSOF[11:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX_CSR register.

10.6.4 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. | GNBREQ[4:0] | | | | | | | |
| | | | | | | | | 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. | OIE | Res. | Res. | Res. | Res. | SIG_ID[4:0] | | | |
| | | | | | | | 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[4:0]**: Signal identification

Selects the DMA request trigger input used for the channel x of the DMA request generator

10.6.5 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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.

10.6.6 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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.

10.6.7 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

Table 53. DMAMUX register map and reset values (continued)

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

11 Nested vectored interrupt controller (NVIC)

11.1 Main features

- 32 maskable interrupt channels (not including the sixteen Cortex®-M0+ system exceptions)
- 4 programmable priority levels (2 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, which enables 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 programming manual PM0223.

11.2 SysTick calibration value register

The SysTick calibration value is set to 1000, which gives a reference time base of 1 ms with the SysTick clock set to 1 MHz.

11.3 Interrupt and exception vectors

Table 54 is the vector table. Information pertaining to a peripheral only applies to devices containing that peripheral.

Table 54. Vector table⁽¹⁾

| Position | Priority | Type of priority | Acronym | Description | Address |
|----------|----------|------------------|-------------------|---|---|
| - | - | - | - | Reserved | 0x0000_0000 |
| - | -3 | fixed | Reset | Reset | 0x0000_0004 |
| - | -2 | fixed | NMI_Handler | Non maskable interrupt. The SRAM parity error, flash ECC double err., HSE CSS and LSE CSS are linked to the NMI vector. | 0x0000_0008 |
| - | -1 | fixed | HardFault_Handler | All class of fault | 0x0000_000C |
| - | - | - | - | Reserved | 0x0000_0010 0x0000_0014 0x0000_0018 0x0000_001C 0x0000_0020 0x0000_0024 0x0000_0028 |
| - | 3 | settable | SVC_Handler | System service call via SVC instruction | 0x0000_002C |

Table 54. Vector table⁽¹⁾ (continued)

| Position | Priority | Type of priority | Acronym | Description | Address |
|-----------------|-----------------|-------------------------|--|---|----------------------------|
| - | - | - | - | Reserved | 0x0000_0030 0x0000_0034 |
| - | 5 | settable | PendSV_Handler | Pendable request for system service | 0x0000_0038 |
| - | 6 | settable | SysTick_Handler | System tick timer | 0x0000_003C |
| 0 | 7 | settable | WWDG | Window watchdog and independent watchdog interrupt | 0x0000_0040 |
| 1 | 8 | settable | PVD_PVM | PVD/PVM1/PVM2/PVM3 interrupt (combined with EXTI lines 16 & 19 & 20 & 21) | 0x0000_0044 |
| 2 | 9 | settable | RTC / TAMP | RTC and TAMP interrupts (combined EXTI lines 19 & 21) | 0x0000_0048 |
| 3 | 10 | settable | FLASH | Flash global interrupt | 0x0000_004C |
| 4 | 11 | settable | RCC / CRS | RCC and CRS global interrupt | 0x0000_0050 |
| 5 | 12 | settable | EXTI0_1 | EXTI lines 0 & 1 interrupt | 0x0000_0054 |
| 6 | 13 | settable | EXTI2_3 | EXTI lines 2 & 3 interrupt | 0x0000_0058 |
| 7 | 14 | settable | EXTI4_15 | EXTI lines 4 to 15 interrupt | 0x0000_005C |
| 8 | 15 | settable | USB | USB global interrupt (combined with EXTI line 33) | 0x0000_0060 |
| 9 | 16 | settable | DMA1_Channel1 | DMA1 channel 1 interrupt | 0x0000_0064 |
| 10 | 17 | settable | DMA1_Channel2_3 | DMA1 channel 2 & 3 interrupts | 0x0000_0068 |
| 11 | 18 | settable | DMA1_Channel4_5_6_7 / DMAMUX / DMA2_Channel1_2_3_4_5 | DMA1 channel 4, 5, 6, 7, DMAMUX, DMA2 channel 1, 2, 3, 4, 5 interrupts | 0x0000_006C |
| 12 | 19 | settable | ADC_COMP | ADC and COMP interrupts (ADC combined with EXTI lines 17 & 18) | 0x0000_0070 |
| 13 | 20 | settable | TIM1_BRK_UP_TRG_COM | TIM1 break, update, trigger and commutation interrupts | 0x0000_0074 |
| 14 | 21 | settable | TIM1_CC | TIM1 Capture Compare interrupt | 0x0000_0078 |
| 15 | 22 | settable | TIM2 | TIM2 global interrupt | 0x0000_007C |
| 16 | 23 | settable | TIM3 | TIM3 global interrupt | 0x0000_0080 |
| 17 | 24 | settable | TIM6_DAC / LPTIM1 | TIM6, LPTIM1 and DAC global interrupt (combined with EXTI line 29) | 0x0000_0084 |
| 18 | 25 | settable | TIM7 / LPTIM2 | TIM7 and LPTIM2 global interrupt (combined with EXTI line 30) | 0x0000_0088 |
| 19 | 26 | settable | TIM15 / LPTIM3 | TIM15 and LPTIM3 global interrupt (combined with EXTI line 29) | 0x0000_008C |
| 20 | 27 | settable | TIM16 | TIM16 global interrupt | 0x0000_0090 |

Table 54. Vector table⁽¹⁾ (continued)

| Position | Priority | Type of priority | Acronym | Description | Address |
|-----------------|-----------------|-------------------------|--------------------|--|----------------|
| 21 | 28 | settable | TSC | TSC global interrupt | 0x0000_0094 |
| 22 | 29 | settable | LCD | LCD global interrupt (combined with EXTI line 32) | 0x0000_0098 |
| 23 | 30 | settable | I2C1 | I2C1 global interrupt (combined with EXTI line 23) | 0x0000_009C |
| 24 | 31 | settable | I2C2 / I2C3 / I2C4 | I2C2/3/4 global interrupt | 0x0000_00A0 |
| 25 | 32 | settable | SPI1 | SPI1 global interrupt | 0x0000_00A4 |
| 26 | 33 | settable | SPI2 / SPI3 | SPI2/3 global interrupt | 0x0000_00A8 |
| 27 | 34 | settable | USART1 | USART1 global interrupt (combined with EXTI line 25) | 0x0000_00AC |
| 28 | 35 | settable | USART2 / LPUART2 | USART2 and LPUART2 global interrupt (combined with EXTI lines 26 & 35) | 0x0000_00B0 |
| 29 | 36 | settable | USART3 / LPUART1 | USART3 and LPUART1 global interrupt (combined with EXTI lines 24 & 28) | 0x0000_00B4 |
| 30 | 37 | settable | USART4 / LPUART3 | USART4 and LPUART3 global interrupt (combined with EXTI lines 20 & 34) | 0x0000_00B8 |
| 31 | 38 | settable | AES / RNG | AES and RNG global interrupts | 0x0000_00BC |

1. The grayed cells correspond to the Cortex®-M0+ system exceptions.

12 Extended interrupt and event controller (EXTI)

The Extended interrupt and event controller (EXTI) manages the CPU and system wake-up through configurable and direct event inputs (lines). It provides wake-up requests to the power control, and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wake-up requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can also be used in Run modes.

The EXTI also includes the EXTI I/O port mux.

12.1 EXTI main features

The EXTI main features are the following:

- System wake-up upon event on any input
- Wake-up flag and CPU interrupt generation for events not having a wake-up flag in their source peripheral
- Configurable events (from I/Os, peripherals not having an associated interrupt pending status bit, or peripherals generating a pulse)
 - Selectable active trigger edge
 - Independent rising and falling edge interrupt pending status bits
 - Individual interrupt and event generation mask, used for conditioning the CPU wake-up, interrupt and event generation
 - SW trigger possibility
- Direct events (from peripherals having an associated flag and interrupt pending status bit)
 - Fixed rising edge active trigger
 - No interrupt pending status bit in the EXTI
 - Individual interrupt and event generation mask for conditioning the CPU wake-up and event generation
 - No SW trigger possibility
- I/O port selector

12.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block, and EXTI mux as shown in [Figure 26](#).

The register block contains all the EXTI registers.

The event input trigger block provides an event input edge trigger logic.

The masking block provides the event input distribution to the different wake-up, interrupt and event outputs, and the masking of these.

The EXTI mux provides the I/O port selection on to the EXTI event signal.

Figure 26. EXTI block diagram

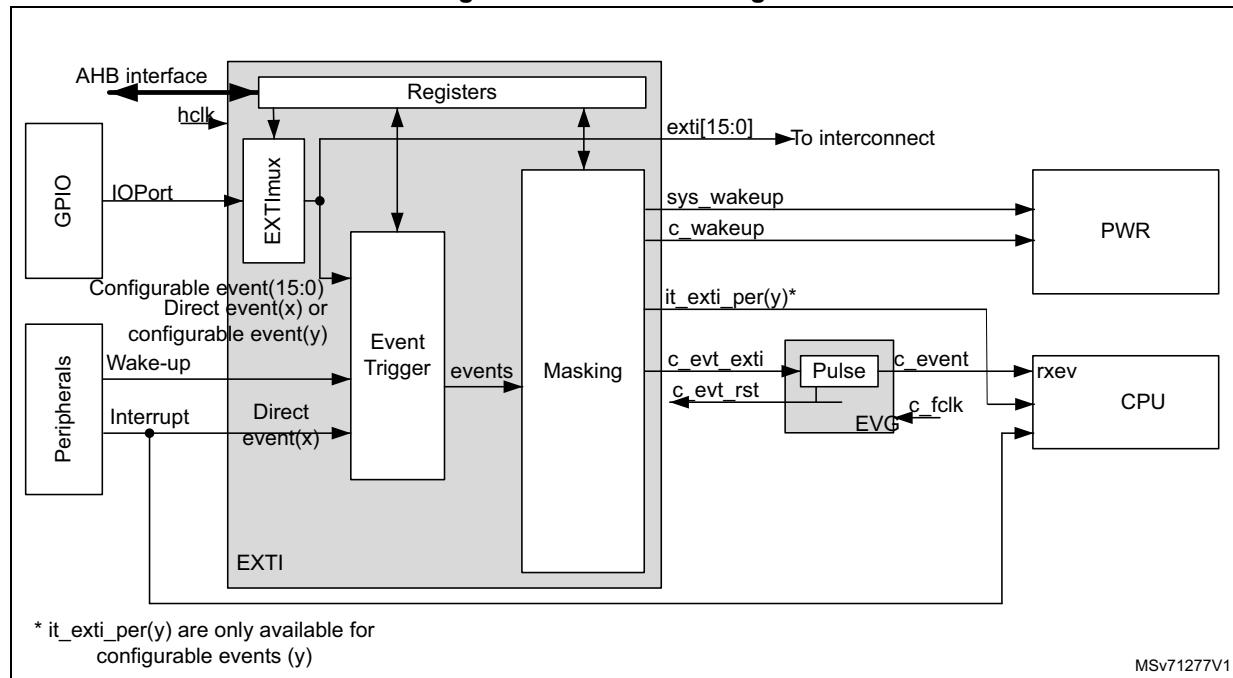


Table 55. EXTI signal overview

| Signal name | I/O | Description |
|-----------------------|-----|--|
| AHB interface | I/O | EXTI register bus interface. When one event is configured to allow security, the AHB interface support secure accesses |
| hclk | I | AHB bus clock and EXTI system clock |
| Configurable event(y) | I | Asynchronous wake-up events from peripherals that do not have an associated interrupt and flag in the peripheral |
| Direct event(x) | I | Synchronous and asynchronous wake-up events from peripherals having an associated interrupt and flag in the peripheral |
| IOPort(n) | I | GPIO ports[15:0] |
| exti[15:0] | O | EXTI output port to trigger other IPs |
| it_exti_per (y) | O | Interrupts to the CPU associated with configurable event (y) |
| c_evt_exti | O | High-level sensitive event output for CPU synchronous to hclk |
| c_evt_rst | I | Asynchronous reset input to clear c_evt_exti |
| sys_wakeup | O | Asynchronous system wake-up request to PWR for ck_sys and hclk |
| c_wakeup | O | Wake-up request to PWR for CPU, synchronous to hclk |

Table 56. EVG pin overview

| Pin name | I/O | Description |
|----------|-----|---|
| c_fclk | I | CPU free-running clock |
| c_evt_in | I | High-level sensitive event input from EXTI, asynchronous to CPU clock |

Table 56. EVG pin overview (continued)

| Pin name | I/O | Description |
|-----------|-----|--|
| c_event | O | Event pulse, synchronous to CPU clock |
| c_evt_RST | O | Event reset signal, synchronous to CPU clock |

12.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate wake-up or interrupt events when the system is in Stop mode are connected to the EXTI.

- Peripheral wake-up signals that generate a pulse or that do not have an interrupt status bits in the peripheral, are connect to an EXTI configurable line. For these events the EXTI provides a status pending bit which requires to be cleared. It is the EXTI interrupt associated with the status bit that interrupts the CPU.
- Peripheral interrupt and wake-up signals that have a status bit in the peripheral which requires to be cleared in the peripheral, are connected to an EXTI direct line. There is no status pending bit within the EXTI. The interrupt or wake-up is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.
- All GPIO ports input to the EXTI multiplexer, allowing to select a port to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC(a) of the CPU.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wake-up signals are connected to the PWR block, and are used to wake up the system and CPU sub-system bus clocks.

12.3 EXTI functional description

Depending on the EXTI line type and wake-up target(s), different logic implementations are used. The applicable features and control or status registers are:

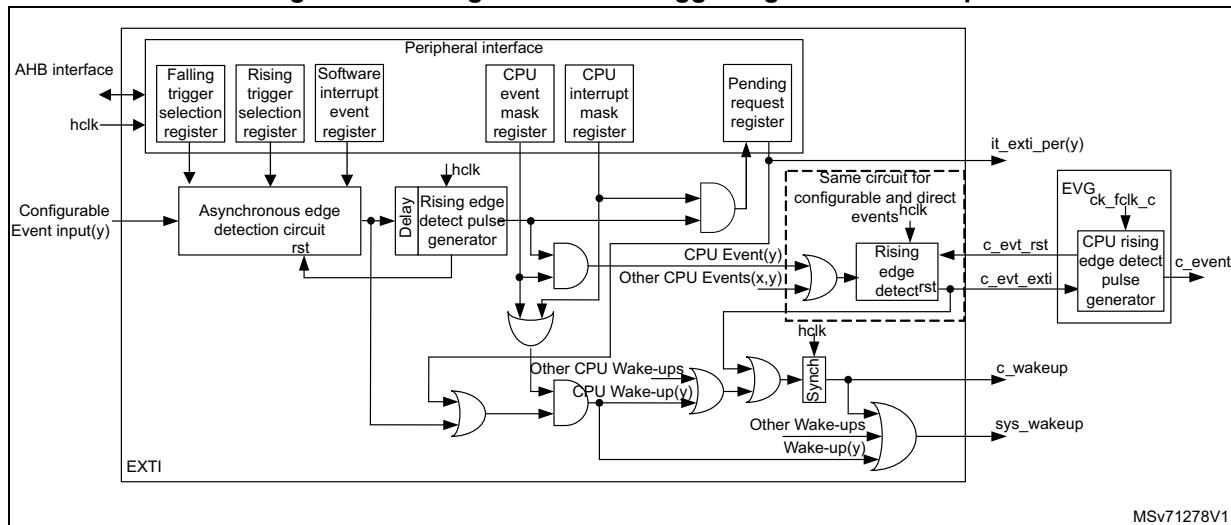
- rising and falling edge event enable through
 - EXTI rising trigger selection register (EXTI_RTSR1)*
 - EXTI falling trigger selection register 1 (EXTI_FTSR1)*
- software trigger through *EXTI software interrupt event register 1 (EXTI_SWIER1)*
- pending interrupt flagging through
 - EXTI rising edge pending register 1 (EXTI_RPR1)*
 - EXTI falling edge pending register 1 (EXTI_FPR1)*
 - EXTI external interrupt selection register x (EXTI_EXTICRx)*
- CPU wake-up and interrupt enable through
 - EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)*
 - EXTI CPU wake-up with interrupt mask register (EXTI_IMR2)*
- CPU wake-up and event enable through
 - EXTI CPU wake-up with event mask register (EXTI_EMR1)*
 - EXTI CPU wake-up with event mask register (EXTI_EMR2)*

Table 57. EXTI event input configurations and register control

| Event input type | Logic implementation | EXTI_RTSR1 | EXTI_FTCSR1 | EXTI_SWIER1 | EXTI_RPR1 | EXTI_IMR1 | EXTI_EMRx |
|------------------|--|------------|-------------|-------------|-----------|-----------|-----------|
| Configurable | Configurable event input wake-up logic | x | x | x | x | x | x |
| Direct | Direct event input wake-up logic | - | - | - | - | x | x |

12.3.1 EXTI configurable event input wake-up

Figure 27 is a detailed representation of the logic associated with configurable event inputs which wake up the CPU sub-system bus clocks and generated an EXTI pending flag and interrupt to the CPU and or a CPU wake-up event.

Figure 27. Configurable event trigger logic CPU wake-up

The software interrupt event register allows triggering 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.

The CPU has its dedicated interrupt mask register and a dedicated event mask registers. The enabled event allows generating an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The event pending registers (EXTI_RPR1 and EXTI_FPR1) is not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers, shared by the CPU. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts need to be acknowledged by software in the EXTI_RPR1 and/or EXTI_FPR1 registers.

When a CPU interrupt or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees the wake-up of the EXTI hclk clock before the asynchronous edge detection circuit is reset.

Note: A detected configurable event interrupt pending request can be cleared by the CPU. The system cannot enter low-power modes as long as an interrupt pending request is active.

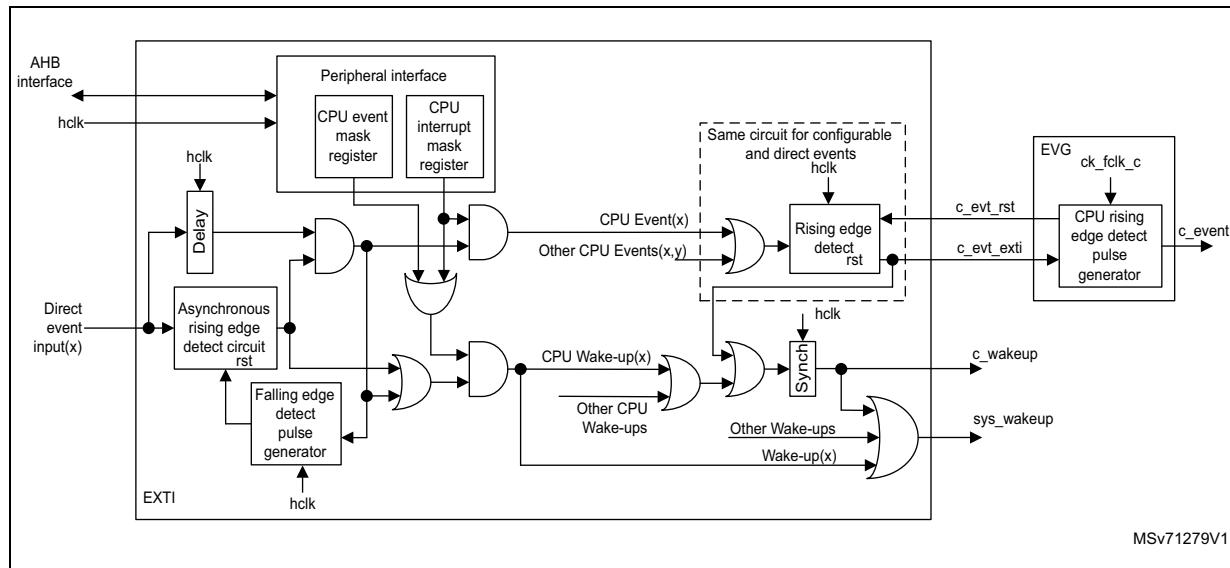
12.3.2 EXTI direct event input wake-up

Figure 28 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 wake-up event. The peripheral synchronous interrupt, associated with the direct wake-up event wakes up the CPU.

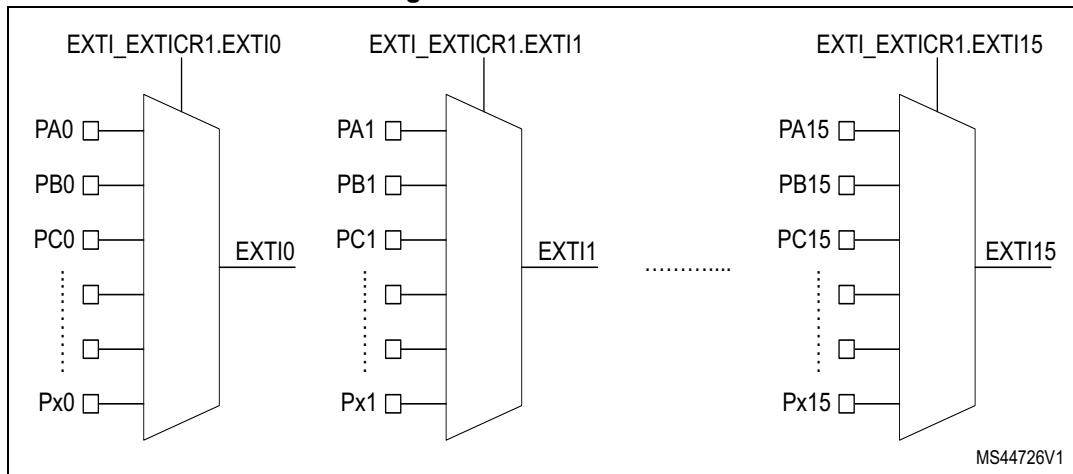
The EXTI direct event is able to generate a CPU event. This CPU event wakes up the CPU. The CPU event may occur before the interrupt flag of the associated peripheral is set.

Figure 28. Direct event trigger logic CPU wake-up



12.3.3 EXTI mux

The EXTI mux allows selecting GPIOs as interrupts and wake-up. The GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI mux output is controlled through the [EXTI external interrupt selection register x \(EXTI_EXTICR_x\)](#) register.

Figure 29. EXTI GPIO mux

The EXTI lines (event inputs) are connected as shown in the following table.

Table 58. EXTI line connections⁽¹⁾

| EXTI line | Line source | Line type |
|-----------|-------------------------------|--------------|
| 0-15 | GPIO | Configurable |
| 16 | PVD output | Configurable |
| 17 | COMP1 output | Configurable |
| 18 | COMP2 output | Configurable |
| 19 | V _{DDUSB} monitoring | Configurable |
| 20 | ADC supply monitoring | Configurable |
| 21 | DAC supply monitoring | Configurable |
| 22 | LCD wake-up | Direct |
| 23 | I ₂ C3 wake-up | Direct |
| 24 | LPTIM1 wake-up | Direct |
| 25 | LPTIM2 wake-up | Direct |
| 26 | LPTIM3 wake-up | Direct |
| 27 | LSE_CSS | Direct |
| 28 | RTC | Direct |
| 29 | TAMP | Direct |
| 30 | LPUART1 wake-up | Direct |
| 31 | LPUART2 wake-up | Direct |
| 32 | LPUART3 wake-up | Direct |
| 33 | I ₂ C1 wake-up | Direct |

Table 58. EXTI line connections⁽¹⁾ (continued)

| EXTI line | Line source | Line type |
|-----------|----------------|-----------|
| 34 | USART1 wake-up | Direct |
| 35 | USART2 wake-up | Direct |
| 36 | USB | Direct |
| 37 | WWDG | Direct |

1. EXTI lines 18, 19, 22, 26, 32 and 36 are reserved on STM32U031xx devices.

12.4 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wake-up 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 wake-up is conditioned by the CPU interrupt mask and CPU event mask.

Table 59. Masking functionality

| CPU interrupt enable IMn of EXTI_IMR | CPU event enable EMn of EXTI_EMR | Configurable event inputs RPIFn of EXTI_RPR, FPIFn of EXTI_FPR | exti(n) interrupt ⁽¹⁾ | CPU event | CPU wake-up |
|--------------------------------------|----------------------------------|--|----------------------------------|-----------|--------------------|
| 0 | 0 | No | Masked | Masked | Masked |
| | 1 | No | Masked | Yes | Yes |
| 1 | 0 | Status latched | Yes | Masked | Yes ⁽²⁾ |
| | 1 | Status latched | Yes | Yes | Yes |

1. The single exti(n) interrupt goes to the CPU. If no interrupt is required for CPU, the exti(n) interrupt must be masked in the CPU NVIC.

2. Only if CPU interrupt is enabled in EXTI_IMR.IMn.

For configurable event inputs, upon an edge on the event input, an event request is generated if that edge (rising or/and falling) is enabled. When the associated CPU interrupt is unmasked, the corresponding RPIFn and/or FPIFn bit is/are set in the EXTI_RPR or/and EXTI_FPR register, waking up the CPU subsystem and activating CPU interrupt signal. The RPIFn and/or FPIFn pending bit is cleared by writing 1 to it, which clears the CPU interrupt request.

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 interrupt is unmasked, the corresponding CPU subsystem is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU event must be unmasked to generate an event. Upon an enabled edge occurring on an event input, a CPU event pulse is generated. There is no event pending bit.

For the configurable event inputs, the software can generate an event request by setting the corresponding bit of the software interrupt/event register EXTI_SWIER1, which has the

effect of a rising edge on the event input. The pending rising edge event flag is set in the EXTI_RPR1 register, irrespective of the EXTI_RTSR1 register setting.

12.5 EXTI registers

The EXTI register map is divided in the following sections:

Table 60. EXTI register map sections

| Address | Description |
|---------------|---|
| 0x000 - 0x01C | General configurable event [31:0] configuration |
| 0x060 - 0x06C | EXTI I/O port multiplexer |
| 0x080 - 0x0BC | CPU input event configuration |

All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

12.5.1 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. | RT21 | RT20 | RT19 | RT18 | RT17 | RT16 |
| | | | | | | | | | | rw | rw | 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 |

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **RTx**: Rising trigger event configuration bit of configurable line x (x = 21 to 0)

Each bit enables/disables the rising edge trigger for the event and interrupt on the corresponding line.

0: Disable

1: Enable

Bits 18 and 19 are available only on STM32U0x3xx devices. They are reserved on STM32U031xx devices.

12.5.2 EXTI falling trigger selection register 1 (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. | FT21 | FT20 | FT19 | FT18 | FT17 | FT16 |
| | | | | | | | | | | rw | rw | 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 |

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **FT_x**: Falling trigger event configuration bit of configurable line x (x = 21 to 0)

Each bit enables/disables the falling edge trigger for the event and interrupt on the corresponding line.

0: Disable

1: Enable

Bits 18 and 19 are available only on STM32U0x3xx devices. They are reserved on STM32U031xx devices.

12.5.3 EXTI software interrupt event register 1 (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. | SWI21 | SWI20 | SWI19 | SWI18 | SWI17 | SWI16 |
| | | | | | | | | | | rw | rw | 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 31:22 Reserved, must be kept at reset value.

Bits 21:0 **SWIx**: Software rising edge event trigger on line x (x = 21 to 0)

Setting of any bit by software triggers a rising edge event on the corresponding line x, resulting in an interrupt, independently of EXTI_RTSR1 and EXTI_FTSR1 settings. The bits are automatically cleared by HW. Reading of any bit always returns 0.

0: No effect

1: Rising edge event generated on the corresponding line, followed by an interrupt
Bits 18 and 19 are available only on STM32U0x3xx devices. They are reserved on STM32U031xx devices.

12.5.4 EXTI rising edge pending register 1 (EXTI_RPR1)

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. | RPIF21 | RPIF20 | RPIF19 | RPIF18 | RPIF17 | RPIF16 |
| | | | | | | | | | | 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 |
| RPIF15 | RPIF14 | RPIF13 | RPIF12 | RPIF11 | RPIF10 | RPIF9 | RPIF8 | RPIF7 | RPIF6 | RPIF5 | RPIF4 | RPIF3 | RPIF2 | RPIF1 | RPIF0 |
| 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 31:22 Reserved, must be kept at reset value.

Bits 21:0 **RPIFx**: Rising edge event pending for configurable line x (x = 21 to 0)

Each bit is set upon a rising edge event generated by hardware or by software (through the EXTI_SWIER1 register) on the corresponding line. Each bit is cleared by writing 1 into it.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

Bits 18 and 19 are available only on STM32U0x3xx devices. They are reserved on STM32U031xx devices.

12.5.5 EXTI falling edge pending register 1 (EXTI_FPR1)

Address offset: 0x010

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. | FPIF21 | FPIF20 | FPIF19 | FPIF18 | FPIF17 | FPIF16 |
| | | | | | | | | | | 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 |
| FPIF15 | FPIF14 | FPIF13 | FPIF12 | FPIF11 | FPIF10 | FPIF9 | FPIF8 | FPIF7 | FPIF6 | FPIF5 | FPIF4 | FPIF3 | FPIF2 | FPIF1 | FPIF0 |
| 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 31:22 Reserved, must be kept at reset value.

Bits 21:0 **FPIFx**: Falling edge event pending for configurable line x (x = 21 to 0)

Each bit is set upon a falling edge event generated by hardware or by software (through the EXTI_SWIER1 register) on the corresponding line. Each bit is cleared by writing 1 into it.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

Bits 18 and 19 are available only on STM32U0x3xx devices. They are reserved on STM32U031xx devices.

12.5.6 EXTI external interrupt selection register x (EXTI_EXTICRx)

Address offset: 0x060 + 0x4 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

EXTIm fields contain only the number of bits in line with the nb_iport configuration.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------------------------|----|----|----|----|----|----|----|----------------------------|----|----|----|----|----|----|----|
| EXTI{4 * (x - 1) + 3}[7:0] | | | | | | | | EXTI{4 * (x - 1) + 2}[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EXTI{4 * (x - 1) + 1}[7:0] | | | | | | | | EXTI{4 * (x - 1)}[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:24 **EXTI{4 * (x - 1) + 3}[7:0]: EXTI{4 * (x - 1) + 3} GPIO port selection**

These bits are written by software to select the source input for EXTI{4 * (x - 1) + 3} external interrupt.

0x00: PA[4 * (x - 1) + 3] pin

0x01: PB[4 * (x - 1) + 3] pin

0x02: PC[4 * (x - 1) + 3] pin

0x03: PD[4 * (x - 1) + 3] pin

0x04: reserved, must not be used

0x05: PF[4 * (x - 1) + 3] pin

Others: reserved, must not be used

Bits 23:16 **EXTI{4 * (x - 1) + 2}[7:0]: EXTI{4 * (x - 1) + 2} GPIO port selection**

These bits are written by software to select the source input for EXTI{4 * (x - 1) + 2} external interrupt.

0x00: PA[4 * (x - 1) + 2] pin

0x01: PB[4 * (x - 1) + 2] pin

0x02: PC[4 * (x - 1) + 2] pin

0x03: PD[4 * (x - 1) + 2] pin

0x04: reserved, must not be used

0x05: PF[4 * (x - 1) + 2] pin

Others: reserved, must not be used

Bits 15:8 **EXTI{4 * (x - 1) + i}[7:0]: EXTI{4 * (x - 1) + 1} GPIO port selection**

These bits are written by software to select the source input for EXTI{4 * (x - 1) + 1} external interrupt.

0x00: PA[4 * (x - 1) + 1] pin

0x01: PB[4 * (x - 1) + 1] pin

0x02: PC[4 * (x - 1) + 1] pin

0x03: PD[4 * (x - 1) + 1] pin

0x04: reserved, must not be used

0x05: PF[4 * (x - 1) + 1] pin

Others: reserved, must not be used

Bits 7:0 **EXTI{4 * (x - 1)}[7:0]**: EXTI{4 * (x - 1)} GPIO port selection

These bits are written by software to select the source input for EXTI{4 * (x - 1)} external interrupt.

- 0x00: PA[4 * (x - 1)] pin
- 0x01: PB[4 * (x - 1)] pin
- 0x02: PC[4 * (x - 1)] pin
- 0x03: PD[4 * (x - 1)] pin
- 0x04: reserved, must not be used
- 0x05: PF[4 * (x - 1)] pin
- Others: reserved, must not be used

12.5.7 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0xFFFF8 0000

Contains register bits for configurable events and direct events.

The reset value is set such as to, by default, enable interrupt from direct lines, and disable interrupt from configurable lines.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| IM31 | IM30 | IM29 | IM28 | IM27 | IM26 | IM25 | IM24 | IM23 | IM22 | IM21 | IM20 | IM19 | IM18 | IM17 | IM16 |
| 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 |

Bits 31:0 **IMx**: CPU wake-up with interrupt mask on line x (x = 31 to 0)

Setting/clearing each bit unmasks/masks the CPU wake-up with interrupt, by an event on the corresponding line.

0: wake-up with interrupt masked

1: wake-up with interrupt unmasked

Bits 18, 19, 22 and 26 are available only on STM32U0x3xx devices, they are reserved on STM32U031xx devices.

12.5.8 EXTI CPU wake-up 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 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| EM31 | EM30 | EM29 | EM28 | EM27 | EM26 | EM25 | EM24 | EM23 | EM22 | EM21 | EM20 | EM19 | EM18 | EM17 | EM16 |
| 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 |

Bits 31:0 **EMx**: CPU wake-up with event generation mask on line x (x = 31 to 0)

Setting/clearing each bit unmasks/masks the CPU wake-up with event generation on the corresponding line.

0: wake-up with event generation masked

1: wake-up with event generation unmasked

Bits 18, 19, 22 and 26 are available only on STM32U0x3xx devices, they are reserved on STM32U031xx devices.

12.5.9 EXTI CPU wake-up with interrupt mask register (EXTI_IMR2)

Address offset: 0x090

Reset value: 0xFFFF FFFF

Contains register bits for configurable events and direct events.

The reset value is set such as to, by default, enable interrupt from direct lines, and disable interrupt from configurable lines.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | IM37 | IM36 | IM35 | IM34 | IM33 | IM32 |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **IMx**: CPU wake-up with interrupt mask on line x (x = 37 to 32)

Setting/clearing this bit unmasks/masks the CPU wake-up with interrupt, by an event on the corresponding line.

0: wake-up with interrupt request from Line x is masked

1: wake-up with interrupt request from Line x is unmasked

Bit IM36 is available only on STM32U0x3xx devices, it is reserved on STM32U031xx devices.

12.5.10 EXTI CPU wake-up with event mask register (EXTI_EMR2)

Address offset: 0x094

Reset value: 0x0000 0000

Contains register bits for configurable events and direct events.

The reset value is set such as to, by default, enable interrupt from direct lines, and disable interrupt from configurable lines.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | EM37 | EM36 | EM35 | EM34 | EM33 | EM32 |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **EMx**: CPU wake-up with event generation mask on line x, (x = 37 to 32)

Setting/clearing each bit unmasks/masks the CPU wake-up with event generation on the corresponding line.

0: wake-up with event generation masked

1: wake-up with event generation unmasked

Bit IM36 is available only on STM32U0x3xx devices, it is reserved on STM32U031xx devices.

12.5.11 EXTI register map

The following table gives the EXTI register map and the reset values.

Table 61. EXTI controller register map and reset values

Table 61. EXTI controller register map and reset values (continued)

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------|------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| 0x084 | EXTI_EMR1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | |
| 0x088-0x08C | Reserved | Res. | | | | | | | | | | | | | | | | | | | |
| 0x090 | EXTI_IMR2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x094 | EXTI_EMR2 | Res. | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

13 Cyclic redundancy check calculation unit (CRC)

13.1 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.

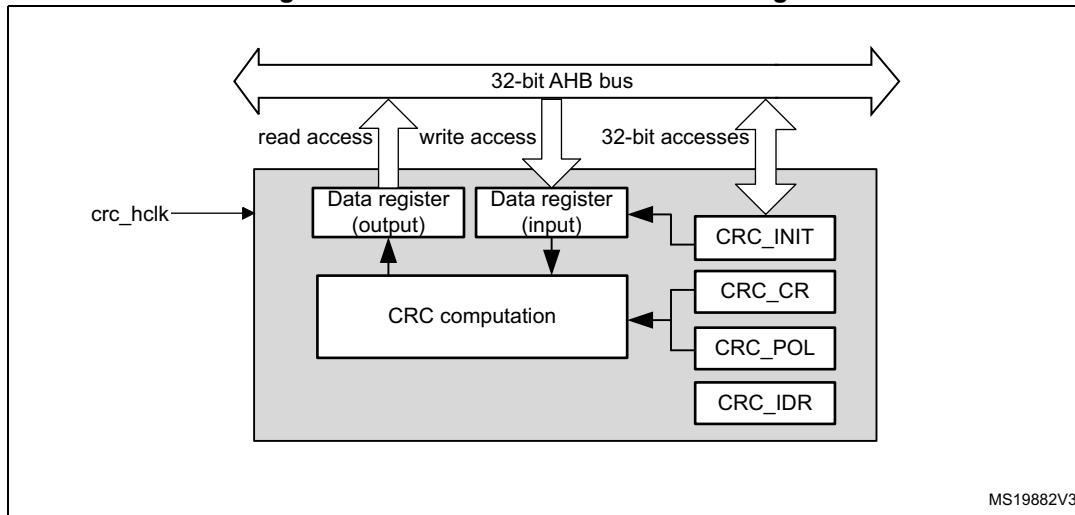
13.2 CRC main features

- 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 options 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

13.3 CRC functional description

13.3.1 CRC block diagram

Figure 30. CRC calculation unit block diagram



13.3.2 CRC internal signals

Table 62. CRC internal input/output signals

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

13.3.3 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. When the RTYPE_IN bit is set in CRC_CR, non-word accesses to CRC_DR register are ignored. 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. If RTYPE_IN bit is set in this register, the input reversing operation can be performed with a byte or half-word granularity, but not with single bit granularity.

For example, with RTYPE_IN cleared, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte (REV_IN[1:0] = 01)
- 0xD458B23C with bit-reversal done by half-word (REV_IN[1:0] = 10)
- 0xB23CD458 with bit-reversal done on the full word (REV_IN[1:0] = 11)

With RTYPE_IN set, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x3C4D1A2B with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x4D3C2B1A with byte-reversal done by word (REV_IN[1:0] = 10)

The output data can also be reversed by setting the REV_OUT[1:0] bits in the CRC_CR register. If RTYPE_OUT is set in this register, the output reversing operation is performed with a byte or half-word granularity, but not with single bit granularity.

With RTYPE_OUT cleared, the operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

With RTYPE_OUT set, 0x11223344 output data are converted as:

- 0x33441122 with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x44332211 with byte-reversal done by word (REV_IN[1:0] = 10)

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 $X + X^2 + \dots + X^n$, while the type of an odd polynomial is $1 + X + X^2 + \dots + X^n$.*

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.

13.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes when RTYPE bit is cleared in CRC_CR. When RTYPE is set, only word accesses are allowed. For the other registers only 32-bit accesses are allowed.

13.4.1 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DR[15:0] | | | | | | | | | | | | | | | |
| 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.

13.4.2 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

13.4.3 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. | RTYPE_OUT | RTYPE_IN | REV_OUT[1:0] | REV_IN[1:0] | | POLYSIZE[1:0] | Res. | Res. | RESSET | | rs |
| | | | | | rw | rw | rw | rw | | rw | rw | | | | |

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

Bit 10 **RTYPE_OUT**: Reverse type output

This bit controls the reversal granularity of the output data.

0: Bit level output

1: Byte or half-word level output

Bit 9 **RTYPE_IN**: Reverse type input

This bit controls the reversal granularity of the input data.

0: Bit level input

1: Byte or half-word level input

Bits 8:7 **REV_OUT[1:0]**: Reverse output data

This bitfield controls the reversal of the bit order of the output data.

00: Bit order not affected (RTYPE_OUT = 0 or 1)

01: Bit-reversed output format (RTYPE_OUT = 0) or half-word reversal done by word (RTYPE_OUT = 1)

10: Bit order not affected (RTYPE_OUT = 0) or byte reversal done by word (RTYPE_OUT = 1)

11: Bit order not affected (RTYPE_OUT = 0 or 1)

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 (RTYPE_IN = 0 or 1)

01: Bit reversal done by byte (RTYPE_IN = 0) or half-word reversal done by word (RTYPE_IN = 1)

10: Bit reversal done by half-word (RTYPE_IN = 0) or byte reversal done by word (RTYPE_IN = 1)

11: Bit reversal done by word (RTYPE_IN = 0) or bit order is not affected (RTYPE_IN = 1)

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

13.4.4 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.

13.4.5 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.

13.4.6 CRC register map

Table 63. 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 | | |
| 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 | | |
| 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 | Res | Res | Res | RES | RESET | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 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 | | |
| 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 | 1 | | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

14 Analog-to-digital converter (ADC)

14.1 Introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 20 multiplexed channels allowing it to measure signals from 16 external and 4 internal sources. 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 analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined higher or lower thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

A built-in hardware oversampler allows analog performances to be improved while off-loading the related computational burden from the CPU.

14.2 ADC main features

- High performance
 - 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
 - ADC conversion time: 0.4 μ s for 12-bit resolution (2.5Msps), faster conversion times can be obtained by lowering resolution.
 - Self-calibration
 - Programmable sampling time
 - Data alignment with built-in data coherency
 - DMA support
- Low-power
 - The application can reduce PCLK frequency for low-power operation while still keeping optimum ADC performance. For example, 0.4 μ s conversion time is kept, whatever the PCLK frequency
 - Wait mode: prevents ADC overrun in applications with low PCLK frequency
 - Auto off mode: ADC is automatically powered off except during the active conversion phase. This dramatically reduces the power consumption of the ADC.
- Analog input channels
 - 16 external analog inputs
 - 1 channel for internal temperature sensor (V_{SENSE})
 - 1 channel for internal reference voltage (V_{REFINT})
 - 1 channel for monitoring external V_{BAT} power supply pin
 - 1 channel for monitoring DAC internal channel input
- Start-of-conversion can be initiated:
 - By software
 - By hardware triggers with configurable polarity (timer events or GPIO input events)
- Conversion modes
 - 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 the end of sampling, end of conversion, end of sequence conversion, and in case of analog watchdog or overrun events
- Analog watchdog
- Oversampler
 - 16-bit data register
 - Oversampling ratio adjustable from 2 to 256x
 - Programmable data shift up to 8-bits
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

14.3 ADC implementation

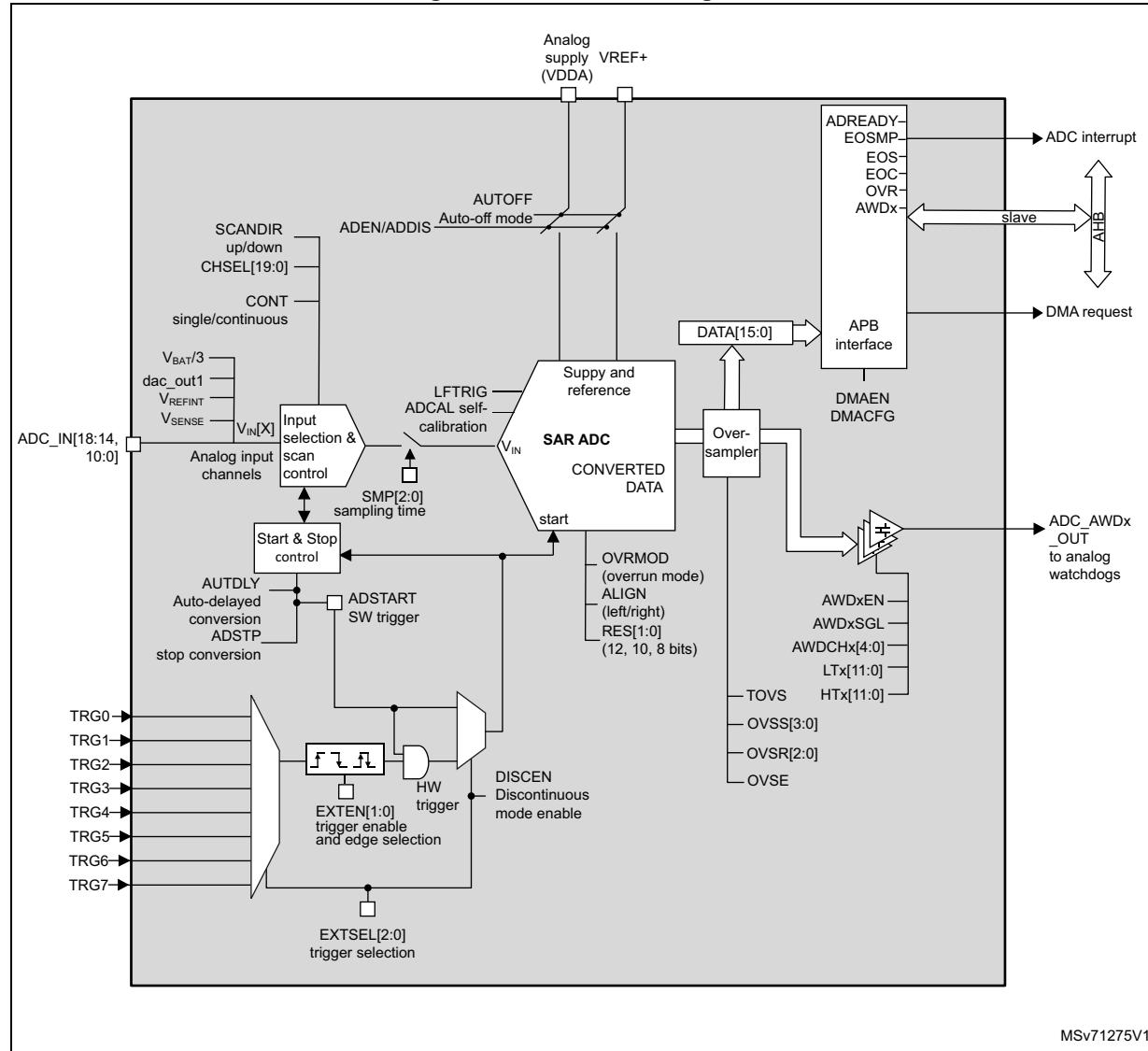
Table 64. ADC main features

| ADC modes/features | STM32U0xx |
|-----------------------------|-----------|
| Resolution | 12 bits |
| Maximum sampling speed | 2.5 Msps |
| Hardware offset calibration | X |
| Single-ended inputs | X |
| Differential inputs | - |
| Oversampling mode | X |
| Data register | 16 bits |
| DMA support | X |
| Number of analog watchdogs | 3 |
| Number of external channels | 16 |
| Number of internal channels | 4 |

14.4 ADC functional description

Figure 31 shows the ADC block diagram and Table 65 gives the ADC pin description.

Figure 31. ADC block diagram



MSv71275V1

1. TRGi are mapped at product level. Refer to [Table External triggers](#) in Section 14.4.1: ADC pins and internal signals.

14.4.1 ADC pins and internal signals

Table 65. ADC input/output pins

| Name | Signal type | Remarks |
|---------|----------------------------------|--|
| VDDA | Input, analog power supply | Analog power supply and positive reference voltage for the ADC |
| VSSA | Input, analog supply ground | Ground for analog power supply |
| VREF+ | Input, analog reference positive | The higher/positive reference voltage for the ADC. |
| ADC_INx | Analog input signals | 16 external analog input channels |

Table 66. ADC internal input/output signals

| Internal signal name | Signal type | Description |
|----------------------|-----------------------|---|
| $V_{IN[x]}$ | Analog Input channels | Connected either to internal channels or to ADC_IN/external channels |
| TRGx | Input | ADC conversion triggers |
| V_{SENSE} | Input | Internal temperature sensor output voltage |
| V_{REFINT} | Input | Internal voltage reference output voltage |
| $V_{BAT/3}$ | Input | VBAT pin input voltage divided by 3 |
| dac_out1 | Input | DAC internal channel1 input |
| ADC_AWDx_OUT | Output | Internal analog watchdog output signal connected to on-chip timers (x = Analog watchdog number = 1,2,3) |

Table 67. External triggers

| Name | Source | EXTSEL[2:0] |
|------|------------|-------------|
| TRG0 | TIM1_TRGO2 | 000 |
| TRG1 | TIM1_CC4 | 001 |
| TRG2 | TIM2_TRGO | 010 |
| TRG3 | TIM3_TRGO | 011 |
| TRG4 | TIM15_TRGO | 100 |
| TRG5 | TIM6_TRGO | 101 |
| TRG6 | Reserved | 110 |
| TRG7 | EXTI11 | 111 |

14.4.2 ADC voltage regulator (ADVREGEN)

The ADC has a specific internal voltage regulator which must be enabled and stable before using the ADC.

The ADC internal voltage regulator can be enabled by setting ADVREGEN bit to 1 in the ADC_CR register. The software must wait for the ADC voltage regulator startup time ($t_{ADCVREG_STUP}$) before launching a calibration or enabling the ADC. This delay must be managed by software (for details on $t_{ADCVREG_STUP}$, refer to the device datasheet).

After ADC operations are complete, the ADC is disabled (ADEN = 0). To keep power consumption low, it is important to disable the ADC voltage regulator before entering low-power mode (LPRun, LPSleep or Stop mode). Refer to [Section : ADC voltage regulator disable sequence](#).

Note: When the internal voltage regulator is disabled, the internal analog calibration is kept.

Analog reference from the power control unit

The internal ADC voltage regulator internally uses an analog reference delivered by the power control unit through a buffer. This buffer is always enabled when the main voltage regulator of the power control unit operates in normal Run mode (refer to Reset and clock control and power control sections).

If the main voltage regulator enters low-power mode (such as Low-power run mode), this buffer is disabled and the ADC cannot be used.

ADC Voltage regulator enable sequence

To enable the ADC voltage regulator, set ADVREGEN bit to 1 in ADC_CR register.

ADC voltage regulator disable sequence

To disable the ADC voltage regulator, follow the sequence below:

1. Make sure that the ADC is disabled (ADEN = 0).
2. Clear ADVREGEN bit in ADC_CR register.

14.4.3 Calibration (ADCAL)

The ADC has a calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is complete.

Calibration should be performed before starting A/D conversion. It removes the offset error which may vary from chip to chip due to process variation.

The calibration is initiated by software by setting bit ADCAL to 1. It can be initiated only when all the following conditions are met:

- the ADC voltage regulator is enabled (ADVREGEN = 1 and LDORDY = 1),
- the ADC is disabled (ADEN = 0), and
- the Auto-off mode is disabled (AUTOFF = 0).

ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. After this, the calibration factor can be read from the ADC_DR register (from bits 6 to 0).

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). When the ADC operating conditions change (V_{REF+} changes are the main contributor to ADC offset variations and temperature change to a lesser extend), it is recommended to re-run a calibration cycle.

The calibration factor is lost in the following cases:

- The power supply is removed from the ADC (for example when the product enters Standby or VBAT mode)
- The ADC peripheral is reset.

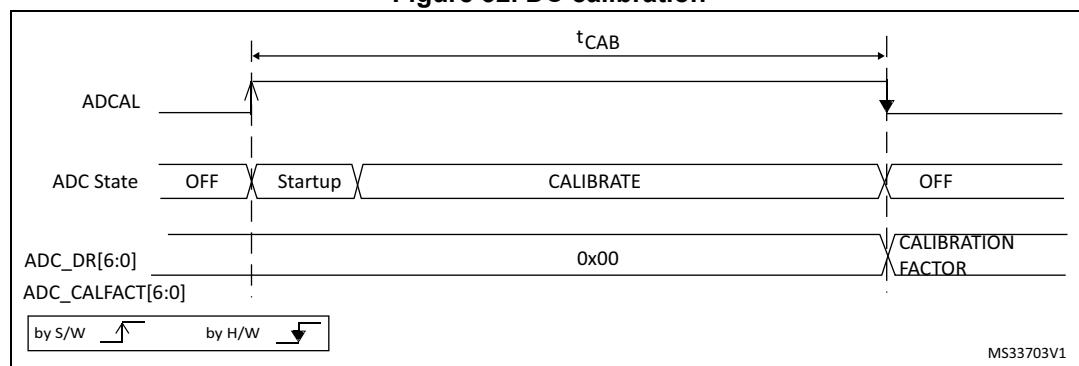
The calibration factor is lost each time power is removed from the ADC (for example when the product enters Standby or VBAT mode). Still, it is possible to save and restore the calibration factor by software to save time when re-starting the ADC (as long as temperature and voltage are stable during the ADC power-down).

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 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.

Software calibration procedure

1. Ensure that ADEN = 0, AUTOFF = 0, ADVREGEN = 1 and DMAEN = 0.
2. Set ADCAL = 1.
3. Wait until ADCAL = 0 (or until EOCAL = 1). This can be handled by interrupt if the interrupt is enabled by setting the EOCALIE bit in the ADC_IER register
4. The calibration factor can be read from bits 6:0 of ADC_DR or ADC_CALFACT registers.
5. To reduce the noise effect of the calibration factor extraction, the software can make average of eight CALFACT[6:0] values (optional).

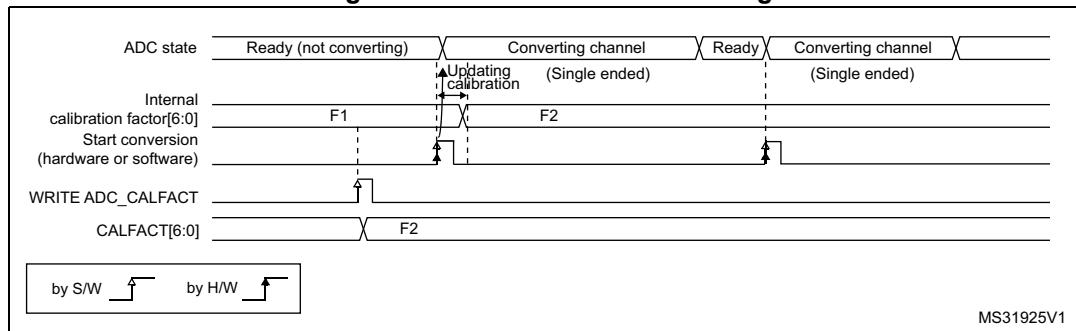
Figure 32. DC calibration



Calibration factor forcing software procedure

1. Ensure that ADEN = 1 and ADSTART = 0 (ADC started with no conversion ongoing)
2. Write ADC_CALFACT with the saved calibration factor
3. The calibration factor is used as soon as a new conversion is launched.

Figure 33. Calibration factor forcing



14.4.4 ADC on-off control (ADEN, ADDIS, ADRDY)

At power-up, the ADC is disabled and put in power-down mode (ADEN = 0).

As shown in [Figure 34](#), the ADC needs a stabilization time of t_{STAB} before it starts converting accurately.

Two control bits are used to enable or disable the ADC:

- Set ADEN = 1 to enable the ADC. The ADRDY flag is set as soon as the ADC is ready for operation.
- Set ADDIS = 1 to disable the ADC and put the ADC in power down mode. The ADEN and ADDIS bits are then automatically cleared by hardware as soon as the ADC is fully disabled.

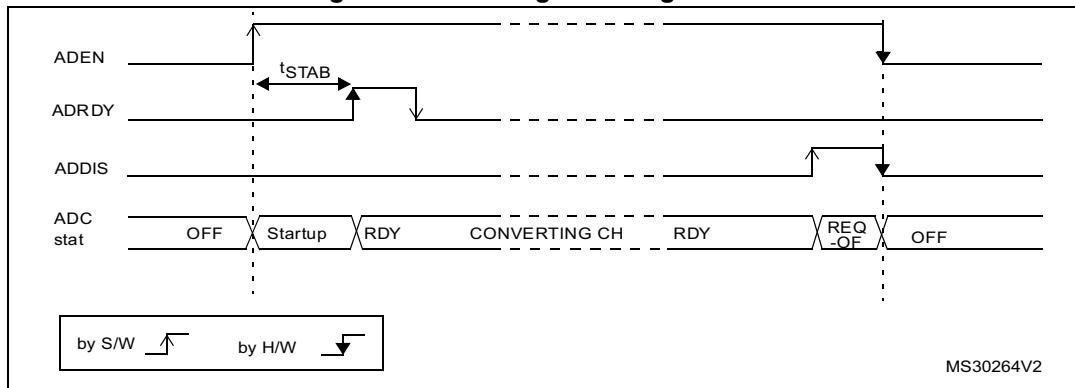
Conversion can then start either by setting ADSTART to 1 (refer to [Section 14.5: Conversion on external trigger and trigger polarity \(EXTSEL, EXTEN\)](#)) or when an external trigger event occurs if triggers are enabled.

Follow this procedure to enable the ADC:

1. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1.
2. Set ADEN = 1 in the ADC_CR register.
3. Wait until ADRDY = 1 in the ADC_ISR register (ADRDY is set after the ADC startup time). This can be handled by interrupt if the interrupt is enabled by setting the ADRDYIE bit in the ADC_IER register.

Follow this procedure to disable the ADC:

1. Check that ADSTART = 0 in the ADC_CR register to ensure that no conversion is ongoing. If required, stop any ongoing conversion by writing 1 to the ADSTP bit in the ADC_CR register and waiting until this bit is read at 0.
2. Set ADDIS = 1 in the ADC_CR register.
3. If required by the application, wait until ADEN = 0 in the ADC_CR register, indicating that the ADC is fully disabled (ADDIS is automatically reset once ADEN = 0).
4. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1 (optional).

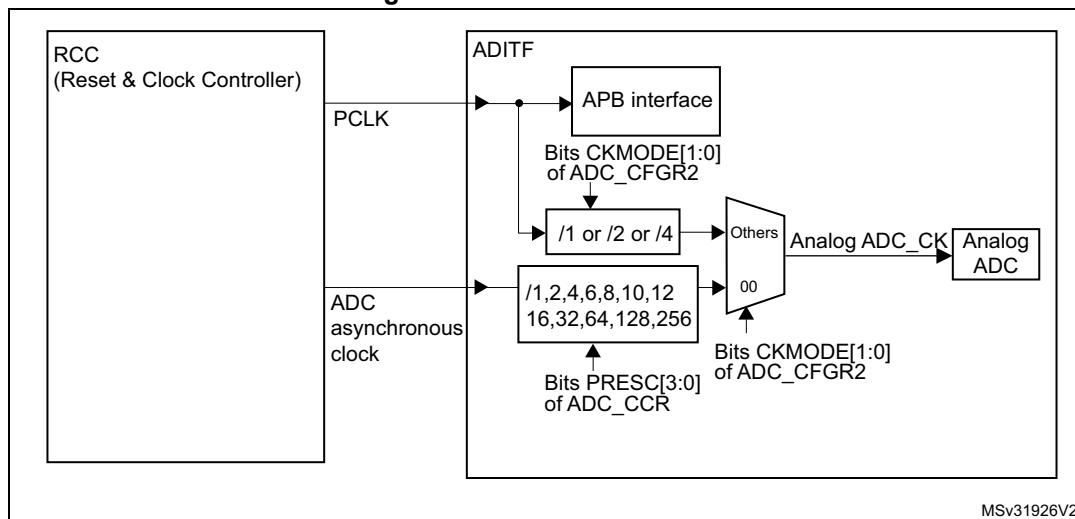
Figure 34. Enabling/disabling the ADC

Note: In Auto-off mode ($AUTOFF = 1$) the power-on/off phases are performed automatically, by hardware and the *ADRDY* flag is not set.

When the bus clock is much faster than the analog *ADC_CK* clock, a minimum delay of ten analog *ADC_CK* cycles must be respected between *ADEN* and *ADDIS* bit settings.

14.4.5 ADC clock (CKMODE, PRESC[3:0])

The ADC has a dual clock-domain architecture, so that the ADC can be fed with a clock (ADC asynchronous clock) independent from the APB clock (PCLK).

Figure 35. ADC clock scheme

1. Refer to *Section Reset and clock control (RCC)* for how the PCLK clock and ADC asynchronous clock are enabled.

The input clock of the analog ADC can be selected between two different clock sources (see [Figure 35: ADC clock scheme](#) to see how the PCLK clock and the ADC asynchronous clock are enabled):

- a) The ADC clock can be a specific clock source, named “ADC asynchronous clock” which is independent and asynchronous with the APB clock.
Refer to RCC Section for more information on generating this clock source.
To select this scheme, bits CKMODE[1:0] of the ADC_CFGR2 register must be reset.
- b) The ADC clock can be derived from the APB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4) according to bits CKMODE[1:0].
To select this scheme, bits CKMODE[1:0] of the ADC_CFGR2 register must be different from “00”.

In option a), the generated ADC clock can eventually be divided by a prescaler (1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128, 256) when programming the bits PRESC[3:0] in the ADC_CCR register).

Option a) has the advantage of reaching the maximum ADC clock frequency whatever the APB clock scheme selected.

Option b) 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 is added by the resynchronizations between the two clock domains).

Table 68. Latency between trigger and start of conversion⁽¹⁾

| ADC clock source | CKMODE[1:0] | Latency between the trigger event and the start of conversion |
|--|-------------|--|
| SYSCLK, PLLPCLK, or HSI16 clock ⁽²⁾ | 00 | Latency is not deterministic (jitter) |
| PCLK divided by 2 | 01 | Latency is deterministic (no jitter) and equal to 3.25 ADC clock cycles |
| PCLK divided by 4 | 10 | Latency is deterministic (no jitter) and equal to 3.125 ADC clock cycles |
| PCLK divided by 1 | 11 | Latency is deterministic (no jitter) and equal to 3 ADC clock cycles |

1. Refer to the device datasheet for the maximum ADC_CLK frequency.

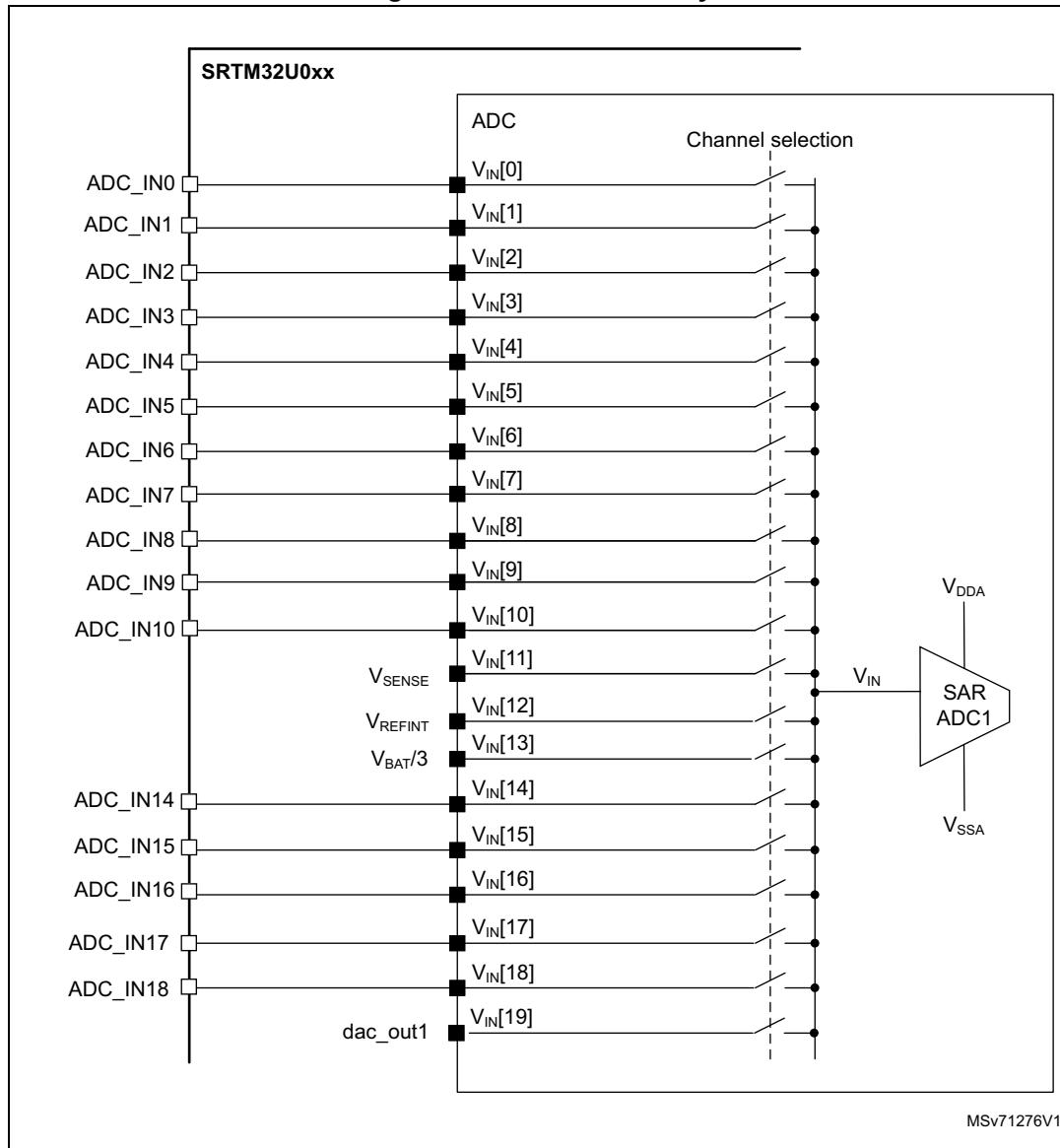
2. Selected with ADCSEL bitfield of the RCC_CCIPR register.

Caution: When selecting CKMODE[1:0] = 11 (PCLK divided by 1), the user must ensure that the PCLK has a 50% duty cycle. This is done by selecting a system clock with a 50% duty cycle and configuring the APB prescaler in bypass modes in the RCC (refer to the Reset and clock controller section). If an internal source clock is selected, the AHB and APB prescalers do not divide the clock.

14.4.6 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described in [Figure 36](#).

Figure 36. ADC connectivity



14.4.7 Configuring the ADC

The software must write the ADCAL and ADEN bits in the ADC_CR register and configure the ADC_CFGR1 and ADC_CFGR2 registers only when the ADC is disabled (ADEN must be cleared).

The software must only write to the ADSTART and ADDIS bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN = 1 and ADDIS = 0).

For all the other control bits in the ADC_IER, ADC_SMPR, ADC_CHSELR and ADC_CCR registers, refer to the description of the corresponding control bit in [Section 14.13: ADC registers](#).

ADC_AWDTRx registers can be modified when conversion is ongoing.

The software must only write to the ADSTP bit in the ADC_CR register if the ADC is enabled (and possibly converting) and there is no pending request to disable the ADC (ADSTART = 1 and ADDIS = 0).

Note: *There is no hardware protection preventing software from making write operations forbidden by the above rules. If such a forbidden write access occurs, the ADC may enter an undefined state. To recover correct operation in this case, the ADC must be disabled (clear ADEN = 0 and all the bits in the ADC_CR register).*

14.4.8 Channel selection (CHSEL, SCANDIR, CHSELRMOD)

There are up to 20 multiplexed channels:

- 16 analog inputs from GPIO pins (ADC_INx)
- 4 internal analog inputs (temperature sensor, internal reference voltage, DAC internal channel1, V_{BAT} channel)

It is possible to convert a single channel or a sequence of channels.

The sequence of the channels to be converted can be programmed in the ADC_CHSELR channel selection register: each analog input channel has a dedicated selection bit (CHSELx).

The ADC scan sequencer can be used in two different modes:

- Sequencer not fully configurable:
 - The order in which the channels are scanned is defined by the channel number (CHSELRMOD bit must be cleared in ADC_CFGR1 register):
 - Sequence length configured through CHSELx bits in ADC_CHSELR register
 - Sequence direction: the channels are scanned in a forward direction (from the lowest to the highest channel number) or backward direction (from the highest to the lowest channel number) depending on the value of SCANDIR bit (SCANDIR = 0: forward scan, SCANDIR = 1: backward scan)
 - Any channel can belong to in these sequences
- Sequencer fully configurable

The CHSELRMOD bit is set in ADC_CFGR1 register.

- Sequencer length is up to 8 channels
- The order in which the channels are scanned is independent from the channel number. Any order can be configured through SQ1[3:0] to SQ8[3:0] bits in ADC_CHSELR register.
- Only channel 0 to channel 14 can be selected in this sequence
- If the sequencer detects SQx[3:0] = 0b1111, the following SQx[3:0] registers are ignored.
- If no 0b1111 is programmed in SQx[3:0], the sequencer scans full eight channels.

After programming ADC CHSELR, SCANDIR and CHSELRMOD bits, it is mandatory to wait for CCRDY flag before starting conversions. It indicates that the new channel setting has

been applied. If a new configuration is required, the CCRDY flag must be cleared prior to starting the conversion.

The software is allowed to program the CHSEL, SCANDIR, CHSELRMOD bits only when ADSTART bit is cleared (which ensures that no conversion is ongoing).

Temperature sensor, DAC output, V_{REFINT} and V_{BAT} internal channels

The temperature sensor is connected to channel ADC V_{IN}[11].

The internal voltage reference V_{REFINT} is connected to channel ADC V_{IN}[12].

V_{BAT} channel is connected to ADC V_{IN}[13] channel.

The internal dac_out1 output voltage is connected to ADC V_{IN}[19] channel.

When V_{REF+} is lower than V_{DDA}, this channel is not converted.

14.4.9 Programmable sampling time (SMPx[2:0])

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the sample and hold capacitor to the input voltage level.

Having a programmable sampling time allows the conversion speed to be trimmed according to the input resistance of the input voltage source.

The ADC samples the input voltage for a number of ADC clock cycles that can be modified using the SMP1[2:0] and SMP2[2:0] bits in the ADC_SMPR register.

Each channel can choose one out of two sampling times configured in SMP1[2:0] and SMP2[2:0] bitfields, through SMPSELx bits in ADC_SMPR register.

The total conversion time is calculated as follows:

$$t_{\text{CONV}} = \text{Sampling time} + 12.5 \times \text{ADC clock cycles}$$

Example:

With ADC_CLK = 16 MHz and a sampling time of 1.5 ADC clock cycles:

$$t_{\text{CONV}} = 1.5 + 12.5 = 14 \text{ ADC clock cycles} = 0.875 \mu\text{s}$$

The ADC indicates the end of the sampling phase by setting the EOSMP flag.

14.4.10 Single conversion mode (CONT = 0)

In Single conversion mode, the ADC performs a single sequence of conversions, converting all the channels once. This mode is selected when CONT = 0 in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in 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 the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

14.4.11 Continuous conversion mode (CONT = 1)

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting all the channels once and then automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when CONT = 1 in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in 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 bits DISCEN = 1 and CONT = 1.

14.4.12 Starting conversions (ADSTART)

Software starts ADC conversions by setting ADSTART = 1.

When ADSTART is set, the conversion:

- Starts immediately if EXTN = 00 (software trigger)
- At the next active edge of the selected hardware trigger if EXTN ≠ 00

The ADSTART bit is also used to indicate whether an ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0, indicating that the ADC is idle.

The ADSTART bit is cleared by hardware:

- In single mode with software trigger (CONT = 0, EXTEN = 00)
 - At any end of conversion sequence (EOS = 1)
- In discontinuous mode with software trigger (CONT = 0, DISCEN = 1, EXTEN = 00)
 - At end of conversion (EOC = 1)
- In all cases (CONT = x, EXTEN = XX)
 - After execution of the ADSTP procedure invoked by software (see [Section 14.4.14: Stopping an ongoing conversion \(ADSTP\) on page 341](#))

Note:

In continuous mode (CONT = 1), the ADSTART bit is not cleared by hardware when the EOS flag is set because the sequence is automatically relaunched.

When hardware trigger is selected in single mode (CONT = 0 and EXTEN = 01), ADSTART is not cleared by hardware when the EOS flag is set (except if DMAEN = 1 and DMACFG = 0 in which case ADSTART is cleared at end of the DMA transfer). This avoids the need for software having to set the ADSTART bit again and ensures the next trigger event is not missed.

After changing channel selection configuration (by programming ADC_CHSEL register or changing CHSELRMOD or SCANDIR), it is mandatory to wait until CCRDY flag is asserted before asserting ADSTART, otherwise the value written to ADSTART is ignored.

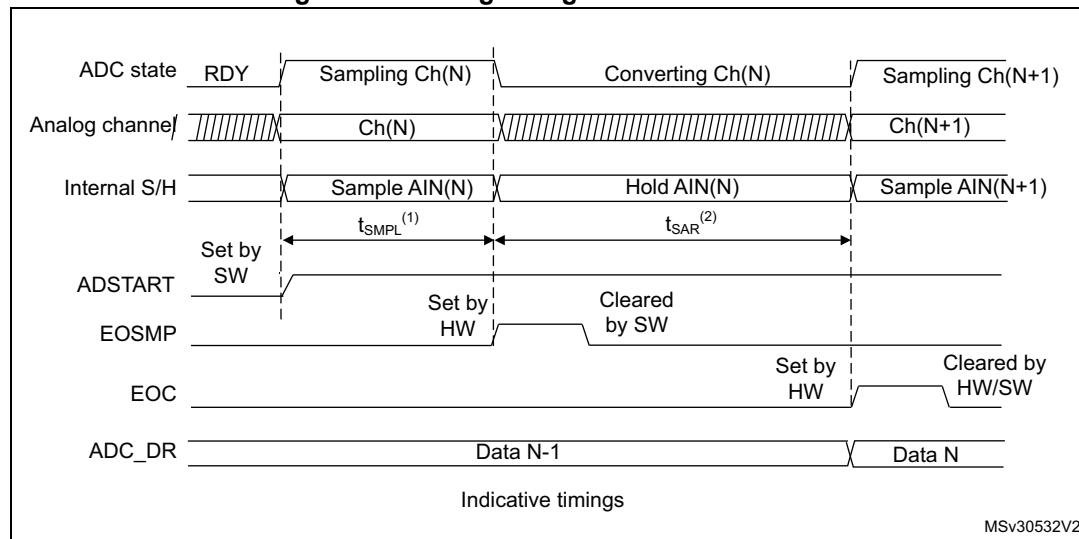
14.4.13 Timings

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:

$$t_{\text{CONV}} = t_{\text{SMPL}} + t_{\text{SAR}} = [1.5 \text{ ns}_{\text{min}} + 12.5 \text{ ns}_{\text{12bit}}] \times t_{\text{ADC_CLK}}$$

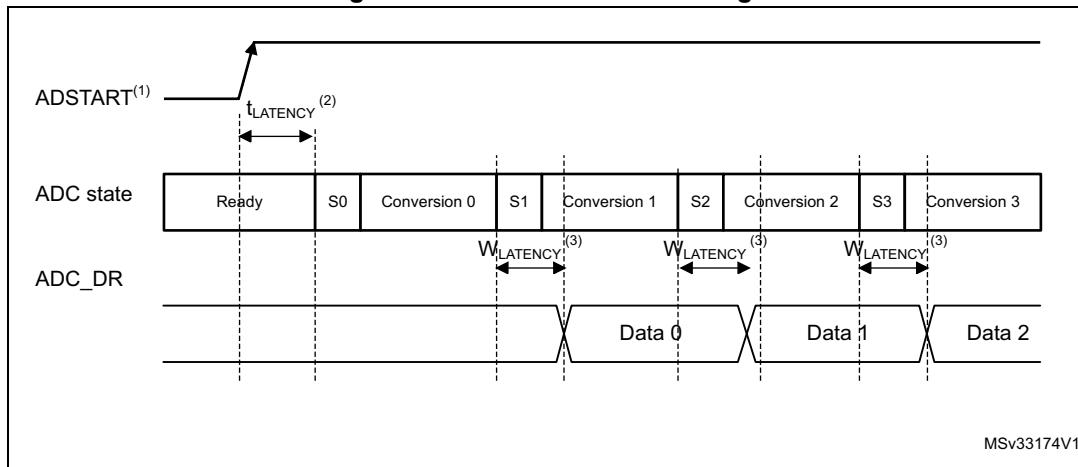
$$t_{\text{CONV}} = t_{\text{SMPL}} + t_{\text{SAR}} = 42.9 \text{ ns}_{\text{min}} + 357.1 \text{ ns}_{\text{12bit}} = 0.400 \mu\text{s}_{\text{min}} \text{ (for } f_{\text{ADC_CLK}} = 35 \text{ MHz)}$$

Figure 37. Analog-to-digital conversion time



1. t_{SMPL} depends on SMP[2:0].

2. t_{SAR} depends on RES[2:0].

Figure 38. ADC conversion timings

1. EXTEN = 00 or EXTEN ≠ 00
2. Trigger latency (refer to datasheet for more details)
3. ADC_DR register write latency (refer to datasheet for more details)

14.4.14 Stopping an ongoing conversion (ADSTP)

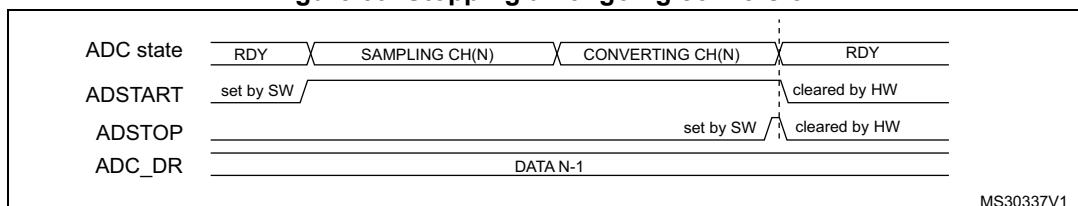
The software can decide to stop any ongoing conversions by setting ADSTP = 1 in the ADC_CR register.

This resets the ADC operation and the ADC is idle, ready for a new operation.

When the ADSTP bit is set by software, any ongoing conversion is aborted and the result is discarded (ADC_DR register is not updated with the current conversion).

The scan sequence is also aborted and reset (meaning that restarting the ADC would restart a new sequence).

Once this procedure is complete, the ADSTP and ADSTART bits are both cleared by hardware and the software must wait until ADSTART=0 before starting new conversions.

Figure 39. Stopping an ongoing conversion

14.5 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)

A conversion or a sequence of conversion can be triggered either by software or by an external event (for example timer capture). If the EXTEN[1:0] control bits are not equal to “0b00”, then external events are able to trigger a conversion with the selected polarity. The trigger selection is effective once software has set bit ADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

If bit ADSTART = 0, any hardware triggers which occur are ignored.

[Table 69](#) provides the correspondence between the EXTEN[1:0] values and the trigger polarity.

Table 69. Configuring the trigger polarity

| Source | EXTEN[1:0] |
|--|------------|
| Trigger detection disabled | 00 |
| Detection on rising edge | 01 |
| Detection on falling edge | 10 |
| Detection on both rising and falling edges | 11 |

Note: *The polarity of the external trigger can be changed only when the ADC is not converting (ADSTART = 0).*

The EXTSEL[2:0] control bits are used to select which of 8 possible events can trigger conversions.

Refer to [Table 67: External triggers](#) in [Section 14.4.1: ADC pins and internal signals](#) for the list of all the external triggers that can be used for regular conversion.

The software source trigger events can be generated by setting the ADSTART bit in the ADC_CR register.

Note: *The trigger selection can be changed only when the ADC is not converting (ADSTART = 0).*

14.5.1 Discontinuous mode (DISCEN)

This mode is enabled by setting the DISCEN bit in the ADC_CFGR1 register.

In this mode (DISCEN = 1), a hardware or software trigger event is required to start each conversion defined in the sequence. On the contrary, if DISCEN = 0, a single hardware or software trigger event successively starts all the conversions defined in the sequence.

Example:

- DISCEN = 1, channels to be converted = 0, 3, 7, 10
 - 1st trigger: channel 0 is converted and an EOC event is generated
 - 2nd trigger: channel 3 is converted and an EOC event is generated
 - 3rd trigger: channel 7 is converted and an EOC event is generated
 - 4th trigger: channel 10 is converted and both EOC and EOS events are generated.
 - 5th trigger: channel 0 is converted and an EOC event is generated
 - 6th trigger: channel 3 is converted and an EOC event is generated
 - ...
- DISCEN = 0, channels to be converted = 0, 3, 7, 10
 - 1st trigger: the complete sequence is converted: channel 0, then 3, 7 and 10. Each conversion generates an EOC event and the last one also generates an EOS event.
 - Any subsequent trigger events restarts the complete sequence.

Note: *It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.*

14.5.2 Programmable resolution (RES) - Fast conversion mode

It is possible to obtain faster conversion times (t_{SAR}) by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] bits in the ADC_CFGR1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

Note: *The RES[1:0] bit must only be changed when the ADEN bit is reset.*

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeros.

Lower resolution reduces the conversion time needed for the successive approximation steps as shown in [Table 70](#).

Table 70. t_{SAR} timings depending on resolution

| RES[1:0] bits | t_{SAR} (ADC clock cycles) | t_{SAR} (ns) at $f_{ADC} = 35$ MHz | t_{SMPL} (min) (ADC clock cycles) | t_{CONV} (ADC clock cycles) (with min. t_{SMPL}) | t_{CONV} (ns) at $f_{ADC} = 35$ MHz |
|------------------|------------------------------------|---|---|---|--|
| 12 | 12.5 | 357 | 1.5 | 14 | 400 |
| 10 | 10.5 | 300 | 1.5 | 12 | 343 |
| 8 | 8.5 | 243 | 1.5 | 10 | 286 |
| 6 | 6.5 | 186 | 1.5 | 8 | 229 |

14.5.3 End of conversion, end of sampling phase (EOC, EOSMP flags)

The ADC indicates each end of conversion (EOC) event.

The ADC sets the EOC flag in the ADC_ISR register as soon as a new conversion data result is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set in the ADC_IER register. The EOC flag is cleared by software either by writing 1 to it, or by reading the ADC_DR register.

The ADC also indicates the end of sampling phase by setting the EOSMP flag in the ADC_ISR register. The EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC_IER register.

The aim of this interrupt is to allow the processing to be synchronized with the conversions. Typically, an analog multiplexer can be accessed in hidden time during the conversion phase, so that the multiplexer is positioned when the next sampling starts.

Note: As there is only a very short time left between the end of the sampling and the end of the conversion, it is recommended to use polling or a WFE instruction rather than an interrupt and a WFI instruction.

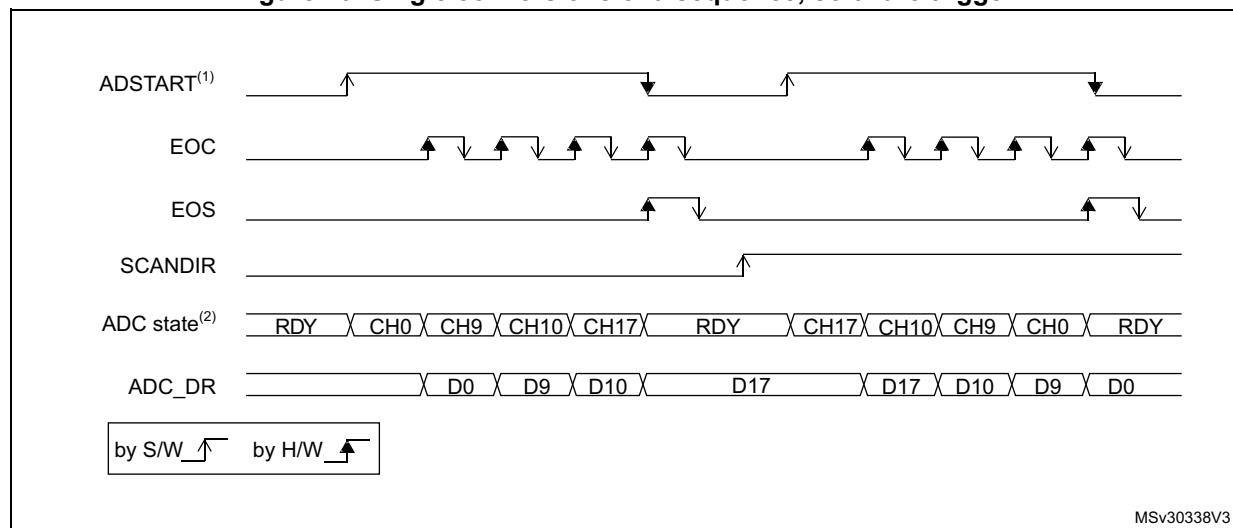
14.5.4 End of conversion sequence (EOS flag)

The ADC notifies the application of each end of sequence (EOS) event.

The ADC sets the EOS flag in the ADC_ISR register as soon as the last data result of a conversion sequence is available in the ADC_DR register. An interrupt can be generated if the EOSIE bit is set in the ADC_IER register. The EOS flag is cleared by software by writing 1 to it.

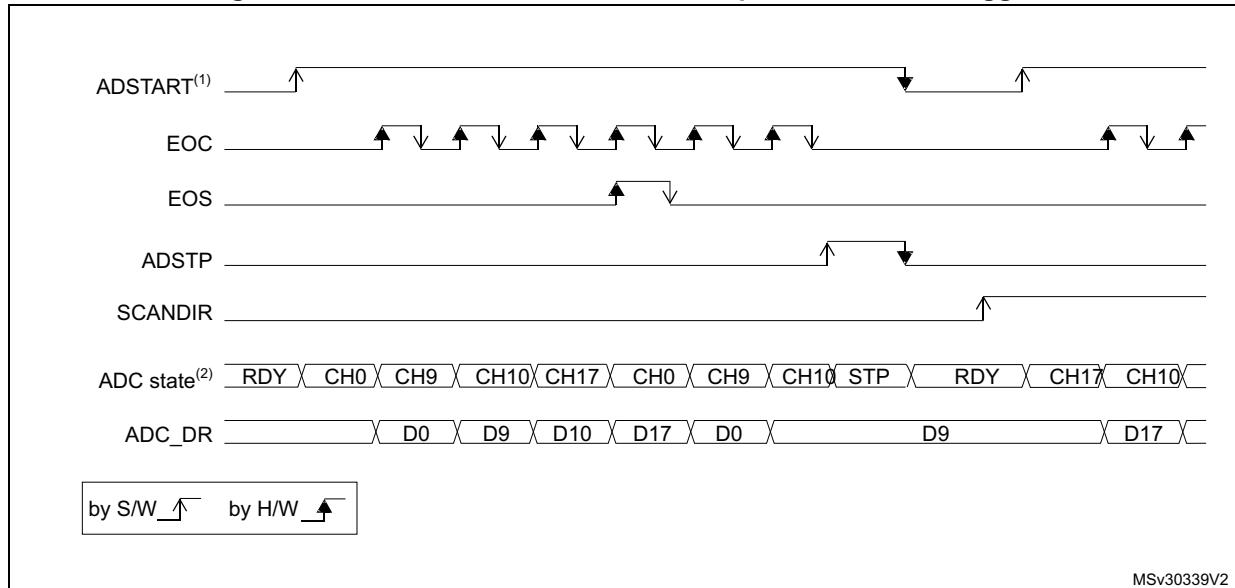
14.5.5 Example timing diagrams (single/continuous modes hardware/software triggers)

Figure 40. Single conversions of a sequence, software trigger

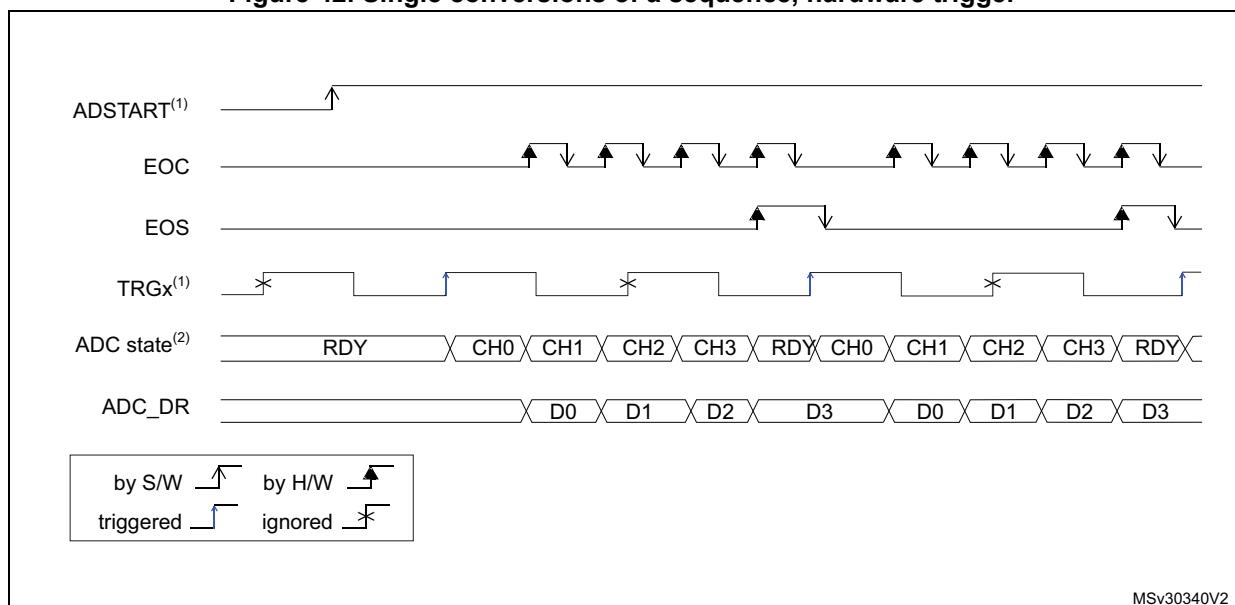


1. EXLEN = 00, CONT = 0

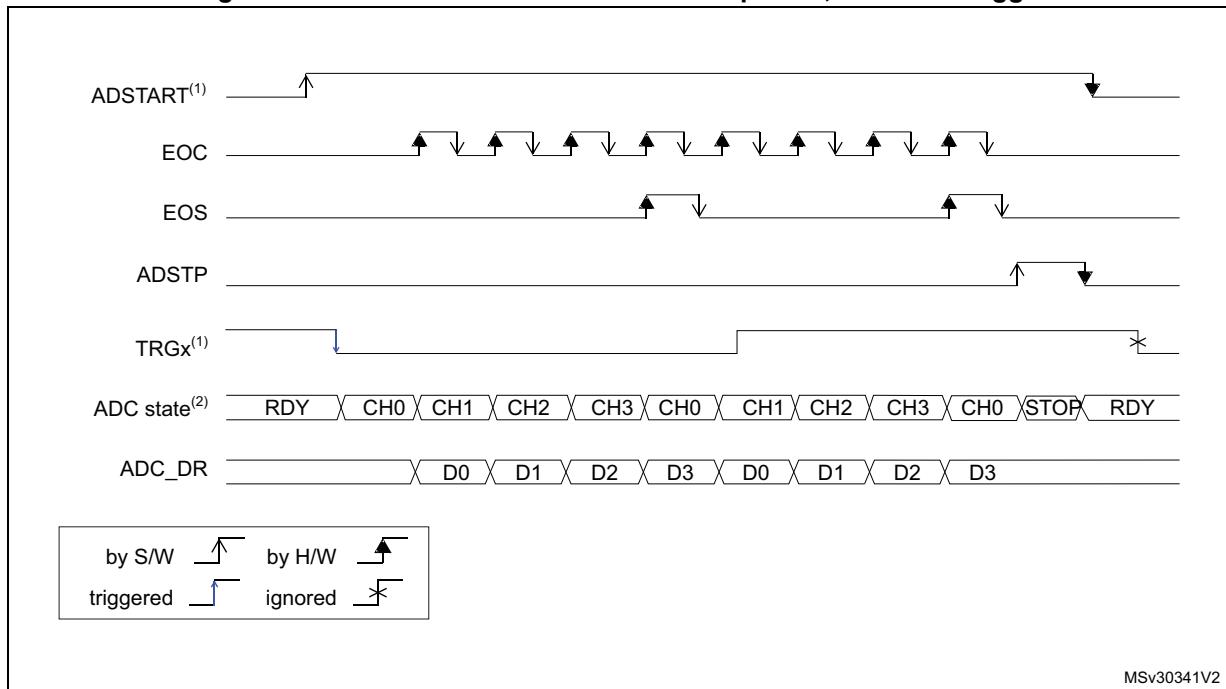
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

Figure 41. Continuous conversion of a sequence, software trigger

1. EXTEN = 00, CONT = 1,
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

Figure 42. Single conversions of a sequence, hardware trigger

1. EXTSEL = TRGx (over-frequency), EXTEN = 01 (rising edge), CONT = 0
2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0

Figure 43. Continuous conversions of a sequence, hardware trigger

MSv30341V2

1. EXTSEL = TRGx, EXTEN = 10 (falling edge), CONT = 1

2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0

14.5.6 Low frequency trigger mode

Once the ADC is enabled or the last ADC conversion is complete, the ADC is ready to start a new conversion. The ADC needs to be started at a predefined time (t_{idle}) otherwise ADC converted data might be corrupted due to the transistor leakage (refer to the device datasheet for the maximum value of t_{idle}).

If the application has to support a time longer than the maximum t_{idle} value (between one trigger to another for single conversion mode or between the ADC enable and the first ADC conversion), then the ADC internal state needs to be rearmed. This mechanism can be enabled by setting LFTRIG bit to 1 in ADC_CFGR2 register. By setting this bit, any trigger (software or hardware) sends a rearm command to ADC. The conversion starts after a one ADC clock cycle delay compared to LFTRIG cleared.

It is not necessary to use this mode when AUTOFF bit is set. For Wait mode, only the first trigger generates an internal rearm command.

14.6 Data management

14.6.1 Data register and data alignment (ADC_DR, ALIGN)

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC_DR data register which is 16-bit wide.

The format of the ADC_DR depends on the configured data alignment and resolution.

The ALIGN bit in the ADC_CFGR1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN = 0) or left-aligned (ALIGN = 1) as shown in [Figure 44](#).

Figure 44. Data alignment and resolution (oversampling disabled: OVSE = 0)

| ALIGN | RES | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|-----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---------|
| 0 | 0x0 | 0x0 | | | | | | | | | | | | | | | |
| | 0x1 | 0x00 | | | | | | | | | | | | | | | DR[9:0] |
| | 0x2 | 0x00 | | | | | | | | | | | | | | | DR[7:0] |
| | 0x3 | 0x00 | | | | | | | | | | | | | | | DR[5:0] |
| 1 | 0x0 | DR[11:0] | | | | | | | | | | | | | | | 0x0 |
| | 0x1 | DR[9:0] | | | | | | | | | | | | | | | 0x00 |
| | 0x2 | DR[7:0] | | | | | | | | | | | | | | | 0x00 |
| | 0x3 | 0x00 | | | | | | | | | | | | | | | 0x0 |

MS30342V1

14.6.2 ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) indicates a data overrun event, when the converted data was not read in time by the CPU or the DMA, before the data from a new conversion is available.

The OVR flag is set in the ADC_ISR register if the EOC flag is still at '1' at the time when a new conversion completes. An interrupt can be generated if the OVRIE bit is set in the ADC_IER register.

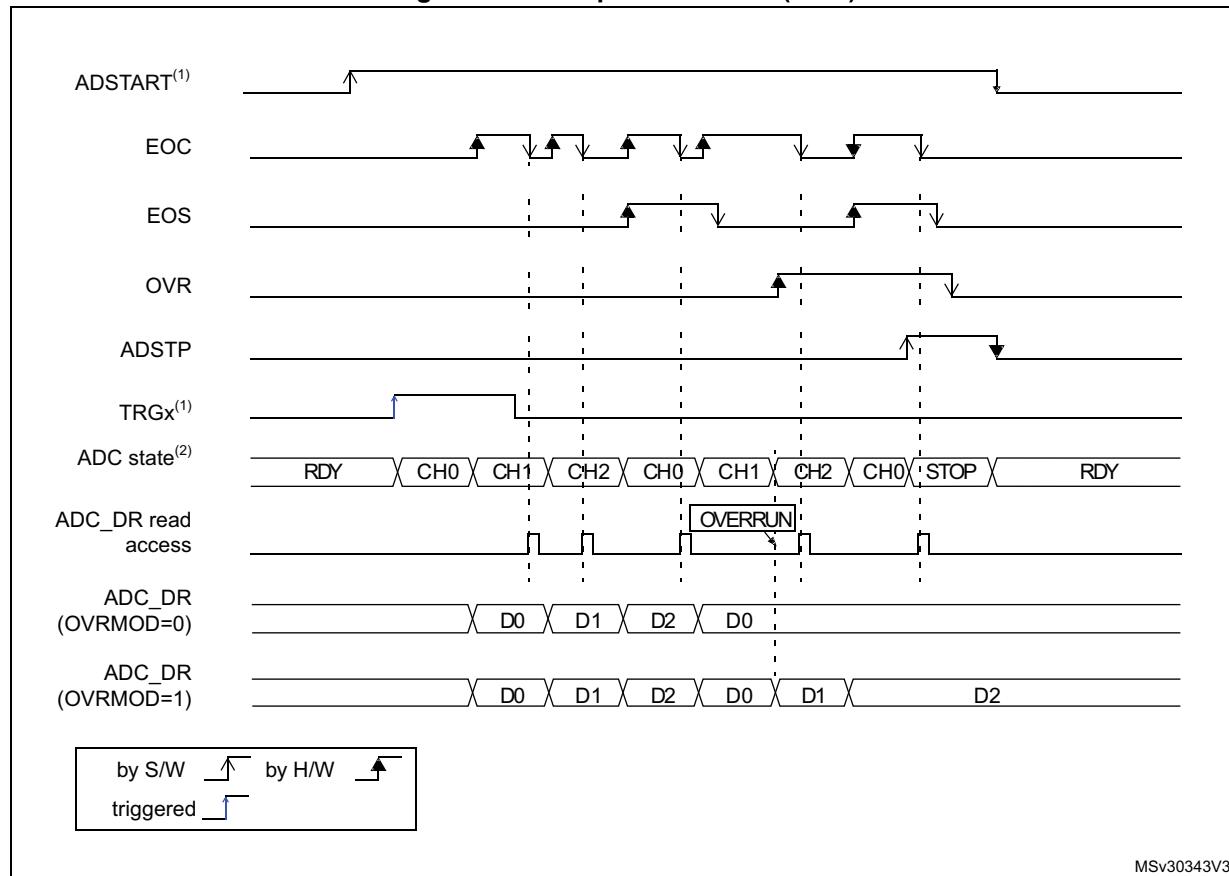
When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC_CR register.

The OVR flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or overwritten when an overrun event occurs by programming the OVRMOD bit in the ADC_CFGR1 register:

- OVRMOD = 0
 - An overrun event preserves the data register from being overwritten: the old data is maintained and the new conversion is discarded. If OVR remains at 1, further conversions can be performed but the resulting data is 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, further conversions can be performed and the ADC_DR register always contains the data from the latest conversion.

Figure 45. Example of overrun (OVR)



14.6.3 Managing a sequence of data converted without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software must use the EOC flag and its associated interrupt to handle each data result. Each time a conversion is complete, the EOC bit is set in the ADC_ISR register and the ADC_DR register can be read. The OVRMOD bit in the ADC_CFGR1 register should be configured to 0 to manage overrun events as an error.

14.6.4 Managing converted data without using the DMA without overrun

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the OVRMOD bit must be configured at 1 and the OVR flag should be ignored by the software. When OVRMOD = 1, an overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion data.

14.6.5 Managing converted data using the DMA

Since all converted channel values are stored in a single data register, it is efficient to use DMA when converting more than one channel. This avoids losing the conversion data results stored in the ADC_DR register.

When DMA mode is enabled (DMAEN bit set in the ADC_CFGR1 register), a DMA request is generated after the conversion of each channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Note: *The DMAEN bit in the ADC_CFGR1 register must be set after the ADC calibration phase.*

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 14.6.2: ADC overrun \(OVR, OVRMOD\) on page 347](#)).

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 in the ADC_CFGR1 register:

- DMA one shot mode (DMACFG = 0).
This mode should be selected when the DMA is programmed to transfer a fixed number of data words.
- DMA circular mode (DMACFG = 1)
This mode should be selected when programming the DMA in circular mode or double buffer mode.

DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs, see [Section 9: Direct memory access controller \(DMA\) on page 264](#)) 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 and its 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.
- The 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 word is available in the data register, even if the DMA has reached the last DMA transfer. This allows the DMA to be configured in circular mode to handle a continuous analog input data stream.

14.7 Low-power features

14.7.1 Wait mode conversion

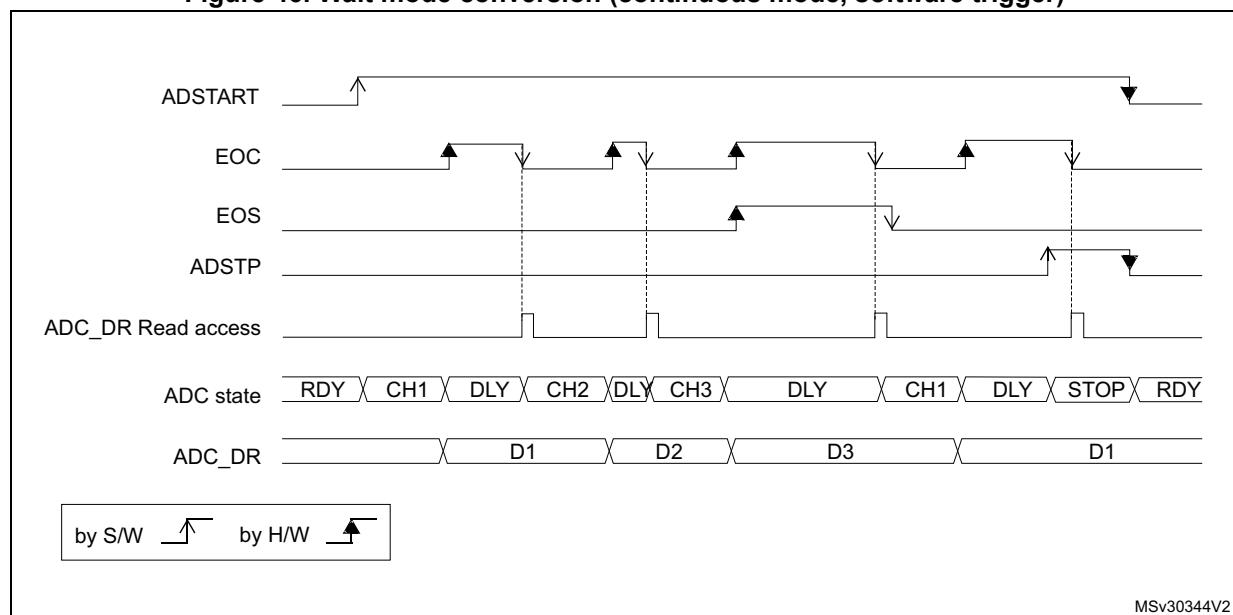
Wait mode conversion can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring.

When the WAIT bit is set in the ADC_CFGR1 register, a new conversion can start only if the previous data has been treated, once the ADC_DR register has been read or if the EOC bit has been cleared.

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data.

Note: Any hardware triggers which occur while a conversion is ongoing or during the wait time preceding the read access are ignored.

Figure 46. Wait mode conversion (continuous mode, software trigger)



1. EXTEN = 00, CONT = 1
2. CHSEL = 0x3, SCANDIR = 0, WAIT = 1, AUTOFF = 0

14.7.2 Auto-off mode (AUTOFF)

The ADC has an automatic power management feature which is called auto-off mode, and is enabled by setting AUTOFF = 1 in the ADC_CFGR1 register.

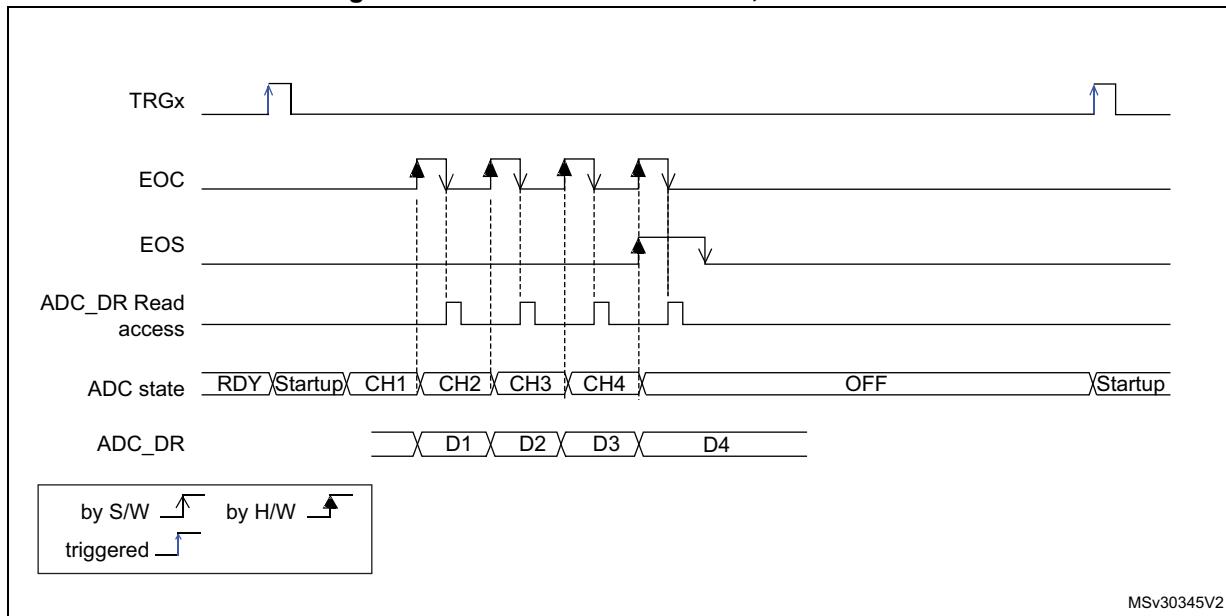
When AUTOFF = 1, the ADC is always powered off when not converting and automatically wakes-up when a conversion is started (by software or hardware trigger). A startup-time is automatically inserted between the trigger event which starts the conversion and the sampling time of the ADC. The ADC is then automatically disabled once the sequence of conversions is complete.

Auto-off mode can cause a dramatic reduction in the power consumption of applications which need relatively few conversions or when conversion requests are timed far enough

apart (for example with a low frequency hardware trigger) to justify the extra power and extra time used for switching the ADC on and off.

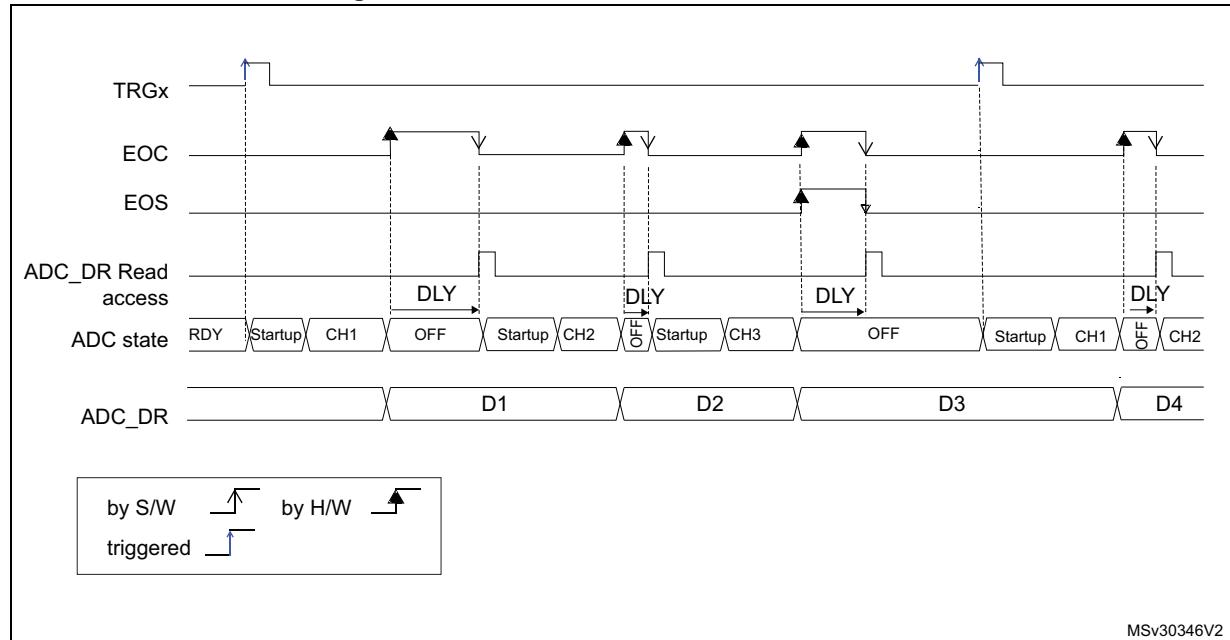
Auto-off mode can be combined with the wait mode conversion (WAIT = 1) for applications clocked at low frequency. This combination can provide significant power savings if the ADC is automatically powered-off during the wait phase and restarted as soon as the ADC_DR register is read by the application (see [Figure 48: Behavior with WAIT = 1, AUTOFF = 1](#)).

Figure 47. Behavior with WAIT = 0, AUTOFF = 1



- EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 1, AUTOFF = 1

Figure 48. Behavior with WAIT = 1, AUTOFF = 1



1. EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 1, AUTOFF = 1

14.8 Analog window watchdogs

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

14.8.1 Description of analog watchdog 1

AWD1 analog watchdog is enabled by setting the AWD1EN bit in the ADC_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels (see [Table 72: Analog watchdog 1 channel selection](#)) remain within a configured voltage range (window) as shown in [Figure 49](#).

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 HT1[11:0] and LT1[11:0] bits of ADC_AWD1TR register. An interrupt can be enabled by setting the AWD1IE bit in the ADC_IER register.

The AWD1 flag is cleared by software by programming it to 1.

When converting data with a resolution of less than 12-bit (according to bits DRES[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).

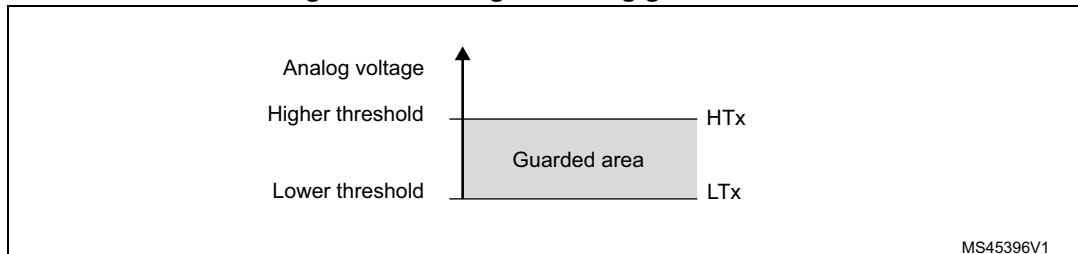
[Table 71](#) describes how the comparison is performed for all the possible resolutions.

Table 71. Analog watchdog comparison

| Resolution bits RES[1:0] | Analog Watchdog comparison between: | | Comments |
|--------------------------------|--|-------------------------|---|
| | Raw converted data, left aligned ⁽¹⁾ | Thresholds | |
| 00: 12-bit | DATA[11:0] | LTx[11:0] and HTx[11:0] | - |
| 01: 10-bit | DATA[11:2],00 | LTx[11:0] and HTx[11:0] | The user must configure LTx[1:0] and HTx[1:0] to "00" |
| 10: 8-bit | DATA[11:4],0000 | LTx[11:0] and HTx[11:0] | The user must configure LTx[3:0] and HTx[3:0] to "0000" |
| 11: 6-bit | DATA[11:6],000000 | LTx[11:0] and HTx[11:0] | The user must configure LTx[5:0] and HTx[5:0] to "000000" |

1. The watchdog comparison is performed on the raw converted data before any alignment calculation.

Table 72 shows how to configure the AWD1SGL and AWD1EN bits in the ADC_CFGR1 register to enable the analog watchdog on one or more channels.

Figure 49. Analog watchdog guarded area**Table 72. Analog watchdog 1 channel selection**

| Channels guarded by the analog watchdog | AWD1SGL bit | AWD1EN bit |
|---|-------------|------------|
| None | x | 0 |
| All channels | 0 | 1 |
| Single ⁽¹⁾ channel | 1 | 1 |

1. Selected by the AWD1CH[4:0] bits

14.8.2 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 AWDxCHy in ADC_AWDxCR ($x = 2, 3$).

The corresponding watchdog is enabled when any AWDxCHy bit ($x = 2, 3$) is set in ADC_AWDxCR register.

When converting data with a resolution of less than 12 bits (configured through DRES[1:0] bits), 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 71 describes how the comparison is performed for all the possible resolutions.

The AWD2/3 analog watchdog status bit is set if the analog voltage converted by the ADC is below a low threshold or above a high threshold. These thresholds are programmed in

HTx[11:0] and LTx[11:0] of ADC_AWDxTR registers ($x = 2$ or 3). An interrupt can be enabled by setting the AWDxIE bit in the ADC_IER register.

The AWD2 and ADW3 flags are cleared by software by programming them to 1.

14.8.3 ADC_AWDx_OUT output signal generation

Each analog watchdog is associated to an internal hardware signal, ADC_AWDx_OUT (x being the watchdog number) that is directly connected to the ETR input (external trigger) of some on-chip timers (refer to the timers section for details on how to select the ADC_AWDx_OUT signal as ETR).

ADC_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_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.
- ADC_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS to 1). Note that stopping conversions (ADSTP set), might clear the ADC_AWDx_OUT state.
- ADC_AWDx_OUT state does not change when the ADC converts the none-guarded channel (see [Figure 52](#))

AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADC_AWDx_OUT (as an example, ADC_AWDx_OUT can toggle while AWDx flag remains at 1 if the software has not cleared the flag).

The ADC_AWDx_OUT signal is generated by the ADC_CLK domain. This signal can be generated even the APB clock is stopped.

The AWD comparison is performed at the end of each ADC conversion. The ADC_AWDx_OUT rising edge and falling edge occurs two ADC_CLK clock cycles after the comparison.

As ADC_AWDx_OUT is generated by the ADC_CLK domain and AWD flag is generated by the APB clock domain, the rising edges of these signals are not synchronized.

Figure 50. ADC_AWDx_OUT signal generation

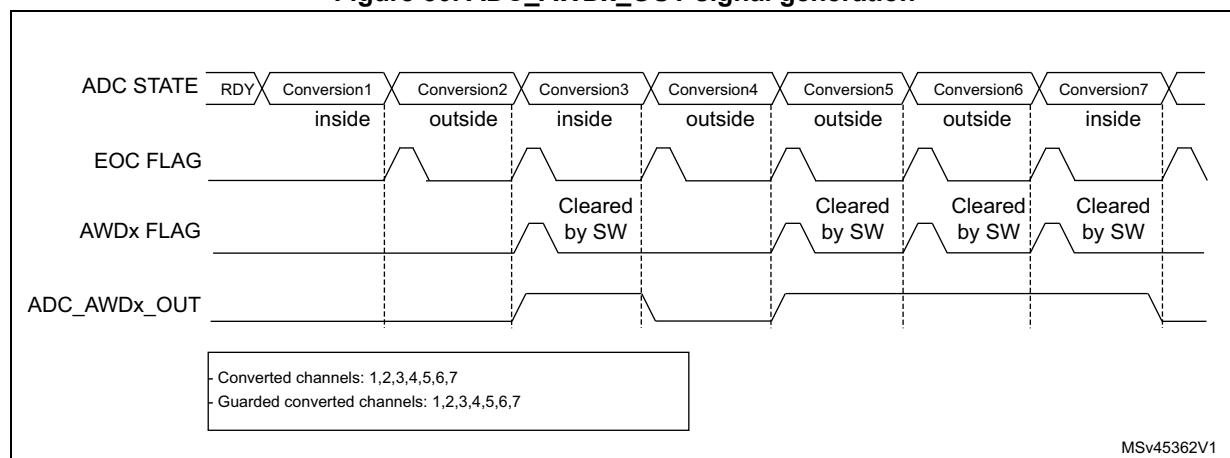
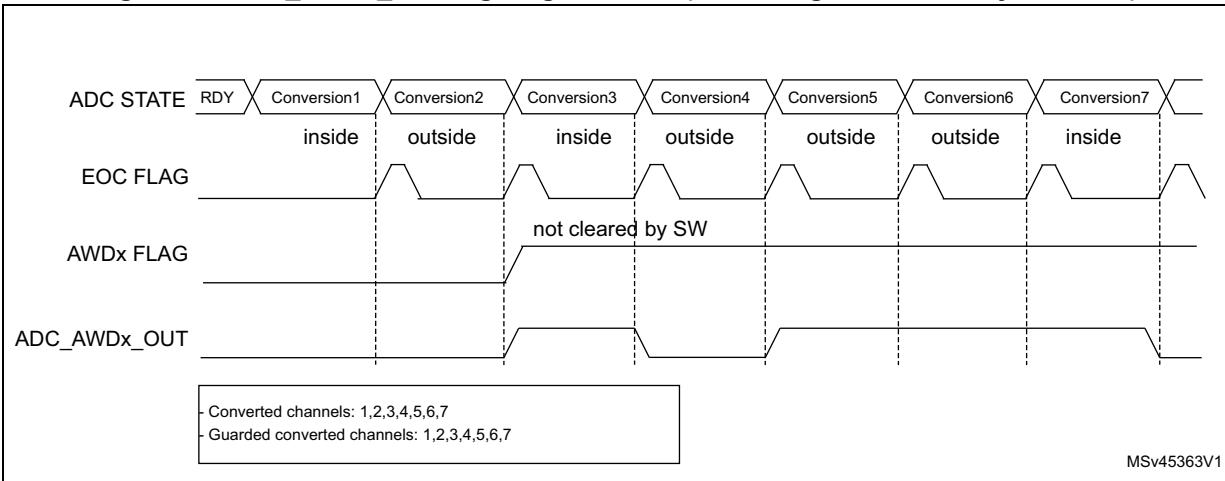
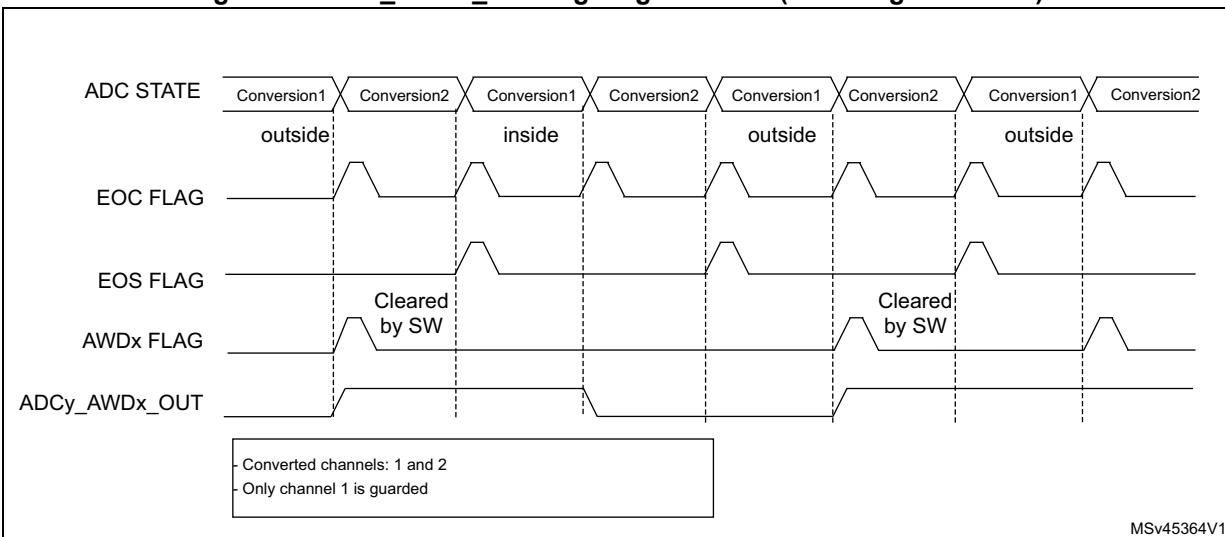
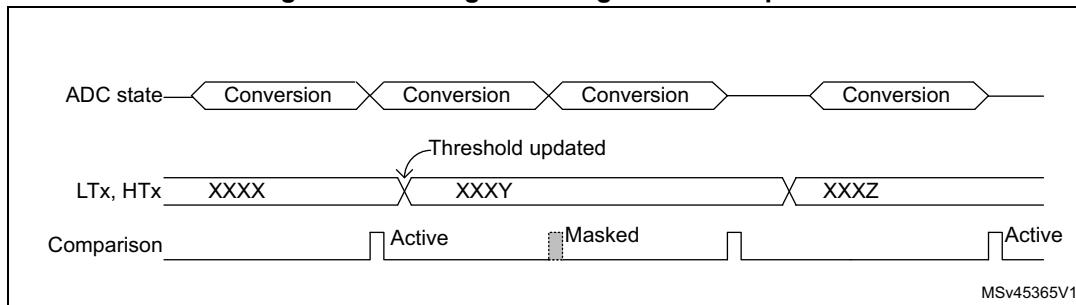


Figure 51. ADC_AWDx_OUT signal generation (AWDx flag not cleared by software)**Figure 52. ADC_AWDx_OUT signal generation (on a single channel)**

14.8.4 Analog Watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed during an analog-to-digital conversion (that is between the start of the conversion and the end of conversion of the ADC internal state). If HTx and LTx bits are programmed during the ADC guarded channel conversion, the watchdog function is masked for this conversion. This mask is cleared when starting a new conversion, and the resulting new AWD threshold is applied starting the next ADC conversion result. AWD comparison is performed at each end of conversion. If the current ADC data are out of the new threshold interval, this does not generate any interrupt or an ADC_AWDx_OUT signal. The Interrupt and the ADC_AWDx_OUT generation only occurs at the end of the ADC conversion that started after the threshold update. If ADC_AWDx_OUT is already asserted, programming the new threshold does not deassert the ADC_AWDx_OUT signal.

Figure 53. Analog watchdog threshold update



14.9 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It can 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:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{N-1} \text{Conversion}(t_n)$$

$n = N - 1$

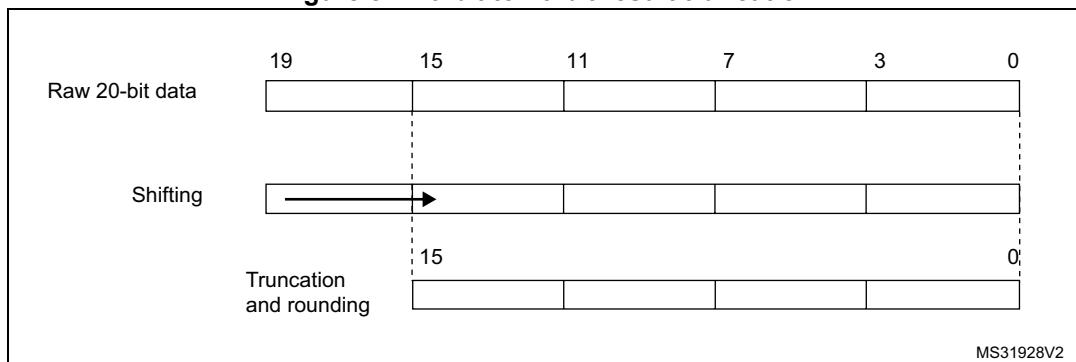
It allows the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC_CFGR2 register. It can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits. It is configured through the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256 x 12-bit), which is first shifted right. The upper bits of the result are then truncated, keeping only 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 intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.*

Figure 54. 20-bit to 16-bit result truncation



The [Figure 55](#) gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 55. Numerical example with 5-bits shift and rounding

| Raw 20-bit data: | 19 | 15 | 11 | 7 | 3 |
|--|----|----|----|---|---|
| | 3 | B | 7 | D | 7 |
| Final result after 5-bits shift and rounding to nearest | 15 | | | | 0 |
| | 1 | D | B | F | |

MS31929V1

[Table 73](#) gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFFF.

Table 73. Maximum output results vs N and M. Grayed values indicates truncation

| Oversampling 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 | 0x0020 |
| 4x | 0x3FFC | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 | 0x0080 | 0x0040 |
| 8x | 0x7FF8 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 | 0x0080 |
| 16x | 0xFFFF0 | 0xFFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 |
| 32x | 0x1FFE0 | 0xFFE0 | 0xFFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 |
| 64x | 0x3FFC0 | 0xFFC0 | 0xFFE0 | 0xFFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 |
| 128x | 0x7FF80 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 |
| 256x | 0xFFFF00 | 0xFF00 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF |

The conversion timings in oversampled mode do not change compared to standard conversion mode: the sample time is maintained equal during the whole oversampling sequence. New data are provided every N conversion, with an equivalent delay equal to $N \times t_{CONV} = N \times (t_{SMPL} + t_{SAR})$. The flags features are raised as following:

- 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 (EOCSEQ) occurs once the sequence of oversampled data is completed (i.e. after $N \times$ sequence length conversions total)

14.9.1 ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are available:

- Single or continuous mode conversions, forward or backward scanned sequences
- 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 (WAIT, AUTOFF)
- 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.*

14.9.2 Analog watchdog

The analog watchdog functionality is available, with the following differences:

- the RES[1:0] bits are ignored, comparison is always done on using the full 12-bits values HTx[11:0] and LTx[11:0]
- the comparison is performed on the most significant 12 bits of the 16 bits oversampled results ADC_DR[15:4]

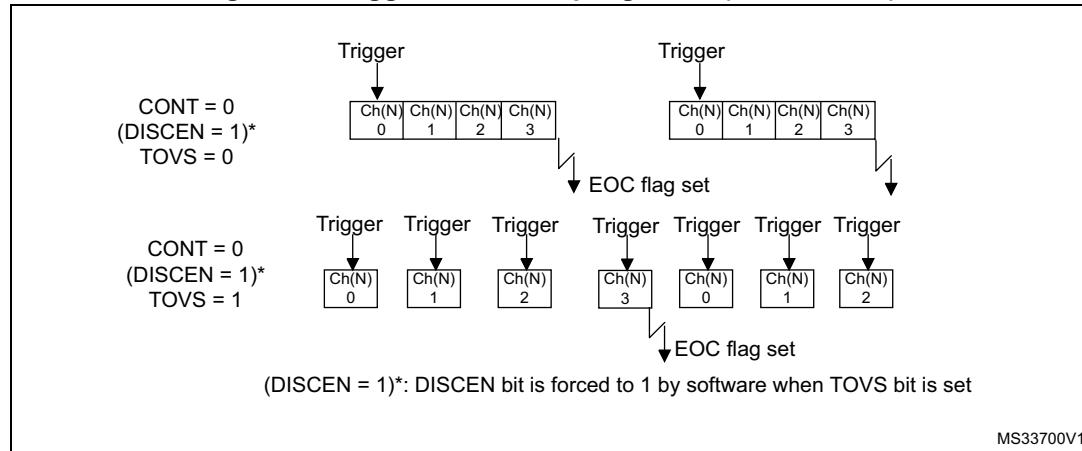
Note: *Care must be taken when using high shifting values. This reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits thus yielding a 12-bit data right-aligned, the affective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HTx[7:0] / LTx[7:0], and HTx[11:8] / LTx[11:8] must be kept reset.*

14.9.3 Triggered mode

The averager can also be used for basic filtering purposes. Although not a very efficient 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 TOVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

Figure 56 below shows how conversions are started in response to triggers in discontinuous mode.

If the TOVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 56. Triggered oversampling mode (TOVS bit = 1)

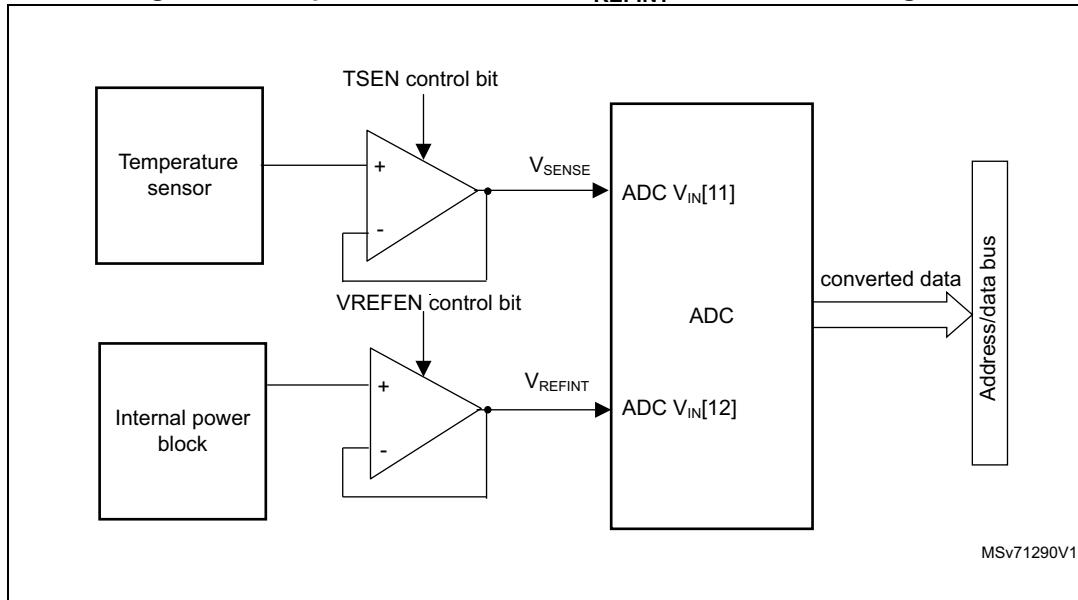
14.10 Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the junction temperature (T_J) of the device. The temperature sensor is internally connected to the ADC $V_{IN}[11]$ input channel which is used to convert the sensor's output voltage to a digital value. The sampling time for the temperature sensor analog pin must be greater than the minimum T_{S_temp} value specified in the datasheet. When not in use, the sensor can be put in power down mode.

The internal voltage reference (VREFINT) provides a stable (bandgap) voltage output for the ADC. VREFINT is internally connected to the ADC $V_{IN}[12]$ input channel. The precise voltage of VREFINT is individually measured for each part by ST during production test and stored in the system memory area.

Figure 57 shows the block diagram of connections between the temperature sensor, the internal voltage reference and the ADC.

The TSEN bit must be set to enable the conversion of ADC $V_{IN}[11]$ (temperature sensor) and the VREFEN bit must be set to enable the conversion of ADC $V_{IN}[12]$ (VREFINT).

Figure 57. Temperature sensor and V_{REFINT} channel block diagram

Reading the temperature

1. Select the ADC V_{IN}[11] input channel.
2. Select an appropriate sampling time specified in the device datasheet (T_{S_temp}).
3. Set the TSEN bit in the ADC_CCR register to wake up the temperature sensor from power down mode and wait for its stabilization time (t_{START}).
4. Start the ADC conversion by setting the ADSTART bit in the ADC_CR register (or by external trigger).
5. Read the resulting V_{SENSE} data in the ADC_DR register.
6. Calculate the temperature using the following formula

$$\text{Temperature (in } ^\circ\text{C)} = \frac{\text{TS_CAL2_TEMP} - \text{TS_CAL1_TEMP}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + \text{TS_CAL1_TEMP}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP (refer to the datasheet for TS_CAL2 value)
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP (refer to the datasheet for TS_CAL1 value)
- TS_DATA is the actual temperature sensor output value converted by ADC
Refer to the specific 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_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and TSEN bits should be set at the same time.

Calculating the actual V_{REF+} voltage using the internal reference voltage

V_{REF+} voltage may be subject to variation or not precisely known. The embedded internal reference voltage (V_{REFINT}) and its calibration data acquired by the ADC during the manufacturing process at V_{REF+_charac} can be used to evaluate the actual V_{REF+} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+_Charac} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value
- VREFINT_DATA is the actual VREFINT 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 the analog power supply and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent of V_{REF+}. For applications where V_{REF+} is known and ADC converted values are right-aligned you can use the following formula to get this absolute value:

$$V_{CHANNELx} = \frac{V_{REF+}}{\text{NUM_CODES}} \times \text{ADC_DATA}_x$$

For applications where V_{REF+} value is not known, you must use the internal voltage reference and V_{REF+} can be replaced by the expression provided in [Section : Calculating the actual V_{REF+} voltage using the internal reference voltage](#), resulting in the following formula:

$$V_{CHANNELx} = \frac{V_{REF+_Charac} \times VREFINT_CAL \times \text{ADC_DATA}_x}{VREFINT_DATA \times \text{NUM_CODES}}$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value
- ADC_DATA_x is the value measured by the ADC on channelx (right-aligned)
- VREFINT_DATA is the actual VREFINT output value converted by the ADC
- NUM_CODES is the number of ADC output codes. For example with 12-bit resolution, it is 2¹² = 4096 or with 8-bit resolution, 2⁸ = 256.

Note: If ADC measurements are done using an output format other than 12 bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

14.11 Battery voltage monitoring

The VBATEN bit in the ADC_CCR register allows the application to measure the backup battery voltage on the VBAT pin. As the V_{BAT} voltage can be higher than V_{REF+}, to ensure

the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider. This bridge is automatically enabled when VBATEN is set, to connect V_{BAT} to the ADC V_{IN}[13] input channel. As a consequence, the converted digital value is V_{BAT}/3. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed for ADC conversion.

14.12 ADC interrupts

An interrupt can be generated by any of the following events:

- End Of Calibration (EOCAL flag)
- ADC power-up, when the ADC is ready (ADRDY flag)
- End of any conversion (EOC flag)
- End of a sequence of conversions (EOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2, AWD3 flags)
- When the Channel configuration is ready (CCRDY flag)
- When the end of sampling phase occurs (EOSMP flag)
- when a data overrun occurs (OVR flag)

Separate interrupt enable bits are available for flexibility.

Table 74. ADC interrupts

| Interrupt event | Event flag | Enable control bit |
|-------------------------------------|------------|--------------------|
| End Of Calibration | EOCAL | EOCALIE |
| ADC ready | ADRDY | ADRDYIE |
| End of conversion | EOC | EOCIE |
| End of sequence of conversions | EOS | EOSIE |
| 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 |
| Channel Configuration Ready | CCRDY | CCRDYIE |
| End of sampling phase | EOSMP | EOSMPIE |
| Overrun | OVR | OVRIE |

14.13 ADC registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

14.13.1 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. | CCRDY | Res. | EOCAL | Res. | AWD3 | AWD2 | AWD1 | Res. | Res. | 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 |

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

Bit 13 **CCRDY**: Channel Configuration Ready flag

This flag bit is set by hardware when the channel configuration is applied after programming to ADC_CHSEL register or changing CHSELRMOD or SCANDIR. It is cleared by software by programming it to it.

0: Channel configuration update not applied.

1: Channel configuration update is applied.

Note: When the software configures the channels (by programming ADC_CHSEL or changing CHSELRMOD or SCANDIR), it must wait until the CCRDY flag rises before configuring again or starting conversions, otherwise the new configuration (or the START bit) is ignored. Once the flag is asserted, if the software needs to configure again the channels, it must clear the CCRDY flag before proceeding with a new configuration.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **EOCAL**: End Of Calibration flag

This bit is set by hardware when calibration is complete. It is cleared by software writing 1 to it.

0: Calibration is not complete

1: Calibration is complete

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD3TR and ADC_AWD3TR registers. It is cleared by software by programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD2TR and ADC_AWD2TR registers. It is cleared by software programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog event occurred

Bit 7 AWD1: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC_TR1 and ADC_HR1 registers. It is cleared by software by programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog event occurred

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 OVR: ADC overrun

This bit is set by hardware when an overrun occurs, meaning that a new conversion has complete 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 sequence flag

This bit is set by hardware at the end of the conversion of a sequence of channels selected by the CHSEL bits. It is cleared by software writing 1 to it.

0: Conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Conversion sequence complete

Bit 2 EOC: End of conversion flag

This bit is set by hardware at the end of each conversion of a channel when a new data result is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.

0: Channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Channel conversion complete

Bit 1 EOSMP: End of sampling flag

This bit is set by hardware during the conversion, at the end of the sampling phase. It is cleared by software by programming it to '1'.

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 (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

14.13.2 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. | CCRD YIE | Res. | EOCAL IE | Res. | AWD3I E | AWD2I E | AWD1I E | Res. | Res. | OVRIE | EOSIE | EOCIE | EOSMP IE | ADRDY IE |
| | | rw | | rw | | rw | rw | rw | | | rw | rw | rw | rw | rw |

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

Bit 13 **CCRDYIE**: Channel Configuration Ready Interrupt enable

This bit is set and cleared by software to enable/disable the channel configuration ready interrupt.

0: Channel configuration ready interrupt disabled

1: Channel configuration ready interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.

Bit 11 **EOCALIE**: End of calibration interrupt enable

This bit is set and cleared by software to enable/disable the end of calibration interrupt.

0: End of calibration interrupt disabled

1: End of calibration interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this 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 interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this 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 interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 OVRIE: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the overrun interrupt.

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 bit is cleared (this ensures that no conversion is ongoing).

Bit 3 EOSIE: End of conversion sequence interrupt enable

This bit is set and cleared by software to enable/disable the end of 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 bit is cleared (this ensures that no conversion is ongoing).

Bit 2 EOCIE: End of conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of 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 bit is cleared (this ensures that no conversion is ongoing).

Bit 1 EOSMPIE: End of sampling flag interrupt enable

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt.

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 bit is cleared (this ensures that no 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 bit is cleared (this ensures that no conversion is ongoing).

14.13.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|------|------|----------|------|------|------|------|------|------|------|-------|------|---------|-------|------|
| ADCAL | Res. | Res. | ADVREGEN | Res. | Res. | Res. | Res. | Res. |
| rs | | | 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. | ADSTP | Res. | ADSTART | ADDIS | ADEN |
| | | | | | | | | | | | rs | | rs | rs | rs |

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC.

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 is in progress.

Note: The software is allowed to set ADCAL only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0, AUTOFF = 0, and ADEN = 0).

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 (ADC enabled and no conversion is ongoing).

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **ADVREGEN**: ADC Voltage Regulator Enable

This bit is set by software, to enable the ADC internal voltage regulator. The voltage regulator output is available after $t_{ADCVREG_STUP}$.

It is cleared by software to disable the voltage regulator. It can be cleared only if ADEN is set to 0.

0: ADC voltage regulator disabled

1: ADC voltage regulator enabled

Note: The software is allowed to program this bit field only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:5 Reserved, must be kept at reset value.

Bit 4 **ADSTP**: ADC stop conversion command

This bit is set by software to stop and discard an ongoing conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC is ready to accept a new start conversion command.

0: No ADC stop conversion command ongoing

1: Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress.

Note: Setting ADSTP to '1' is only effective when ADSTART = 1 and ADDIS = 0 (ADC is enabled and may be converting and there is no pending request to disable the ADC)

Bit 3 Reserved, must be kept at reset value.

Bit 2 ADSTART: ADC start conversion command

This bit is set by software to start ADC conversion. Depending on the EXLEN [1:0] configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- In single conversion mode (CONT = 0, DISCEN = 0), when software trigger is selected (EXLEN = 00): at the assertion of the end of Conversion Sequence (EOS) flag.
- In discontinuous conversion mode (CONT = 0, DISCEN = 1), when the software trigger is selected (EXLEN = 00): at the assertion of the end of Conversion (EOC) flag.
- In all other cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware.

0: No ADC conversion is ongoing.

1: Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting.

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).

After writing to ADC_CHSELR register or changing CHSELRMOD or SCANDIRW, it is mandatory to wait until CCRDY flag is asserted before setting ADSTART, otherwise, the value written to ADSTART is ignored.

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: Setting ADDIS to ‘1’ is only effective when ADEN = 1 and ADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 ADEN: ADC enable command

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag 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, ADSTP = 0, ADSTART = 0, ADDIS = 0 and ADEN = 0)

14.13.4 ADC configuration register 1 (ADC_CFGR1)

Address offset: 0x0C

Reset value: 0x0000 0000

The software is allowed to program ADC_CFGR1 only when ADEN is cleared in ADC_CR.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------------|------|------|------------|------------|------|----|----|----|-------------|-------------|---------------|-------------|------------|-----------|------|------------|
| Res. | | | | | | | | | AWD1E N | AWD1S GL | CHSEL RMOD | Res. | Res. | Res. | Res. | DISCE N |
| | 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 |
| AUTOF F | WAIT | CONT | OVRM OD | EXTEN[1:0] | Res. | | | | EXTSEL[2:0] | ALIGN | RES[1:0] | SCAND IR | DMAC FG | DMAE N | | |
| | 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 **AWD1CH[4:0]**: Analog watchdog 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 AWD

00001: ADC analog input Channel 1 monitored by AWD

.....

10011: ADC analog input Channel 19 monitored by AWD

Others: Reserved

Note: The channel selected by the AWDCH[4:0] bits must be also set into the CHSELR register.

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

Bit 23 **AWD1EN**: Analog watchdog enable

This bit is set and cleared by software.

0: Analog watchdog 1 disabled

1: Analog watchdog 1 enabled

Bit 22 **AWD1SGL**: Enable the watchdog 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 AWDCH[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

Bit 21 **CHSELRMOD**: Mode selection of the ADC_CHSELR register

This bit is set and cleared by software to control the ADC_CHSELR feature:

0: Each bit of the ADC_CHSELR register enables an input

1: ADC_CHSELR register is able to sequence up to 8 channels

Note: If CCRDY is not yet asserted after channel configuration (writing ADC_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit is ignored.

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 DISCEN: Discontinuous mode

This bit is set and cleared by software to enable/disable discontinuous mode.

0: Discontinuous mode disabled

1: Discontinuous mode enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

Bit 15 AUTOFF: Auto-off mode

This bit is set and cleared by software to enable/disable auto-off mode.

0: Auto-off mode disabled

1: Auto-off mode enabled

Bit 14 WAIT: Wait conversion mode

This bit is set and cleared by software to enable/disable wait conversion mode.

0: Wait conversion mode off

1: Wait conversion mode on

Bit 13 CONT: Single / continuous conversion mode

This bit is set and cleared by software. If it is set, 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 bits DISCEN = 1 and CONT = 1.

Bit 12 OVRMOD: Overrun management mode

This bit is set and cleared by software and configure the way data overruns are 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.

Bits 11:10 EXTN[1:0]: External trigger enable and polarity selection

These bits are set and cleared by software to select the external trigger polarity and enable the trigger.

00: Hardware trigger detection disabled (conversions can be started 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

Bit 9 Reserved, must be kept at reset value.

Bits 8:6 **EXTSEL[2:0]**: External trigger selection

These bits select the external event used to trigger the start of conversion (refer to [Table 67: External triggers](#) for details):

- 000: TRG0
- 001: TRG1
- 010: TRG2
- 011: TRG3
- 100: TRG4
- 101: TRG5
- 110: TRG6
- 111: TRG7

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to [Figure 44: Data alignment and resolution \(oversampling disabled: OVSE = 0\) on page 347](#)

- 0: Right alignment
- 1: Left alignment

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

- 00: 12 bits
- 01: 10 bits
- 10: 8 bits
- 11: 6 bits

Bit 2 **SCANDIR**: Scan sequence direction

This bit is set and cleared by software to select the direction in which the channels are scanned in the sequence. It is effective only if CHSELMOD bit is cleared.

- 0: Upward scan (from CHSEL0 to CHSEL)
- 1: Backward scan (from CHSEL to CHSEL0)

Note: If CCRDY is not yet asserted after channel configuration (writing ADC_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit is ignored.

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 14.6.5: Managing converted data using the DMA on page 348](#).

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows the DMA controller to be used to manage automatically the converted data. For more details, refer to [Section 14.6.5: Managing converted data using the DMA on page 348](#).

- 0: DMA disabled
- 1: DMA enabled

14.13.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

The software is allowed to program ADC_CFGR2 only when ADEN is cleared in ADC_CR.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------------|--------|------|------|------|------|------|-----------|------|------|------|-----------|------|------|------|------|
| CKMODE[1:0] | LFTRIG | Res. | 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 |
| Res. | Res. | Res. | Res. | Res. | Res. | TOVS | OVSS[3:0] | | | | OVSR[2:0] | | | Res. | OVSE |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | | rw |

Bits 31:30 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define how the analog ADC is clocked:

00: ADCCLK (Asynchronous clock mode), generated at product level (refer to RCC section)

01: PCLK/2 (Synchronous clock mode)

10: PCLK/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 ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 **LFTRIG:** Low frequency trigger mode enable

This bit is set and cleared by software.

0: Low Frequency Trigger Mode disabled

1: Low Frequency Trigger Mode enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bits 28:10 Reserved, must be kept at reset value.

Bit 9 **TOVS:** Triggered Oversampling

This bit is set and cleared by software.

0: All oversampled conversions for a channel are done consecutively after a trigger

1: Each oversampled conversion for a channel needs a trigger

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bits 8:5 **OVSS[3:0]:** Oversampling shift

This bit is set and cleared by software.

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

Others: Reserved

Note: The software is allowed to write this bit only when ADEN bit is cleared.

Bits 4:2 **OVS[2:0]**: Oversampling ratio

This bit field defines the number of 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 this bit only when ADEN bit is cleared.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **OVSE**: Oversampler Enable

This bit is set and cleared by software.

0: Oversampler disabled

1: Oversampler enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared.

14.13.6 ADC sampling time register (ADC_SMPR)

Address offset: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------------|-------------|-------------|-------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|-------------|-------------|
| Res. | Res. | Res. | Res. | SMPSE L19 | SMPSE L18 | SMPSE L17 | SMPSE L16 | SMPSE L15 | SMPSE L14 | SMPSE L13 | SMPSE L12 | SMPSE L11 | SMPSE L10 | SMPSE L9 | SMPSE L8 |
| | | | | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SMPSE L7 | SMPSE L6 | SMPSE L5 | SMPSE L4 | SMPSE L3 | SMPSE L2 | SMPSE L1 | SMPSE L0 | Res. | SMP2[2:0] | | | Res. | SMP1[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | | rw | rw | rw |

Bits 27:8 **SMPSELx**: Channel-x sampling time selection (x = 19 to 0)

These bits are written by software to define which sampling time is used.

0: Sampling time of CHANNELx use the setting of SMP1[2:0] register.

1: Sampling time of CHANNELx use the setting of SMP2[2:0] register.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SMP2[2:0]: Sampling time selection 2**

These bits are written by software to select the sampling time that applies to all channels.

- 000: 1.5 ADC clock cycles
- 001: 3.5 ADC clock cycles
- 010: 7.5 ADC clock cycles
- 011: 12.5 ADC clock cycles
- 100: 19.5 ADC clock cycles
- 101: 39.5 ADC clock cycles
- 110: 79.5 ADC clock cycles
- 111: 160.5 ADC clock cycles

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMP1[2:0]: Sampling time selection 1**

These bits are written by software to select the sampling time that applies to all channels.

- 000: 1.5 ADC clock cycles
- 001: 3.5 ADC clock cycles
- 010: 7.5 ADC clock cycles
- 011: 12.5 ADC clock cycles
- 100: 19.5 ADC clock cycles
- 101: 39.5 ADC clock cycles
- 110: 79.5 ADC clock cycles
- 111: 160.5 ADC clock cycles

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

14.13.7 ADC watchdog threshold register (ADC_AWD1TR)

Address offset: 0x20

Reset value: 0xFFFF 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.

Refer to [Section 14.8: Analog window watchdogs on page 352](#).

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.

Refer to [Section 14.8: Analog window watchdogs on page 352](#).

14.13.8 ADC watchdog threshold register (ADC_AWD2TR)

Address offset: 0x24

Reset value: 0xFFFF 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
|------|------|------|------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|--|
| Res. | Res. | Res. | Res. | HT2[11:0] | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| Res. | Res. | Res. | Res. | LT2[11:0] | | | | | | | | | | | | | |
| | | | | rw | 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 **HT2[11:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 352](#).

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

Bits 11:0 **LT2[11:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 352](#).

14.13.9 ADC channel selection register (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

- Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFGR1). Refer to the current section.
- ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFGR1). Refer to next section.

CHSELRMOD = 0 in ADC_CFGR1

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

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

Bits 19:0 CHSEL[19:0]: Channel-x selection

These bits are written by software and define which channels are part of the sequence of channels to be converted.

0: Input Channel-x is not selected for conversion

1: Input Channel-x is selected for conversion

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

If CCRDY is not yet asserted after channel configuration (writing ADC_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit is ignored.

14.13.10 ADC channel selection register [alternate] (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

- Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFGR1). Refer to the current previous section.
- ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFGR1). Refer to this section.

CHSELRMOD = 1 in ADC_CFGR1:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------|----|----|----|----------|----|----|----|----------|----|----|----|----------|----|----|----|
| SQ8[3:0] | | | | SQ7[3:0] | | | | SQ6[3:0] | | | | SQ5[3:0] | | | |
| rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SQ4[3:0] | | | | SQ3[3:0] | | | | SQ2[3:0] | | | | SQ1[3:0] | | | |
| rw | rw | rw | rw |

Bits 31:28 **SQ8[3:0]:** 8th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates the end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

0000: CH0

0001: CH1

...

1100: CH12

1101: CH13

1110: CH14

1111: No channel selected (End of sequence)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:24 **SQ7[3:0]:** 7th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:20 **SQ6[3:0]:** 6th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 19:16 SQ5[3:0]: 5th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:12 SQ4[3:0]: 4th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 11:8 SQ3[3:0]: 3rd conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 7:4 SQ2[3:0]: 2nd conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 3:0 SQ1[3:0]: 1st conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

14.13.11 ADC watchdog threshold register (ADC_AWD3TR)

Address offset: 0x2C

Reset value: 0x0FFF 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
|------|------|------|------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|--|
| Res. | Res. | Res. | Res. | HT3[11:0] | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| Res. | Res. | Res. | Res. | LT3[11:0] | | | | | | | | | | | | | |
| | | | | rw | 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 **HT3[11:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 352](#).

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

Bits 11:0 **LT3[11:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 352](#).

14.13.12 ADC 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 |
| DATA[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 **DATA[15:0]**: Converted data

These bits are read-only. They contain the conversion result from the last converted channel. The data are left- or right-aligned as shown in [Figure 44: Data alignment and resolution \(oversampling disabled: OVSE = 0\) on page 347](#).

Just after a calibration is complete, DATA[6:0] contains the calibration factor.

14.13.13 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. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AWD2 CH15 | AWD2 CH14 | AWD2 CH13 | AWD2 CH12 | AWD2 CH11 | AWD2 CH10 | AWD2 CH9 | AWD2 CH8 | AWD2 CH7 | AWD2 CH6 | AWD2 CH5 | AWD2 CH4 | AWD2 CH3 | AWD2 CH2 | AWD2 CH1 | AWD2 CH0 |
| 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 **AWD2CH[19:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 2 (AWD2).

0: ADC analog channel-x is not monitored by AWD2

1: ADC analog channel-x is monitored by AWD2

Note: The channels selected through ADC_AWD2CR must be also configured into the ADC_CHSELR registers. The software is allowed to write this bit only when ADEN = 0.

14.13.14 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. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AWD3 CH15 | AWD3 CH14 | AWD3 CH13 | AWD3 CH12 | AWD3 CH11 | AWD3 CH10 | AWD3 CH9 | AWD3 CH8 | AWD3 CH7 | AWD3 CH6 | AWD3 CH5 | AWD3 CH4 | AWD3 CH3 | AWD3 CH2 | AWD3 CH1 | AWD3 CH0 |
| 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 **AWD3CH[19:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 3 (AWD3).

0: ADC analog channel-x is not monitored by AWD3

1: ADC analog channel-x is monitored by AWD3

Note: The channels selected through ADC_AWD3CR must be also configured into the ADC_CHSELR registers. The software is allowed to write this bit only when ADEN = 0.

14.13.15 ADC calibration factor (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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 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. | rw | rw | rw | rw | rw | rw |
| | | | | | | | | | | | | | | | |
| CALFACT[6:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

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

Bits 6:0 **CALFACT[6:0]**: Calibration factor

These bits are written by hardware or by software.

- Once a 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 conversion is launched.
- Just after a calibration is complete, DATA[6:0] contains the calibration factor.

Note: Software can write these bits only when ADEN=1 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

14.13.16 ADC common configuration 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. | VBAT EN | TSEN | VREF EN | PRESC[3:0] | | | | Res. | Res. |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |

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

Bit 24 **VBATEN**: V_{BAT} enable

This bit is set and cleared by software to enable/disable the V_{BAT} channel.

- 0: V_{BAT} channel disabled
- 1: V_{BAT} channel enabled

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing)

Bit 23 **TSEN**: Temperature sensor enable

This bit is set and cleared by software to enable/disable the temperature sensor.

- 0: Temperature sensor disabled
- 1: Temperature sensor enabled

Note: Software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT}.

- 0: V_{REFINT} disabled
- 1: V_{REFINT} enabled

Note: Software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 21:18 **PRES[3:0]**: ADC prescaler

Set and cleared by software to select the frequency of the clock to the ADC.

- 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: Software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 17:0 Reserved, must be kept at reset value.

14.14 ADC register map

The following table summarizes the ADC registers.

Table 75. ADC 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 | CCRDY | 12 | 11 | EOCAL | 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 | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 75. 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 | | |
|--------|-------------------------------------|-------------|----------|----------|----------|----------|----------|----------|----------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|
| 0x04 | ADC_IER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 0x08 | ADC_CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x0C | ADC_CFGR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 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. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x14 | ADC_SMPR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 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 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1C | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x20 | ADC_AWD1TR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 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 | |
| 0x24 | ADC_AWD2TR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 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 | |
| 0x28 | ADC_CHSELR (CHSELRLMOD=0) | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x28 | ADC_CHSELR (CHSELRLMOD=1) | SQ8[3:0] | SQ7[3:0] | SQ6[3:0] | SQ5[3:0] | SQ4[3:0] | SQ3[3:0] | SQ2[3:0] | SQ1[3:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | |
| 0x2C | ADC_AWD3TR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 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 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x38 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3C | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x40 | ADC_DR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA0 | ADC_AWD2CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 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 75. ADC 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 | | | |
|--------|--------------------|-------------|----------|------|------|------|------|------|------|------|------|--------|-------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|------|---|---|
| 0xA4 | ADC_AWD3CR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | AWD3CH19 | AWD3CH18 | AWD3CH17 | AWD3CH16 | AWD3CH15 | AWD3CH14 | AWD3CH13 | AWD3CH12 | AWD3CH11 | AWD3CH10 | AWD3CH9 | AWD3CH8 | AWD3CH7 | AWD3CH6 | AWD3CH5 | AWD3CH4 | AWD3CH3 | AWD3CH2 | AWD3CH1 | AWD3CH0 | 0 | | |
| | | Reset value | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | | | |
| | | .. | Reserved | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | | | |
| 0xB4 | ADC_CALFACT | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | | |
| | | .. | Reserved | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | | |
| 0x308 | ADC_CCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | TSEN | VREFEN | PRES3 | PRES2 | PRES1 | PRES0 | PRESCO | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| | | .. | Reserved | — | — | — | — | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

Refer to [Section 2.2](#) for the register boundary addresses.

15 Digital-to-analog converter (DAC)

15.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features one single channel. An input reference pin, VREF+ (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to *voltage reference buffer (VREFBUF)* section.

The DACx_OUT1 pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the Sample and hold mode.

15.2 DAC main features

The DAC main features are the following (see [Figure 58: DAC block diagram](#))

- One DAC interface
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Single DAC channel
- DMA capability including DMA underrun error detection
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- The DAC output can be disconnected from the DACx_OUT1 output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from VREF+ pin or internal VREFBUF reference

[Figure 58](#) shows the block diagram of a DAC channel and [Table 77](#) gives the pin description.

15.3 DAC implementation

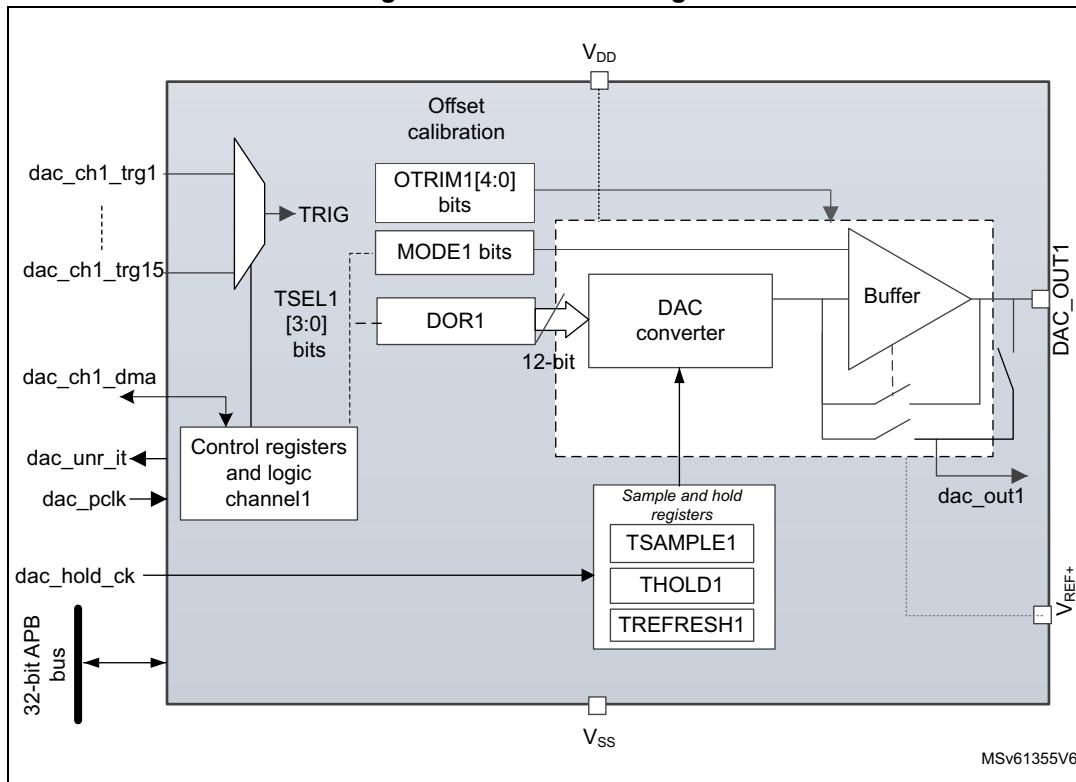
Table 76. DAC features

| DAC features | DAC |
|-----------------------|-----------------|
| Dual channel | - |
| Output buffer | X |
| I/O connection | DAC_OUT1 to PA4 |
| Maximum sampling time | 1 Msps |
| Autonomous mode | - |
| VREF+ pin | X |

15.4 DAC functional description

15.4.1 DAC block diagram

Figure 58. DAC block diagram



1. MODE1 bits in the DAC_MCR control the output mode and allow switching between the Normal mode in buffer/unbuffered configuration and the Sample and hold mode.

15.4.2 DAC pins and internal signals

The DAC includes:

- One output channel
- The DACx_OUT1 can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI clock source (dac_hold_ck) for static conversion.

The DAC includes one output channel. The output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUT1 output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The Sample and hold block and its associated registers can run in Stop mode using the LSI clock source (dac_hold_ck).

Table 77. DAC input/output pins

| Pin name | Signal type | Remarks |
|-----------|----------------------------------|--|
| VREF+ | Input, analog positive reference | The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet) |
| VDD | Input, analog supply | Analog power supply |
| VSS | Input, analog supply ground | Ground for analog power supply |
| DACx_OUT1 | Analog output signal | DACx channel1 analog output |

Table 78. DAC internal input/output signals

| Internal signal name | Signal type | Description |
|----------------------------|---------------|--|
| dac_ch1_dma | Bidirectional | DAC channel1 DMA request/acknowledge |
| dac_ch1_trgx (x = 1 to 14) | Inputs | DAC channel1 trigger inputs/acknowledge |
| dac_unr_it | Output | DAC underrun interrupt |
| dac_pclk | Input | DAC peripheral clock |
| dac_hold_ck | Input | DAC low-power clock used in Sample and hold mode |
| dac_out1 | Analog output | DAC channel1 output for on-chip peripherals |

Table 79. DAC interconnection

| Signal name | Source | Source type |
|--------------|-----------|---|
| dac_hold_ck | ck_lsi | LSI clock |
| dac_ch1_trg1 | tim1_trgo | Internal signal from on-chip timers TIM1_TGO_CKTM |

Table 79. DAC interconnection (continued)

| Signal name | Source | Source type |
|---------------|------------|--|
| dac_ch1_trg2 | tim2_trgo | Internal signal from on-chip timers TIM2_TGO_CKTM |
| dac_ch1_trg3 | tim3_trgo | internal signal from on-chip timers TIM3_TGO_CKTM |
| dac_ch1_trg5 | tim6_trgo | internal signal from on-chip timers TIM6_TGO_CKTM |
| dac_ch1_trg6 | tim7_trgo | internal signal from on-chip timers TIM7_TGO_CKTM |
| dac_ch1_trg8 | tim15_trgo | internal signal from on-chip timers TIM15_TGO_CKTM |
| dac_ch1_trg11 | lptim1_out | Internal signal from on-chip timers LPTIM1_OUT |
| dac_ch1_trg12 | lptim2_out | Internal signal from on-chip timers LPTIM2_OUT |
| dac_ch1_trg14 | exti9 | External pin EXTI[9] |

15.4.3 DAC channel enable

The DAC channel can be powered on by setting its corresponding EN1 bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

Note: *The EN1 bit enables the analog DAC channel1 only. The DAC channel1 digital interface is enabled even if the EN1 bit is reset.*

15.4.4 DAC data format

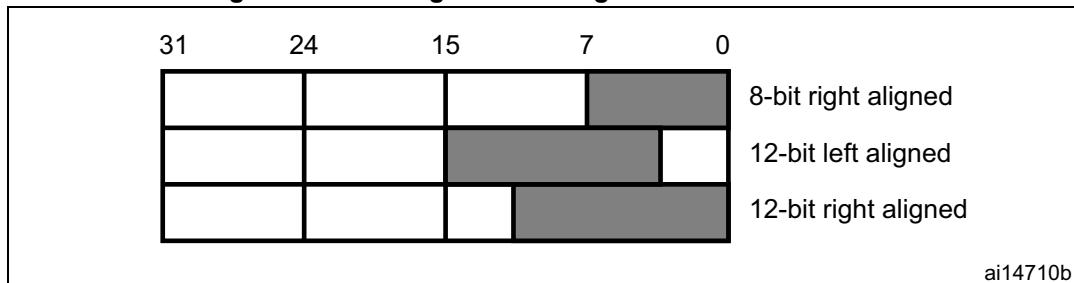
Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

There are three possibilities:

- 8-bit right alignment: the software has to load data into the DAC_DHR8R1[7:0] bits (stored into the DHR1[11:4] bits)
- 12-bit left alignment: the software has to load data into the DAC_DHR12L1 [15:4] bits (stored into the DHR1[11:0] bits)
- 12-bit right alignment: the software has to load data into the DAC_DHR12R1 [11:0] bits (stored into the DHR1[11:0] bits)

Depending on the loaded DAC_DHRyyx register, the data written by the user is shifted and stored into the corresponding DHR1 (data holding registerx, which are internal non-memory-mapped registers). The DHR1 register is then loaded into the DOR1 register either automatically, by software trigger or by an external event trigger.

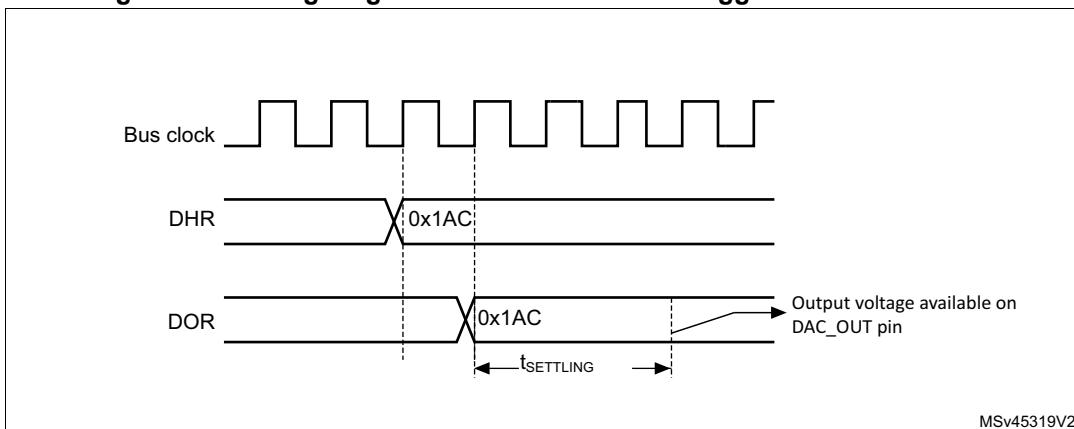
Figure 59. Data registers in single DAC channel mode

15.4.5 DAC conversion

The DAC_DOR1 cannot be written directly and any data transfer to the DAC channel1 must be performed by loading the DAC_DHR1 register (write operation to DAC_DHR8R1, DAC_DHR12L1, DAC_DHR12R1).

Data stored in the DAC_DHR1 register are automatically transferred to the DAC_DOR1 register after one dac_pclk clock cycle, if no hardware trigger is selected (TEN1 bit in DAC_CR register is reset). However, when a hardware trigger is selected (TEN1 bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_pclk clock cycles after the trigger signal.

When DAC_DOR1 is loaded with the DAC_DHR1 contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

Figure 60. Timing diagram for conversion with trigger disabled TEN = 0

15.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+} .

The analog output voltage on the DAC channel pin is determined by the following equation:

$$\text{DACoutput} = V_{REF} \times \frac{\text{DOR}}{4096}$$

where

all voltages are expressed in volt.

15.4.7 DAC trigger selection

If the TEN1 control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSEL1[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSEL1[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in [Section 15.4.2: DAC pins and internal signals](#).

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHR1 register are transferred into the DAC_DOR1 register. The DAC_DOR1 register is updated three dac_pclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DOR1 register has been loaded with the DAC_DHR1 register contents.

Note: *TSEL1[3:0] bit cannot be changed when the EN1 bit is set.*

When software trigger is selected, the transfer from the DAC_DHR1 register to the DAC_DOR1 register takes only one dac_pclk clock cycle.

15.4.8 DMA requests

The DAC channel has a DMA capability. One DMA channel is used to service DAC channel DMA request.

When an external trigger (but not a software trigger) occurs while the DMAEN1 bit is set, the value of the DAC_DHR1 register is transferred into the DAC_DOR1 register when the transfer is complete, and a DMA request is generated.

As DAC_DHR1 to DAC_DOR1 data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHR1 before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channel1 underrun flag DMAUDR1 in the DAC_SR register is set, reporting the error condition. The DAC channel1 continues to convert old data.

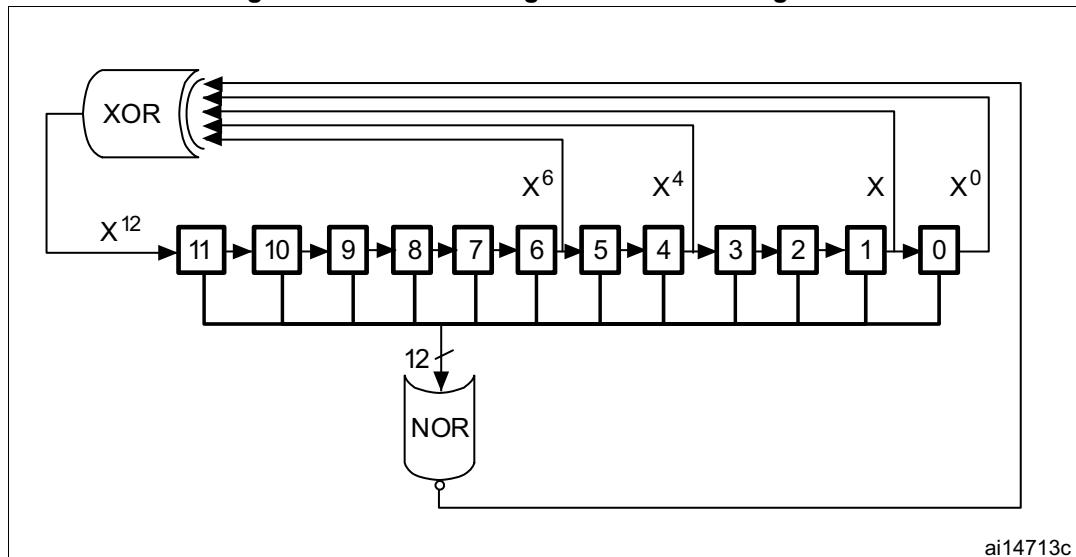
The software must clear the DMAUDR1 flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channel1 to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For DAC channel1, an interrupt is also generated if its corresponding DMAUDRIE1 bit in the DAC_CR register is enabled.

15.4.9 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVE1[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_pclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 61. DAC LFSR register calculation algorithm

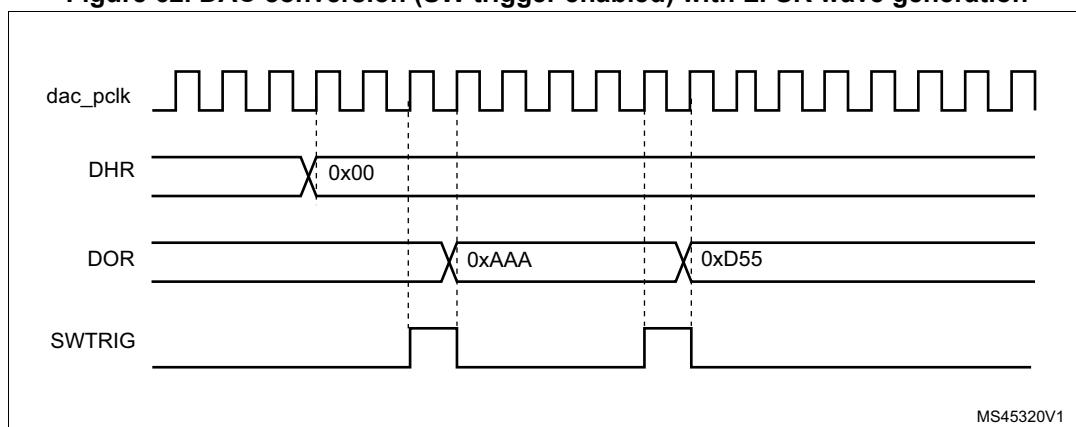


The LFSR value, that may be masked partially or totally by means of the MAMP1[3:0] bits in the DAC_CR register, is added up to the DAC_DHR1 contents without overflow and this value is then transferred into the DAC_DOR1 register.

If LFSR is 0x0000, a '1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVE1[1:0] bits.

Figure 62. DAC conversion (SW trigger enabled) with LFSR wave generation



Note:

The DAC trigger must be enabled for noise generation by setting the TEN1 bit in the DAC_CR register.

15.4.10 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVE1[1:0] to 10. The amplitude is configured through the MAMP1[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_pclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHR1 register without overflow and the sum is transferred into the DAC_DOR1 register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMP1[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVE1[1:0] bits.

Figure 63. DAC triangle wave generation

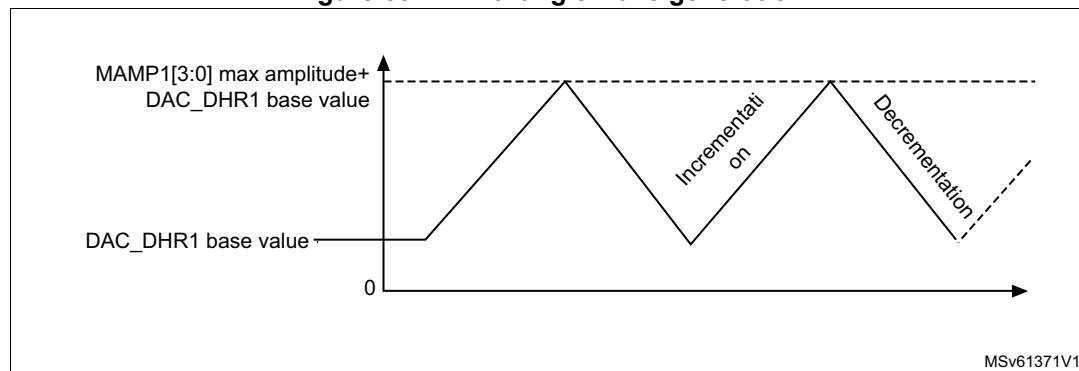
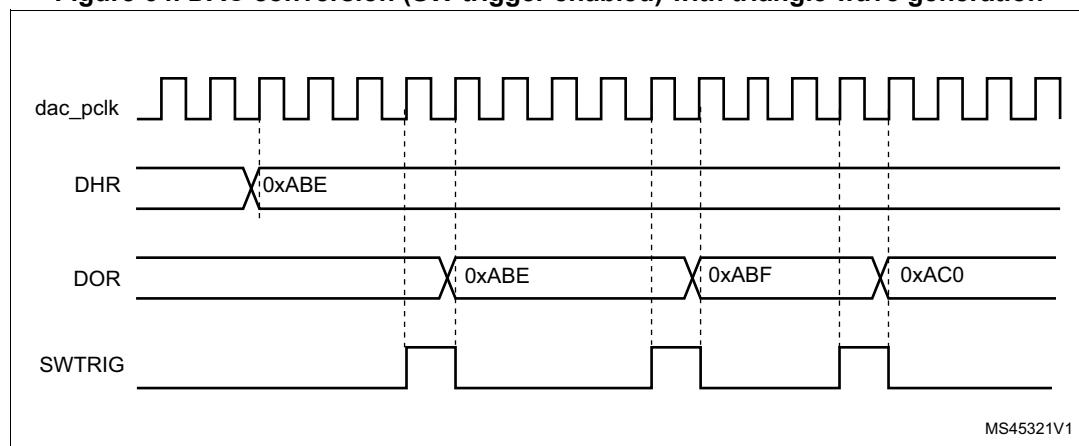


Figure 64. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TEN1 bit in the DAC_CR register.

The MAMP1[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

15.4.11 DAC channel modes

The DAC channel can be configured in Normal mode or Sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In Normal mode, there are four combinations, by changing the buffer state and by changing the DAC_x_OUT1 pin interconnections.

To enable the output buffer, the MODE1[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODE1[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In Sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI low-speed clock (dac_hold_ck) in addition to the dac_pclk clock, allowing using the DAC channel in deep low power modes such as Stop mode.

The LSI low-speed clock (dac_hold_ck) must not be stopped when the Sample and hold mode is enabled.

The sample/hold mode operations can be divided into 3 phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLE1[9:0] bits in DAC_SHSR1 register. During the write of the TSAMPLE1[9:0] bits, the BWST1 bit in DAC_SR register is set to 1 to synchronize between both clocks domains (APB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLD1[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESH1[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI clock periods. As an example, to configure a sample time of 350 μ s, a hold time of 2 ms and a refresh time of 100 μ s assuming LSI ~32 KHz is selected:

12 cycles are required for sample phase: TSAMPLE1[9:0] = 11,

62 cycles are required for hold phase: THOLD1[9:0] = 62,

and 4 cycles are required for refresh period: TREFRESH1[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus Normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 80. Sample and refresh timings

| Buffer State | $t_{SAMP}^{(1)(2)}$ | $t_{REFRESH}^{(2)(3)}$ |
|--------------|--|--|
| Enable | $7 \mu\text{s} + (10 * R_{BON} * C_{SH})$ | $7 \mu\text{s} + (R_{BON} * C_{SH}) * \ln(2 * N_{LSB})$ |
| Disable | $3 \mu\text{s} + (10 * R_{BOFF} * C_{SH})$ | $3 \mu\text{s} + (R_{BOFF} * C_{SH}) * \ln(2 * N_{LSB})$ |

1. In the above formula the settling to the desired code value with $\frac{1}{2}$ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in Sample and hold mode.
3. The tolerated voltage drop during the hold phase "Vd" is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with $\frac{1}{2}$ LSB error accuracy requires $\ln(2 * N_{lsb})$ constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

$$C_{SH} = 100 \text{ nF}$$

$$V_{DD} = 3.0 \text{ V}$$

Sampling phase:

$$t_{SAMP} = 7 \mu\text{s} + (10 * 2000 * 100 * 10^{-9}) = 2.007 \text{ ms}$$

(where $R_{BON} = 2 \text{ k}\Omega$)

Refresh phase:

$$t_{REFRESH} = 7 \mu\text{s} + (2000 * 100 * 10^{-9}) * \ln(2 * 10) = 606.1 \mu\text{s}$$

(where $N_{LSB} = 10$ (10 LSB drop during the hold phase))

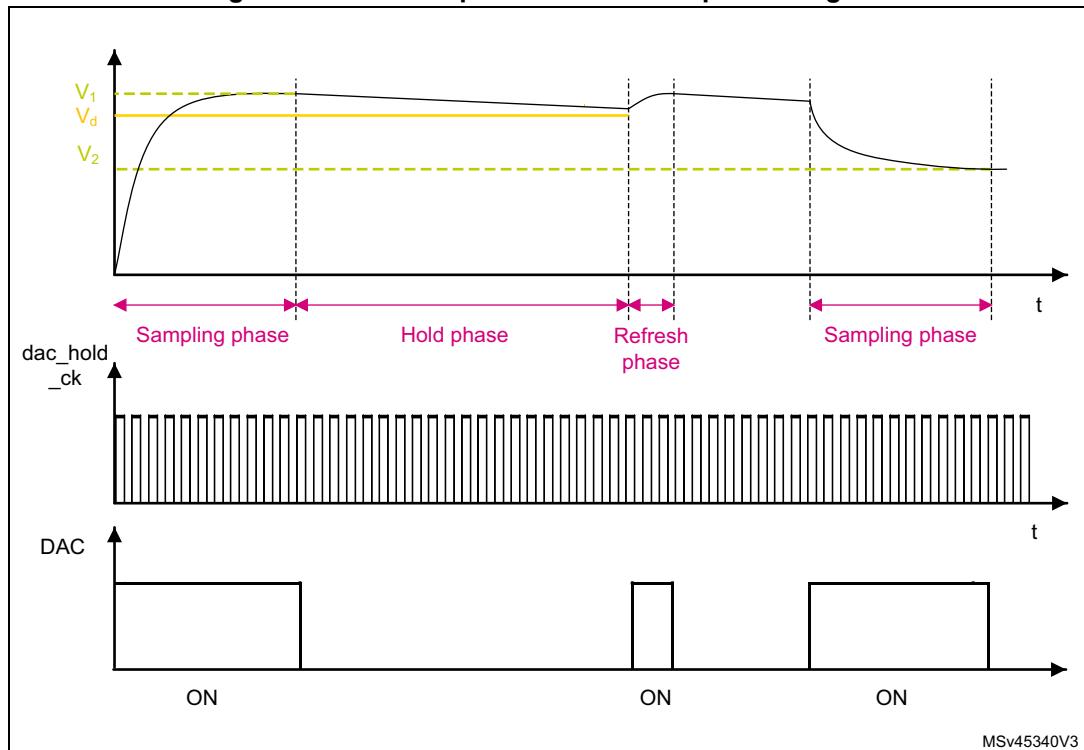
Hold phase:

$$D_V = i_{leak} * t_{hold} / C_{SH} = 0.0073 \text{ V} \text{ (10 LSB of 12bit at 3 V)}$$

$i_{leak} = 150 \text{ nA}$ (worst case on the IO leakage on all the temperature range)

$$t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 \text{ ms}$$

Figure 65. DAC Sample and hold mode phase diagram



Like in Normal mode, the Sample and hold mode has different configurations.

To enable the output buffer, MODE1[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODE1[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

When MODE1[2:0] bits are equal to 111, an internal capacitor, C_{Lint} , holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All Sample and hold phases are interruptible, and any change in DAC_DHR1 immediately triggers a new sample phase.

Table 81. Channel output modes summary

| MODE1[2:0] | | | Mode | Buffer | Output connections |
|------------|---|---|-------------|----------|--|
| 0 | 0 | 0 | Normal mode | Enabled | Connected to external pin |
| 0 | 0 | 1 | | | Connected to external pin and to on chip-peripherals (such as comparators) |
| 0 | 1 | 0 | | Disabled | Connected to external pin |
| 0 | 1 | 1 | | | Connected to on chip peripherals (such as comparators) |

Table 81. Channel output modes summary (continued)

| MODE1[2:0] | | | Mode | Buffer | Output connections |
|------------|---|---|----------------------|----------|--|
| 1 | 0 | 0 | Sample and hold mode | Enabled | Connected to external pin |
| 1 | 0 | 1 | | | Connected to external pin and to on chip peripherals (such as comparators) |
| 1 | 1 | 0 | | Disabled | Connected to external pin and to on chip peripherals (such as comparators) |
| 1 | 1 | 1 | | | Connected to on chip peripherals (such as comparators) |

15.4.12 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{\text{out}} = ((D / 2^N) \times G \times V_{\text{ref}}) + V_{\text{OS}}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{ref} is the nominal full-scale voltage, and V_{OS} is the offset voltage. For an ideal DAC channel, G = 1 and $V_{\text{OS}} = 0$.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{OS} , a calibration is required by a trimming technique.

The calibration is only valid when the DAC channel is operating with buffer enabled (MODE1[2:0] = 0b000 or 0b001 or 0b100 or 0b101). If applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to $V_{\text{REF+}}/2$ signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAG1 bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

The DAC buffer offset is factory trimmed. The default value of OTRIM1[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.

- User trimming

The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, $V_{\text{REF+}}$ values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the nominal factory trimming conditions

In addition, when V_{DD} is removed (example the device enters in Standby or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to EN1 bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODE1[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channel calibration, by setting the CEN1 bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
 - a) Write a code into OTRIM1[4:0] bits, starting by 0b00000.
 - b) Wait for t_{TRIM} delay.
 - c) Check if CAL_FLAG1 bit in DAC_SR is set to 1.
 - d) If CAL_FLAG1 is set to 1, the OTRIM1[4:0] trimming code is found and can be used during device operation to compensate the output value, else increment OTRIM1[4:0] and repeat sub-steps from (a) to (d) again.

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIM1[4:0] bits in a faster way.

The commutation/toggle of CAL_FLAG1 bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIM1[4:0] bits in DAC_CCR register.

Note: A t_{TRIM} delay must be respected between the write to the OTRIM1[4:0] bits and the read of the CAL_FLAG1 bit in DAC_SR register in order to get a correct value. This parameter is specified into datasheet electrical characteristics section.

If V_{DDA} , V_{REF+} and temperature conditions do not change during device operation while it enters more often in Standby and VBAT modes, the software may store the OTRIM1[4:0] bits found in the first user calibration in the flash or in back-up registers, then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CEN1 bit is set, it is not allowed to set EN1 bit.

15.4.13 DAC channel conversion modes

Four conversion modes are possible.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TEN1.
2. Configure the trigger sources by setting different values in the TSEL1[3:0] bits.
3. Load the DAC channel data into the desired DHR registers (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_pclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TEN1.
2. Configure the trigger sources by setting different values in the TSEL1[3:0] bits.
3. Configure the DAC channel WAVE1[1:0] bits as 01 and the same LFSR mask value in the MAMP1[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bits, TEN1.
2. Configure the trigger sources by setting different values in the TSEL1[3:0] bits.
3. Configure the DAC channel WAVE1[1:0] bits as 1x and the same maximum amplitude value in the MAMP1[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DAC channel triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel triangle counter is then updated.

Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the trigger sources by setting different values in STRSTTRIGSEL1[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the DAC channel WAVE1[1:0] bits to 11 and set the same STRSTDATA1[11:0], STINCDATA1[15:0] and STDIR1 values for each register.

When a DAC channel trigger arrives, the DAC channel sawtooth counter updates the DHR1 register and transfers it into DAC_DOR1 (three APB clock cycles later).

15.5 DAC in low-power modes

Table 82. Effect of low-power modes on DAC

| Mode | Description |
|-----------------|---|
| Sleep | No effect, DAC used with DMA |
| Stop 0 / Stop 1 | The DAC remains active with a static value if the Sample and hold mode is selected using LSI clock. |

Table 82. Effect of low-power modes on DAC (continued)

| Mode | Description |
|----------|--|
| Stop 2 | The DAC registers content is lost and must be reinitialized after exiting Stop 2. The DAC must be disabled before entering Stop 2. |
| Standby | The DAC peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode. |
| Shutdown | |

15.6 DAC interrupts

Table 83. DAC interrupts

| Interrupt acronym | Interrupt event | Event flag | Enable control bit | Interrupt clear method | Exit Sleep mode | Exit Stop mode | Exit Standby mode |
|-------------------|-----------------|------------|--------------------|------------------------|-----------------|----------------|-------------------|
| DAC | DMA underrun | DMAUDR1 | DMAUDRI E1 | Write DMAUDRx = 1 | Yes | No | No |

15.7 DAC registers

Refer to [Section 1 on page 51](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

15.7.1 DAC control register (DAC_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. | CEN1 | DMAU DRIE1 | DMAE N1 | MAMP1[3:0] | | | | WAVE1[1:0] | | TSEL1[3] | TSEL1[2] | TSEL1[1] | TSEL1[0] | TEN1 | EN1 |
| | 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 30:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in Normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

- 0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
- 0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
- 0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
- 0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
- 0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
- 0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
- 0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
- 0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
- 1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
- 1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
- 1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
- ≥ 1011 : Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

- 00: wave generation disabled
- 01: Noise wave generation enabled
- 1x: Triangle wave generation enabled

Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection

These bits select the external event used to trigger DAC channel1

- 0000: SWTRIG1
- 0001: dac_ch1_trg1
- 0010: dac_ch1_trg2
- ...
- 1111: dac_ch1_trg15

Refer to the trigger selection tables in [Section 15.4.2: DAC pins and internal signals](#) for details on trigger configuration and mapping.

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.

- 0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_pclk clock cycle later to the DAC_DOR1 register
- 1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_pclk clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHR1 register to the DAC_DOR1 register takes only one dac_pclk clock cycle.

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.

- 0: DAC channel1 disabled
- 1: DAC channel1 enabled

15.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SWTRIG1 |
| | | | | | | | | | | | | | | | w |

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

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

15.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|----------------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | | | | | | | | | | | | DACC1DHR[11:0] |
| | | | | | rw |

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

15.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

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 |
| DACC1DHR[11:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data

These bits are written by software.

They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

15.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | DACC1DHR[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data

These bits are written by software. They specify 8-bit data for DAC channel1.

15.7.6 DAC channel1 data output register (DAC_DOR1)

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 |
| DACC1DOR[11:0] | | | | | | | | | | | | | | | |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output

These bits are read-only, they contain data output for DAC channel1.

15.7.7 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|-----------|---------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BWST1 | CAL_FLAG1 | DMAUDR1 | Res. |
| r | r | rc_w1 | | | | | | | | | | | | | |

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

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after Sample and hold mode enable and is set each time the software writes the register DAC_SHSR1. It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).

- 0: There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written
- 1: There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

- 0: calibration trimming value is lower than the offset correction value
- 1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

- 0: No DMA underrun error condition occurred for DAC channel1
- 1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bits 10:0 Reserved, must be kept at reset value.

15.7.8 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x0000 00XX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | OTRIM1[4:0] |
| | | | | | | | | | | | | rw | rw | rw | rw |

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

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

15.7.9 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | MODE1[2:0] |
| | | | | | | | | | | | | | | | rw |

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

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

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

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

Bits 2:0 **MODE1[2:0]**: DAC channel1 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel1 mode:

– DAC channel1 in Normal mode

000: DAC channel1 is connected to external pin with Buffer enabled

001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled

010: DAC channel1 is connected to external pin with Buffer disabled

011: DAC channel1 is connected to on chip peripherals with Buffer disabled

– DAC channel1 in sample & hold mode

100: DAC channel1 is connected to external pin with Buffer enabled

101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel1 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN1 = 0.

15.7.10 DAC channel1 sample and hold sample time register (DAC_SHSR1)

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | Res. | TSAMPLE1[9:0] | | | | | | | | | |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 9:0 **TSAMPLE1[9:0]**: DAC channel1 sample time (only valid in Sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. In the latter case, the write can be done only when BWST1 of DAC_SR register is low. If BWST1 = 1, the write operation is ignored.

Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI clock period.

15.7.11 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | THOLD1[9:0] | | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

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

Bits 9:0 **THOLD1[9:0]**: DAC channel1 hold time (only valid in Sample and hold mode)

Hold time = (THOLD[9:0]) x LSI clock period

Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

15.7.12 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

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

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

Bits 7:0 **TREFRESH1[7:0]**: DAC channel1 refresh time (only valid in Sample and hold mode)

Refresh time = (TREFRESH[7:0]) x LSI clock period

Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

15.7.13 DAC register map

Table 84 summarizes the DAC registers.

Table 84. DAC register map and reset values

Table 84. DAC 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 |
|-----------|------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|---|---|---|---|---|---|---|---|---|
| 0x40 | DAC_SHSR1 | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x44 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x48 | DAC_SHHR | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4C | DAC_SHRR | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 0x50-0x54 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x58-0x60 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x64-0x68 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

16 Voltage reference buffer (VREFBUF)

This section applies to STM32U073xx and STM32U083xx devices only.

16.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for ADC and also as voltage reference for external components through the VREF+ pin.

16.2 VREFBUF functional description

The internal voltage reference buffer supports two voltages^(a), which are configured with VRS bits in the VREFBUF_CSR register:

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 85. VREF buffer modes

| ENVR | HIZ | VREF buffer configuration |
|------|-----|--|
| 0 | 0 | VREFBUF buffer off mode: – V _{REF+} pin pulled-down to V _{SSA} |
| 0 | 1 | External voltage reference mode (default value): – VREFBUF buffer off – V _{REF+} pin input mode |
| 1 | 0 | Internal voltage reference mode: – VREFBUF buffer on – V _{REF+} pin connected to VREFBUF buffer output |
| 1 | 1 | Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state |

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

16.3 VREFBUF registers

16.3.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x000

Reset value: 0x0000 0002

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | VRR | VRS | HIZ | ENVR |
| | | | | | | | | | | | | r | rw | rw | rw |

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

Bit 3 **VRR**: Voltage reference buffer ready

0: the voltage reference buffer output is not ready.

1: the voltage reference buffer output reached the requested level.

Bit 2 **VRS**: Voltage reference scale

This bit selects the value generated by the voltage reference buffer.

0: Voltage reference set to V_{REF_OUT1} (around 2.048 V).

1: Voltage reference set to V_{REF_OUT2} (around 2.5 V).

Bit 1 **HIZ**: High impedance mode

This bit controls the analog switch to connect or not the V_{REF+} pin.

0: V_{REF+} pin is internally connected to the voltage reference buffer output.

1: V_{REF+} pin is high impedance.

Refer to [Table 85: VREF buffer modes](#) for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR**: Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.

0: Internal voltage reference mode disable (external voltage reference mode).

1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

16.3.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x004

Reset value: 0x0000 00XX

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

Bits 31:6 Reserved, must be kept at reset value.

16.3.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 86. VREFBUF 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 | VREFBUF_CSR | Res. | ENVR | | | |
| | 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 | | | |
| 0x04 | VREFBUF_CCR | Res. | TRIM[5:0] | | | |
| | 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 | | | |

Refer to [Section 2.2](#) for the register boundary addresses.

17 Comparator (COMP)

17.1 Introduction

The devices embed up to two ultra-low-power comparators, COMP1 and COMP2.

The comparators can be used for a variety of functions including:

- Wakeup from low-power mode triggered by an analog signal,
- Analog signal conditioning,
- Cycle-by-cycle current control loop when combined with a PWM output from a timer.

17.2 COMP main features

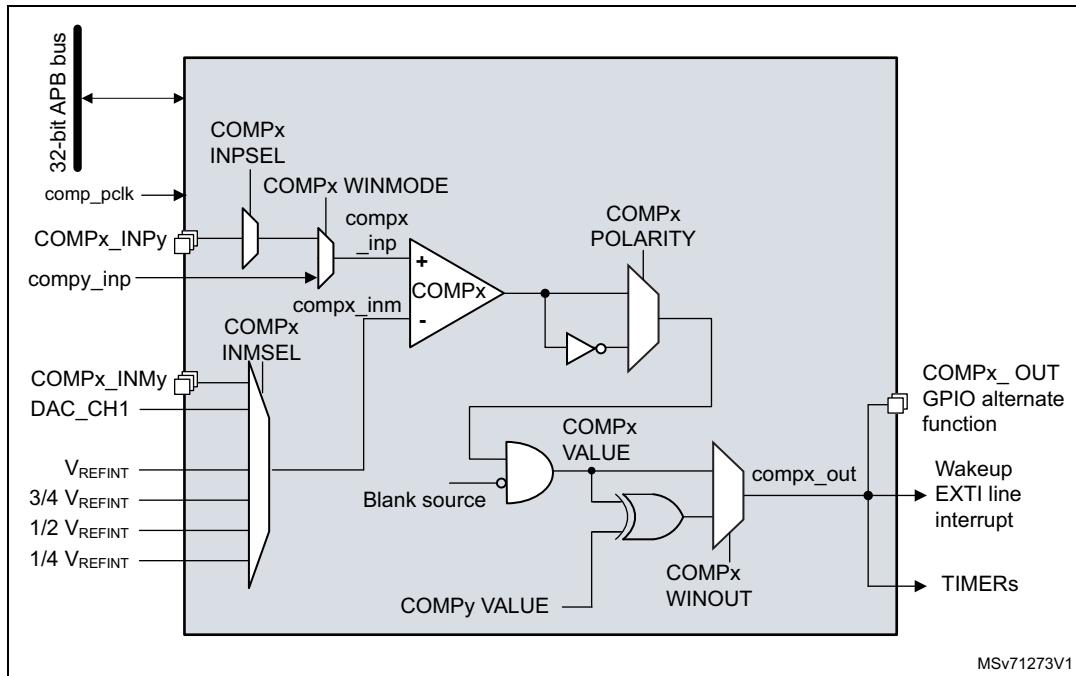
- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
 - Multiplexed I/O pins
 - DAC Channel1
 - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed / consumption
- The outputs can be redirected to an I/O or to timer inputs for triggering:
 - Break events for fast PWM shutdowns
- Comparator outputs with blanking source
- The two comparators can be combined in a window comparator
- Each comparator has interrupt generation capability with wakeup from Sleep and Stop modes (through the EXTI controller)

17.3 COMP functional description

17.3.1 COMP block diagram

The block diagram of the comparators is shown in [Figure 66](#).

Figure 66. Comparator block diagram



17.3.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The comparator output can be connected to the I/Os using the alternate function channel given in “Alternate function mapping” table in the datasheet. The output can also be internally redirected to a variety of timer input for the following purposes:

- Emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- Cycle-by-cycle current control, using OCREF_CLR inputs
- Input capture for timing measures

It is possible to have the comparator output simultaneously redirected internally and externally.

Table 87. COMP internal input/output signals

| Signal name | Signal type | Description |
|-------------|----------------|----------------------------------|
| compx_inm | Analog input | Inverting input source for COMPx |
| compy_inp | Analog input | Noninverting input for COMPy |
| comp_pclk | Digital input | APB clock for both COMP channels |
| compx_out | Digital output | COMPx output |

Table 88. COMP1 noninverting input assignment

| COMP1_INP | COMP1 INPSEL[2:0]⁽¹⁾ |
|------------------|--|
| COMP1_INP1 | 000 |
| COMP1_INP2 | 001 |
| COMP1_INP3 | 010 |
| COMP1_INP4 | 011 |
| COMP1_INP5 | 100 |
| Not connected | 101 |

1. Other combinations are reserved.

Table 89. COMP1 inverting input assignment

| COMP1_INM | COMP1 INMSEL[3:0]⁽¹⁾ |
|--------------------------|--|
| $\frac{1}{4} V_{REFINT}$ | 0000 |
| $\frac{1}{2} V_{REFINT}$ | 0001 |
| $\frac{3}{4} V_{REFINT}$ | 0010 |
| V_{REFINT} | 0011 |
| DAC Channel1 | 0100 |
| COMP1_INM1 | 0101 |
| COMP1_INM2 | 0110 |
| COMP1_INM3 | 0111 |
| COMP1_INM4 | 1000 |
| COMP1_INM5 | 1001 |

1. Other combinations are reserved.

Table 90. COMP2 noninverting input assignment

| COMP2_INP | COMP2 INPSEL[1:0] |
|------------------|--------------------------|
| COMP2_INP1 | 00 |
| COMP2_INP2 | 01 |
| COMP2_INP3 | 10 |
| COMP2_INP4 | 11 |

Table 91. COMP2 inverting input assignment

| COMP2_INM | COMP2 INMSEL[3:0]⁽¹⁾ |
|--------------------------|--|
| $\frac{1}{4} V_{REFINT}$ | 0000 |
| $\frac{1}{2} V_{REFINT}$ | 0001 |
| $\frac{3}{4} V_{REFINT}$ | 0010 |
| V_{REFINT} | 0011 |

Table 91. COMP2 inverting input assignment (continued)

| COMP2_INM | COMP2 INMSEL[3:0] ⁽¹⁾ |
|--------------|----------------------------------|
| DAC Channel1 | 0100 |
| COMP2_INM1 | 0101 |
| COMP2_INM2 | 0110 |
| COMP2_INM3 | 0111 |
| COMP2_INM4 | 1000 |
| COMP2_INM5 | 1001 |

1. Other combinations are reserved.

17.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the APB clock.

There is no clock enable control bit provided in the RCC controller. Reset and clock enable bits are common for COMP and SYSCFG.

Important: The polarity selection logic and the output redirection to the port works independently from the APB clock. This allows the comparator to work even in Stop mode.

17.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

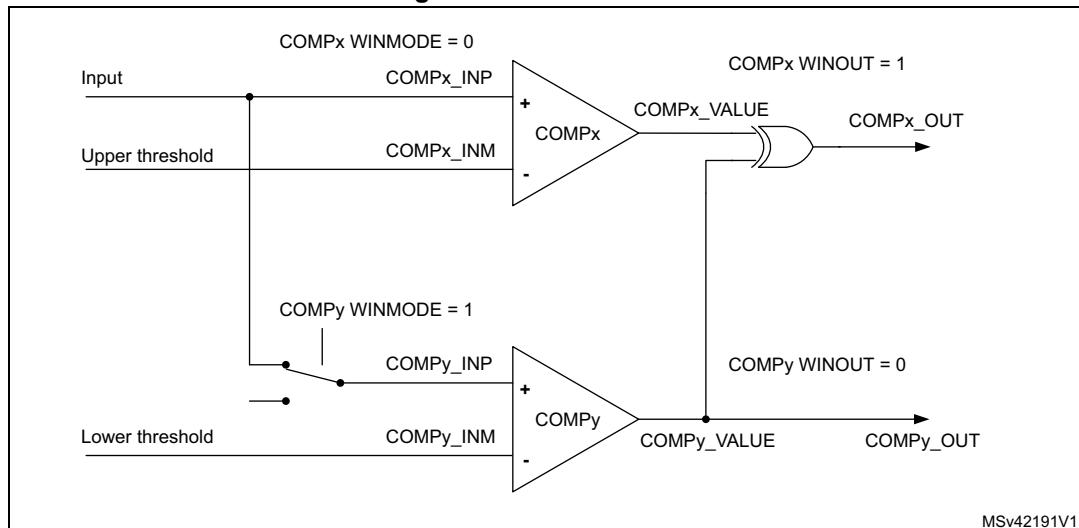
Once the programming is completed, the COMPx LOCK bit can be set to 1. This causes the whole register to become read-only, including the COMPx LOCK bit.

The write protection can only be reset by a MCU reset.

17.3.5 Window comparator

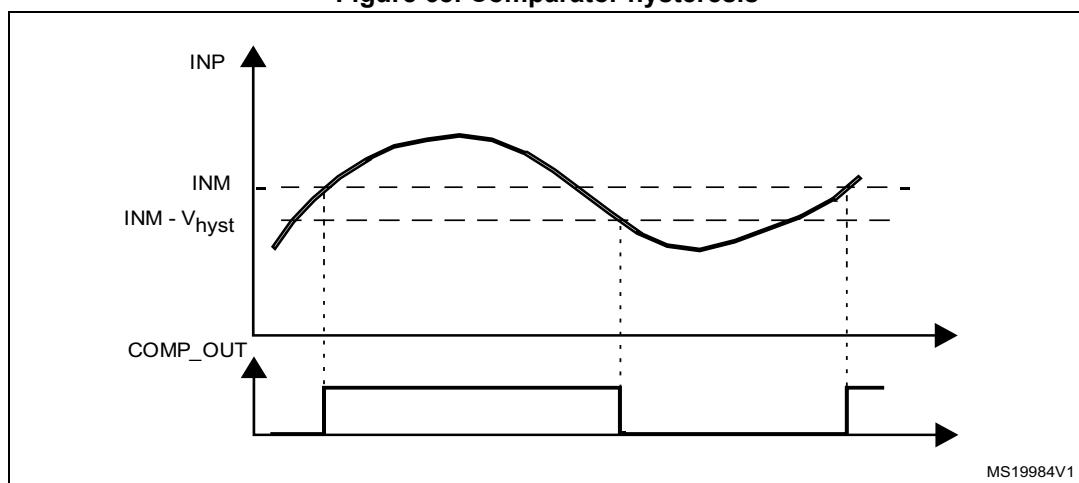
The purpose of window comparator is to monitor the analog voltage if it is within specified voltage range defined by lower and upper threshold.

COMP1 and COMP2 can combine to create a window comparator. The monitored analog voltage is connected to the noninverting (plus) inputs of comparators connected together and the upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators. Two noninverting inputs can be connected internally together by enabling WINMODE bit to save one IO for other purposes.

Figure 67. Window mode

17.3.6 Hysteresis

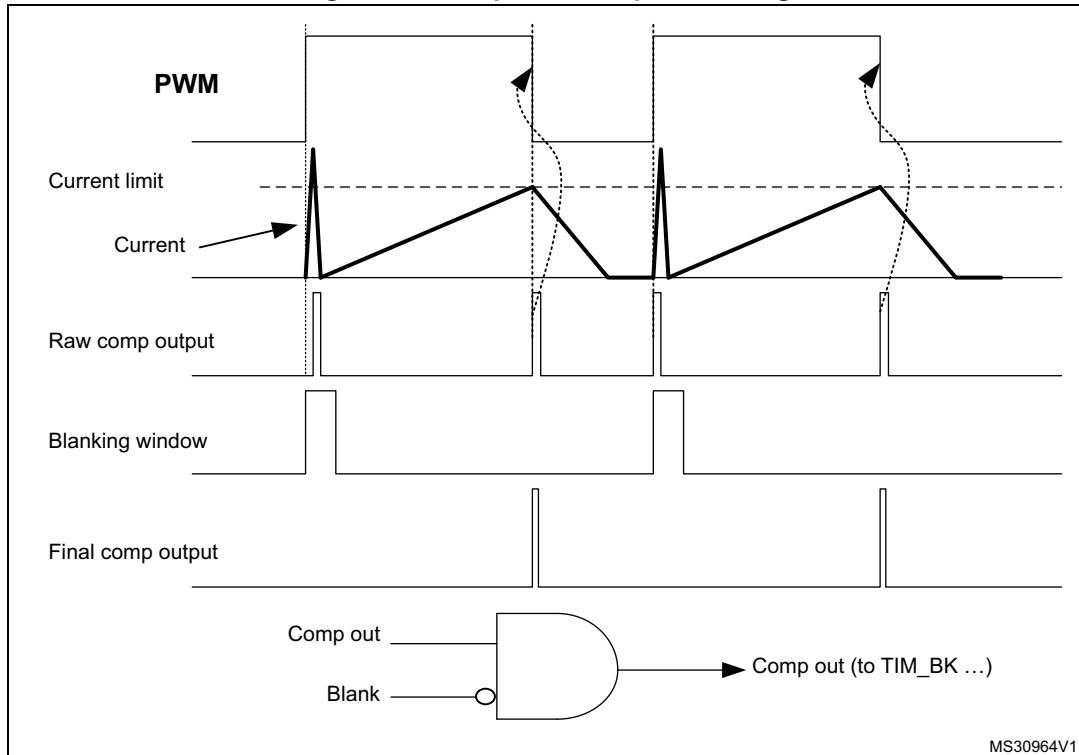
The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

Figure 68. Comparator hysteresis

17.3.7 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). It consists of a selection of a blanking window which is a timer output compare signal. The selection is done by software (refer to the comparator register description for possible blanking signals). Then, the complementary of the blanking signal is ANDed with the comparator output to provide the wanted comparator output. See the example provided in the figure below.

Figure 69. Comparator output blanking



MS30964V1

17.3.8 COMP power and speed modes

COMPx power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The PWRMODE[1:0] bitfields of the COMPx_CSR registers allow setting the comparators to high speed with full power or medium speed with medium power. Refer to [Section 17.6: COMP registers](#).

17.4 COMP low-power modes

Table 92. Comparator behavior in the low-power modes

| Mode | Description |
|-----------------|---|
| Sleep | No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode. |
| Low-power run | No effect. |
| Low-power sleep | No effect. COMP interrupts cause the device to exit the Low-power sleep mode. |
| Stop 0 | No effect on the comparators. |
| Stop 1 | Comparator interrupts cause the device to exit the Stop mode. |

Table 92. Comparator behavior in the low-power modes (continued)

| Mode | Description |
|----------|---|
| Standby | The COMP registers are powered down and must be reinitialized after exiting Standby or Shutdown mode. |
| Shutdown | |

17.5 COMP interrupts

The comparator outputs are internally connected to the Extended interrupts and events controller. Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from low-power modes.

Refer to Interrupt and events section for more details.

To enable COMPx interrupt, it is required to follow this sequence:

1. Configure and enable the EXTI line corresponding to the COMPx output event in interrupt mode and select the rising, falling or both edges sensitivity
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines
3. Enable COMPx.

Table 93. Interrupt control bits

| Interrupt event | Enable control bit | Exit from Sleep mode | Exit from Stop modes | Exit from Standby mode |
|-----------------|--------------------|----------------------|----------------------|------------------------|
| COMP1 output | Through EXTI | Yes | Yes | N/A |
| COMP2 output | Through EXTI | Yes | Yes | N/A |

17.6 COMP registers

17.6.1 Comparator 1 control and status register (COMP1_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------|--------|------|------|---------|-------------|------|----|---------------|----|----|----|------|--------------|-----------|----|
| LOCK | VALUE | Res. | Res. | Res. | Res. | Res. | | BLANKSEL[4:0] | | | | | PWRMODE[1:0] | HYST[1:0] | |
| rs | r | | | | | | 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 |
| POLARITY | WINOUT | Res. | Res. | WINMODE | INPSEL[2:0] | | | INMSEL[3:0] | | | | Res. | Res. | Res. | EN |
| rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | | | | rw |

Bit 31 **LOCK**: COMP1_CSR register lock

This bit is set by software and cleared by a system reset. It locks the comparator 3 control bits. When locked, all register bits are read-only.

0: Not locked

1: Locked

Bit 30 **VALUE**: Comparator 1 output status

This bit is read-only. It reflects the level of the comparator 1 output after the polarity selector and blanking, as indicated in [Figure 66](#).

Bits 29:25 Reserved, must be kept at reset value.

Bits 24:20 **BLANKSEL[4:0]**: Comparator 1 blanking source selector

This bitfield is controlled by software (if not locked). It selects the blanking source:

00000: No blanking

00001: TIM1 OC4 enabled as blanking source

00010: TIM1 OC5 enabled as blanking source

00100: TIM2 OC3 enabled as blanking source

01000: TIM3 OC3 enabled as blanking source

10000: TIM15 OC2 enabled as blanking source

Others: Reserved, must not be used

Bits 19:18 **PWRMODE[1:0]**: Comparator 1 power mode selector

This bitfield is controlled by software (if not locked). It selects the power consumption and as a consequence the speed of the comparator 1:

00: High speed/high power

01-10: Medium speed/medium power

11: Low speed/low power

Bits 17:16 **HYST[1:0]**: Comparator 1 hysteresis selector

This bitfield is controlled by software (if not locked). It selects the hysteresis of the comparator 1:

00: No hysteresis

01: Low hysteresis

10: Medium hysteresis

11: High hysteresis

Bit 15 POLARITY: Comparator 1 polarity selector

This bit is controlled by software (if not locked). It selects the comparator 1 output polarity:

0: Non-inverted

1: Inverted

Bit 14 WINOUT: Comparator 1 output selector

This bit is controlled by software (if not locked). It selects the comparator 1 output:

0: COMP1_VALUE

1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see [Figure 67](#))

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 WINMODE: Comparator 1 noninverting input selector for window mode

This bit is controlled by software (if not locked). It selects the signal for COMP1_INP input of the comparator 1:

0: Signal selected with INPSEL[2:0] bitfield of this register

1: COMP2_INP signal of the comparator 2 (required for window mode, see [Figure 67](#))

Bits 10:8 INPSEL[2:0]: Comparator 1 signal selector for noninverting input

This bitfield is controlled by software (if not locked). It selects the signal for the noninverting input COMP1_INP of the comparator 1 (also see the WINMODE bit):

Refer to [Table 88: COMP1 noninverting input assignment](#).

Bits 7:4 INMSEL[3:0]: Comparator 1 signal selector for inverting input INM

This bitfield is controlled by software (if not locked). It selects the signal for the inverting input COMP1_INM of the comparator 1:

Refer to [Table 89: COMP1 inverting input assignment](#).

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

Bit 0 EN: Comparator 1 enable bit

This bit is controlled by software (if not locked). It enables the comparator 1:

0: Disable

1: Enable

17.6.2 Comparator 2 control and status register (COMP2_CSR)

Address offset: 0x04

Reset value: 0x0000 0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
|----------|--------|------|------|---------|------|-------------|-------------|---------------|----|----|----|----|----|------|--------------|-----------|----|
| LOCK | VALUE | Res. | Res. | Res. | Res. | Res. | Res. | BLANKSEL[4:0] | | | | | | | PWRMODE[1:0] | HYST[1:0] | |
| rs | r | | | | | | | 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 | |
| POLARITY | WINOUT | Res. | Res. | WINMODE | Res. | INPSEL[1:0] | INMSEL[3:0] | | | | | | | Res. | Res. | Res. | EN |
| rw | rw | | | rw | | rw | rw | rw | rw | rw | rw | rw | | | | rw | |

Bit 31 **LOCK:** COMP2_CSR register lock

This bit is set by software and cleared by a system reset. It locks the comparator 3 control bits. When locked, all register bits are read-only.

- 0: Not locked
- 1: Locked

Bit 30 **VALUE:** Comparator 2 output status

This bit is read-only. It reflects the level of the comparator 2 output after the polarity selector and blanking, as indicated in [Figure 66](#).

Bits 29:25 Reserved, must be kept at reset value.

Bits 24:20 **BLANKSEL[4:0]:** Comparator 2 blanking source selector

This bitfield is controlled by software (if not locked). It selects the blanking source:

- 00000: No blanking
- 00001: TIM1 OC4 enabled as blanking source
- 00010: TIM1 OC5 enabled as blanking source
- 00100: TIM2 OC3 enabled as blanking source
- 01000: TIM3 OC3 enabled as blanking source
- 10000: TIM15 OC2 enabled as blanking source
- Others: Reserved, must not be used

Bits 19:18 **PWRMODE[1:0]:** Comparator 2 power mode selector

This bitfield is controlled by software (if not locked). It selects the power consumption and as a consequence the speed of the comparator 2:

- 00: High speed/high power
- 01-10: Medium speed/medium power
- 11: Low speed/low power

Bits 17:16 **HYST[1:0]:** Comparator 2 hysteresis selector

This bitfield is controlled by software (if not locked). It selects the hysteresis of the comparator 2:

- 00: No hysteresis
- 01: Low hysteresis
- 10: Medium hysteresis
- 11: High hysteresis

Bit 15 **POLARITY:** Comparator 2 polarity selector

This bit is controlled by software (if not locked). It selects the comparator 2 output polarity:

- 0: Non-inverted
- 1: Inverted

Bit 14 **WINOUT:** Comparator 2 output selector

This bit is controlled by software (if not locked). It selects the comparator 2 output:

- 0: COMP2_VALUE
- 1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see [Figure 67](#))

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WINMODE:** Comparator 2 noninverting input selector for window mode

This bit is controlled by software (if not locked). It selects the signal for COMP2_INP input of the comparator 2:

- 0: Signal selected with INPSEL[1:0] bitfield of this register
- 1: COMP1_INP signal of the comparator 1 (required for window mode, see [Figure 67](#))

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **INPSEL[1:0]**: Comparator 2 signal selector for noninverting input

This bitfield is controlled by software (if not locked). It selects the signal for the noninverting input COMP2_INP of the comparator 2 (also see the WINMODE bit):

Refer to [Table 90: COMP2 noninverting input assignment](#).

Bits 7:4 **INMSEL[3:0]**: Comparator 2 signal selector for inverting input INM

This bitfield is controlled by software (if not locked). It selects the signal for the inverting input COMP2_INM of the comparator 2:

Refer to [Table 91: COMP2 inverting input assignment](#).

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

Bit 0 **EN**: Comparator 2 enable bit

This bit is controlled by software (if not locked). It enables the comparator 2:

- 0: Disable
- 1: Enable

17.6.3 COMP register map

The following table summarizes the comparator registers.

The comparator registers share SYSCFG peripheral register base addresses.

Table 94. COMP register map and reset values

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

18 Operational amplifiers (OPAMP)

18.1 Introduction

The device embeds an operational amplifier with two inputs and one output. The three I/Os can be connected to the external pins. This enables any type of external interconnections. The operational amplifier can be configured internally as a follower or as an amplifier with a noninverting gain ranging from 2 to 16.

The positive input can be connected to the internal DAC.

The output can be connected to the internal ADC.

18.2 OPAMP main features

- Rail-to-rail input voltage range
- Low input offset voltage (1.5 mV after calibration, 3 mV with factory calibration)
- Low-power mode (current consumption reduced to 30 µA instead of 100 µA)
- Fast wakeup time (10 µs in normal mode, 30 µs in low-power mode)
- Gain bandwidth of 1.6 MHz

18.3 OPAMP functional description

The OPAMP has several modes.

When the OPAMP is disabled, the output is high-Z.

When it is enabled, it can be in calibration mode (where any OPAMP input and output is disconnected), or in functional mode.

There are two functional modes: the low-power mode or the normal mode. In functional mode, the OPAMP inputs and output of the OPAMP are connected as described in the [Section 18.3.3: Signal routing](#).

18.3.1 OPAMP reset and clocks

The operational amplifier clock is necessary for accessing the registers. The clock can be switched off using the peripheral clock enable register when the application does not need to have read or write access to those registers. See OPAMPEN bit in [Section 5.4.15: APB peripheral clock enable register 1 \(RCC_APBENR1\)](#).

The bit OPAEN enables and disables the OPAMP operation. The OPAMP registers configurations should be changed before enabling the OPAEN bit in order to avoid spurious effects on the output.

When the output of the operational amplifier is no more needed, the operational amplifier can be disabled to save power. All the configurations previously set (including the calibration) are maintained while OPAMP is disabled.

18.3.2 Initial configuration

The default configuration of the operational amplifier is a functional mode where the three IOs are connected to external pins. In the default mode, the operational amplifier uses the factory trimming values. See electrical characteristics section of the datasheet for factory trimming conditions, usually the temperature is 30 °C and the voltage is 3 V. The trimming values can be adjusted. See [Section 18.3.5: Calibration](#) for changing the trimming values. The default configuration uses the normal mode, which provides the highest performance. Bit OPALPM can be set in order to switch the operational amplifier to low-power mode and reduced performance. Both normal and low-power mode characteristics are defined in the section “electrical characteristics” of the datasheet. Before utilization, the bit OPA_RANGE of OPAMP_CSR must be set to 1 if V_{DDA} is above 2.4V, or kept at 0 otherwise.

As soon as the OPAEN bit in the OPAMP_CSR register is set, the operational amplifier is functional. The two input pins and the output pin are connected as defined in [Section 18.3.3: Signal routing](#) and the default connection settings can be changed.

Note: *The inputs and output pins must be configured in analog mode (default state) in the corresponding GPIOx_MODER register.*

18.3.3 Signal routing

The OPAMP_CSR register determines the routing for the operational amplifier pins.

The connections of the operational amplifier OPAMP1 are described in the table below.

Table 95. Operational amplifier possible connections

| Signal | Pin | Internal | comment |
|-------------|-----|-----------------------------|---|
| OPAMP1_VINM | PA1 | OPAMP1_OUT or PGA | controlled by bits OPAMODE and VM_SEL. |
| OPAMP1_VINP | PA0 | DAC1_OUT1 | controlled by bit VP_SEL. |
| OPAMP1_VOUT | PA3 | ADC1_IN8 (through the GPIO) | The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC. |

18.3.4 OPAMP modes

The operational amplifier inputs and outputs are all accessible on terminals. The amplifier can be used in multiple configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

Note: *The amplifier output pin is directly connected to the output pad to minimize the output impedance. It cannot be used as a general purpose I/O, even if the amplifier is configured as a PGA and only connected to the ADC channel.*

Note: *The impedance of the signal must be maintained below a level that avoids the input leakage to create significant artifacts (due to a resistive drop in the source). Please refer to the electrical characteristics section in the datasheet for further details.*

Standalone mode (external gain setting mode)

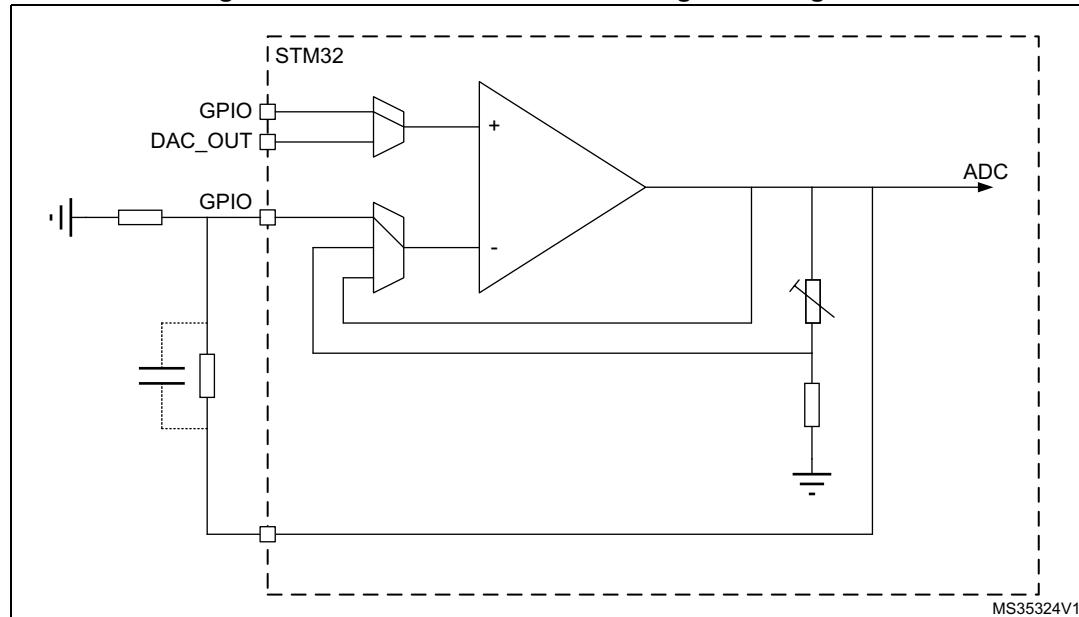
The procedure to use the OPAMP in standalone mode is presented below.

Starting from the default value of OPAMP_CSR, and the default state of GPIOx_MODER, configure bit OPA_RANGE according to the V_{DDA} voltage. As soon as the OPAEN bit is set, the two input pins and the output pin are connected to the operational amplifier.

This default configuration uses the factory trimming values and operates in normal mode (highest performance). The behavior of the OPAMP can be changed as follows:

- OPALPM can be set to “operational amplifier low-power” mode in order to save power.
- USERTRIM can be set to modify the trimming values for the input offset.

Figure 70. Standalone mode: external gain setting mode



Follower configuration mode

The procedure to use the OPAMP in follower mode is presented below.

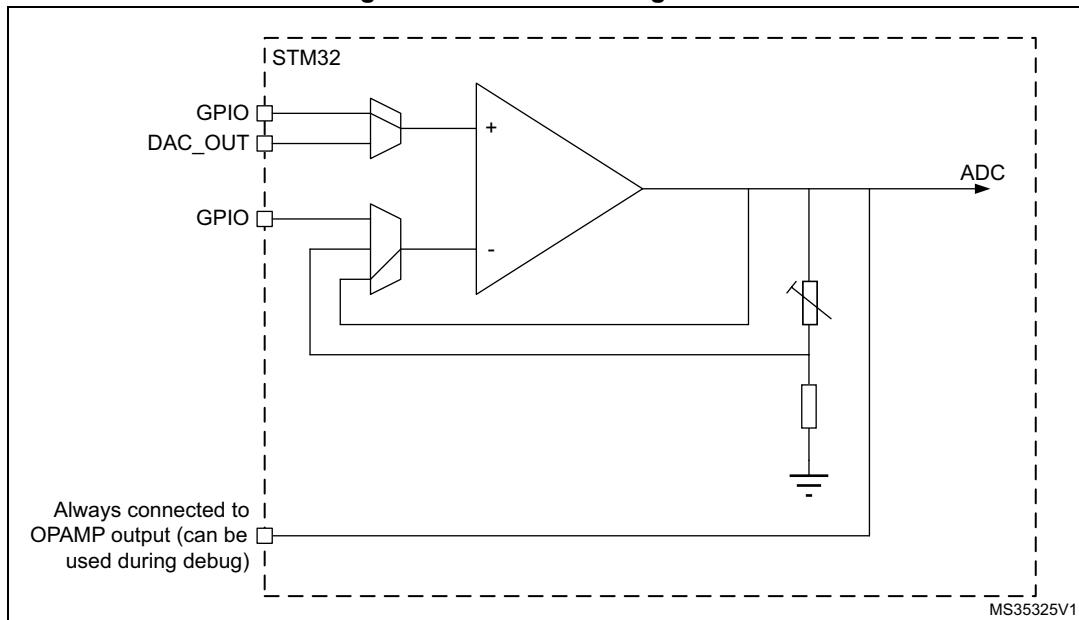
- Configure OPAMODE bits as “internal follower”.
- Configure VP_SEL bits as “GPIO connected to VINP”.
- As soon as the OPAEN bit is set, the signal on pin OPAMP_VINP is copied to pin OPAMP_VOUT.

Note:

The pin corresponding to OPAMP_VINM is free for another usage.

Note:

The signal on the operational amplifier output is also seen as an ADC input. As a consequence, the OPAMP configured in follower mode can be used to perform impedance adaptation on input signals before feeding them to the ADC input, assuming the input signal frequency is compatible with the operational amplifier gain bandwidth specification.

Figure 71. Follower configuration

Programmable Gain Amplifier mode

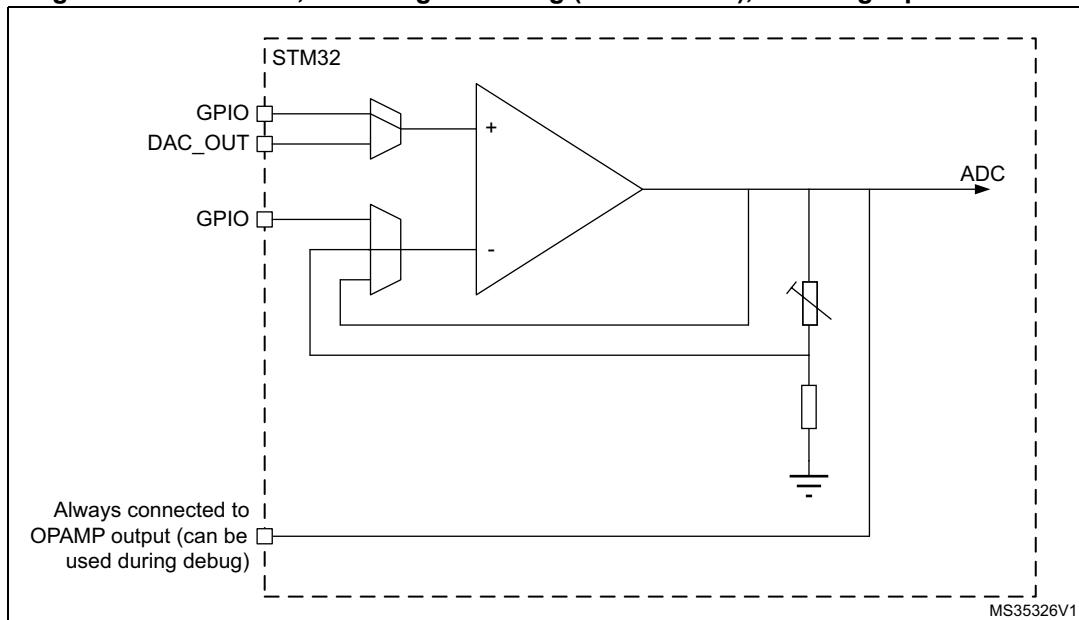
The procedure to use the OPAMP to amplify the amplitude of an input signal is presented below.

- Configure OPAMODE bits as “internal PGA enabled”.
- Configure PGA_GAIN bits as “internal PGA Gain 2, 4, 8 or 16”.
- Configure VM_SEL bits as “inverting not externally connected”.
- Configure VP_SEL bits as “GPIO connected to VINP”.

As soon as the OPAEN bit is set, the selected gain amplifies the signal on the OPAMP_VINP pin. This is visible on the OPAMP_VOUT pin.

Note: To avoid saturation, the input voltage should stay below V_{DDA} divided by the selected gain.

Figure 72. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used



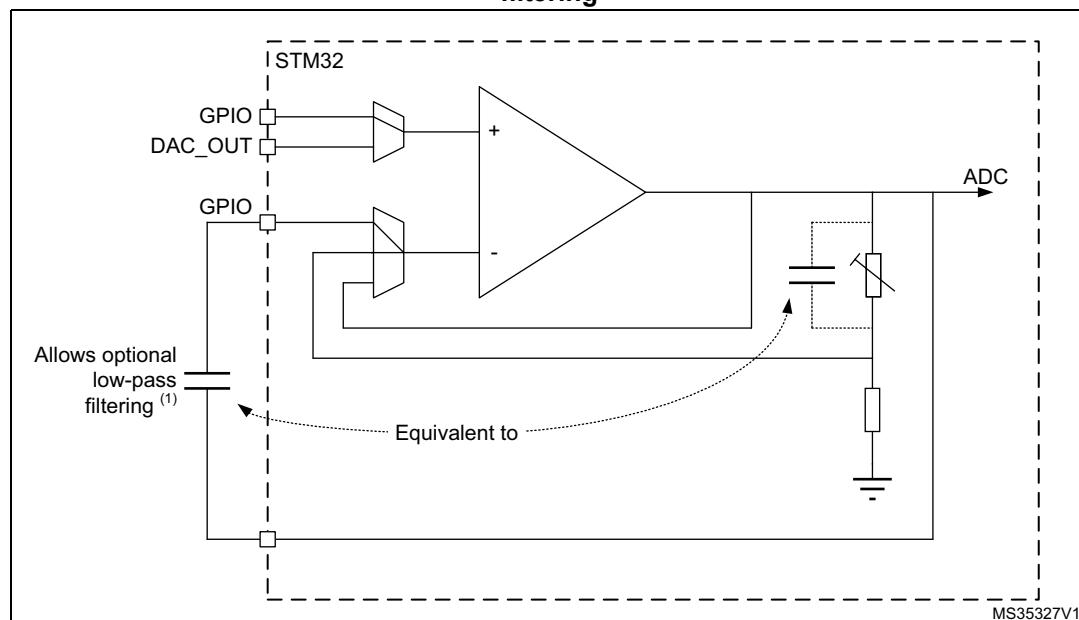
Programmable Gain Amplifier mode with external filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal, with an external filtering, is presented below.

- Configure OPAMODE bits as “internal PGA enabled”.
- Configure PGA_GAIN bits as “internal PGA Gain 2, 4, 8 or 16”.
- Configure VM_SEL bits as “GPIO connected to VINM”.
- Configure VP_SEL bits as “GPIO connected to VINP”.

Any external connection on VINP can be used in parallel with the internal PGA, for example a capacitor can be connected between VOUT and VINM for filtering purposes. See datasheet for the value of resistors used in the PGA resistor network.

Figure 73. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering



1. The gain depends on the cutoff frequency.

18.3.5 Calibration

At startup, the trimming values are initialized with the preset ‘factory’ trimming value.

The user can trim the operational amplifier offset. Specific registers allow different trimming values for normal mode and for low-power mode.

The aim of the calibration is to cancel as much as possible the OPAMP inputs offset voltage. The calibration circuitry reduces the inputs offset voltage to less than +/-1.5 mV within stable voltage and temperature conditions.

For each mode of the operational amplifier, two trimming values need to be trimmed, one for N differential pair and one for P differential pair.

There are two registers for trimming the operational amplifier offset, one for normal mode (OPAMP_OTR) and one low-power mode (OPAMP_LPOTR). Each register is composed of five bits for P differential pair trimming, and five bits for N differential pair trimming. These are the ‘user’ values.

Using the USERTRIM bit in the OPAMP_CSR register, the user can switch from ‘factory’ values to ‘user’ trimmed values. This bit is reset at startup. The ‘factory’ value is then applied by default to the OPAMP trimming registers.

User is liable to change the trimming values in calibration or in functional mode.

The offset trimming registers are typically configured after the calibration operation is initialized by setting bit CALON to 1. When CALON = 1 the inputs of the operational amplifier are disconnected from the functional environment.

- Setting CALSEL to 1 initializes the offset calibration for the P differential pair (low voltage reference used).
- Resetting CALSEL to 0 initializes the offset calibration for the N differential pair (high voltage reference used).

When CALON = 1, the CALOUT bit reflects the influence of the trimming value selected by CALSEL and OPALPM. When the value of CALOUT switches between two consecutive trimming values, this means that those two values are the best trimming values. The CALOUT flag needs up to 1 ms after the trimming value is changed to become steady (see $t_{OFFTRIM}^{max}$ delay specification in the electrical characteristics section of the datasheet).

Note:

The closer the trimming value is to the optimum trimming value, the longer it takes to stabilize. The maximum stabilization time remains below 1 ms in any case.

Table 96. Operating modes and calibration

| Mode | Control bits | | | | Output | |
|------------------------------------|--------------|--------|-------|--------|------------------|-------------|
| | OPAEN | OPALPM | CALON | CALSEL | V _{OUT} | CALOUT flag |
| Normal operating mode | 1 | 0 | 0 | X | analog | 0 |
| Low-power mode | 1 | 1 | 0 | X | analog | 0 |
| Power down | 0 | X | X | X | Z | 0 |
| Offset cal high for normal mode | 1 | 0 | 1 | 0 | analog | X |
| Offset cal low for normal mode | 1 | 0 | 1 | 1 | analog | X |
| Offset cal high for low-power mode | 1 | 1 | 1 | 0 | analog | X |
| Offset cal low for low-power mode | 1 | 1 | 1 | 1 | analog | X |

Calibration procedure

Here are the steps to perform full operational amplifier calibration:

1. Select correct OPA_RANGE in OPAMP_CSR, then set the OPAEN bit in OPAMP_CSR to 1 to enable the operational amplifier.
2. Set the USERTRIM bit in the OPAMP_CSR register to 1.
3. Choose a calibration mode (refer to [Table 96: Operating modes and calibration](#)). The steps 3 to 4 must be repeated four times. For the first iteration, select :
 - Normal mode, offset cal high (N differential pair)
 The above calibration mode corresponds to OPALPM=0 and CALSEL=0 in the OPAMP_CSR register.
4. Increment TRIMOFFSETN[4:0] in OPAMP_OTR, starting from 00000b and until CALOUT changes to 1 in OPAMP_CSR.

Note: CALOUT switches from 0 to 1 for offset cal high and from 1 to 0 for offset cal low.

Note: Between the write to the OPAMP_OTR register and the read of the CALOUT value, make sure to wait for the $t_{OFFTRIM}^{max}$ delay specified in the electrical characteristics section of the datasheet, to get the correct CALOUT value.

The commutation means that the offset is correctly compensated and that the corresponding trim code must be saved in the OPAMP_OTR register.

Repeat steps 3 to 4 for:

- Normal_mode and offset cal low
- Low power mode and offset cal high
- Low power mode and offset cal low

If a mode is not in use, you do not need to perform the corresponding calibration.

Note: During the whole calibration phase the external connection of the operational amplifier output must not pull up or down currents higher than 500 μ A.

During the calibration procedure, it is necessary to set up OPAMODE bits as 00 or 01 (PGA disable) or 11 (internal follower).

18.4 OPAMP low-power modes

Table 97. Effect of low-power modes on the OPAMP

| Mode | Description |
|-----------------|---|
| Sleep | No effect. |
| Stop 0 / Stop 1 | No effect, OPAMP registers content is kept. |
| Stop 2 | OPAMP registers content is kept. OPAMP must be disabled before entering Stop 2 mode. |
| Standby | The OPAMP registers are powered down and must be re-initialized after exiting Standby or Shutdown mode. |
| Shutdown | |

18.5 OPAMP registers

18.5.1 OPAMP1 control/status register (OPAMP1_CSR)

Address offset: 0x000

Reset value: 0x0000 0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-----------|-----------|---------|-------|------|--------|-------------|------|------|---------------|--------------|---------|-------|------|------|------|------|
| OPA_RANGE | Res. | 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 |
| CAL_OUT | USER TRIM | CAL SEL | CALON | Res. | VP_SEL | VM_SEL[9:8] | Res. | Res. | PGA_GAIN[5:4] | OPAMODE[3:2] | OPA_LPM | OPAEN | | | | |
| r | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | w | rw | rw | | |

Bit 31 **OPA_RANGE**: Operational amplifier power supply range for stability

All AOP must be in power down to allow AOP-RANGE bit write. It applies to all AOP embedded in the product.

0: Low range (VDDA < 2.4V)

1: High range (VDDA > 2.4V)

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

Bit 15 **CALOUT**: Operational amplifier calibration output

During calibration mode offset is trimmed when this signal toggle.

Bit 14 **USERTRIM**: allows to switch from 'factory' AOP offset trimmed values to AOP offset 'user' trimmed values

This bit is active for both mode normal and low-power.

0: 'factory' trim code used

1: 'user' trim code used

Bit 13 **CALSEL**: Calibration selection

0: NMOS calibration (200mV applied on OPAMP inputs)

1: PMOS calibration (VDDA-200mV applied on OPAMP inputs)

Bit 12 **CALON**: Calibration mode enabled

0: Normal mode

1: Calibration mode (all switches opened by HW)

Bit 11 Reserved, must be kept at reset value.

Bit 10 **VP_SEL**: Non inverted input selection

0: GPIO connected to VINP

1: DAC connected to VINP

Bits 9:8 **VM_SEL[9:8]**: Inverting input selection

These bits are used only when OPAMODE = 00, 01 or 10.

00: GPIO connected to VINM (valid also in PGA mode for filtering)

01: Reserved

1x: Inverting input not externally connected. These configurations are valid only when OPAMODE = 10 (PGA mode)

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

Bits 5:4 **PGA_GAIN[5:4]**: Operational amplifier Programmable amplifier gain value

- 00: internal PGA Gain 2
- 01: internal PGA Gain 4
- 10: internal PGA Gain 8
- 11: internal PGA Gain 16

Bits 3:2 **OPAMODE[3:2]**: Operational amplifier PGA mode

- 00: internal PGA disable
- 01: internal PGA disable
- 10: internal PGA enable, gain programmed in PGA_GAIN
- 11: internal follower

Bit 1 **OPALPM**: Operational amplifier Low Power Mode

The operational amplifier must be disable to change this configuration.

- 0: operational amplifier in normal mode
- 1: operational amplifier in low-power mode

Bit 0 **OPAEN**: Operational amplifier Enable

- 0: operational amplifier disabled
- 1: operational amplifier enabled

18.5.2 OPAMP1 offset trimming register in normal mode (OPAMP1_OTR)

Address offset: 0x04

Reset value: 0x0000 XXXX (factory trimmed values)

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

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

Bits 12:8 **TRIMOFFSETP[12:8]**: Trim for PMOS differential pairs

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

Bits 4:0 **TRIMOFFSETN[4:0]**: Trim for NMOS differential pairs

18.5.3 OPAMP1 offset trimming register in low-power mode (OPAMP1_LPOTR)

Address offset: 0x08

Reset value: 0x0000 XXXX (factory trimmed values)

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

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

Bits 12:8 **TRIMLPOFFSETP[12:8]**: Low-power mode trim for PMOS differential pairs

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

Bits 4:0 **TRIMLPOFFSETN[4:0]**: Low-power mode trim for NMOS differential pairs

18.5.4 OPAMP register map

Table 98. OPAMP register map and reset values

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------|---------------------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 0x00 | OPAMP1_CSR | OFA_RANGE | Res. |
| | | Reset value | 0 | | | | | | | | | | | | | | |
| 0x04 | OPAMP1_OTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | Reset value | | | | | | | | | | | | | | | |
| 0x08 | OPAMP1_LPOTR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | Reset value | | | | | | | | | | | | | | | |

1. Factory trimmed values.

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

19 Liquid crystal display controller (LCD)

This section applies to STM32U083xx and STM32U073xx devices only.

19.1 LCD introduction

The LCD controller is a digital controller/driver for monochrome passive liquid crystal display (LCD) with up to eight common terminals, and up to 52 segment terminals to drive 208 (4x52) or 384 (8x48) LCD picture elements (pixels). The exact number of terminals depends on the device pinout as described in the datasheet.

The LCD is made up of several segments (pixels or complete symbols) that can be turned visible or invisible. Each segment consists of a layer of liquid crystal molecules aligned between two electrodes. When a voltage greater than a threshold voltage is applied across the liquid crystal, the segment becomes visible. The segment voltage must be alternated to avoid an electrophoresis effect in the liquid crystal (which degrades the display). The waveform across a segment must then be generated, so to avoid having a direct current (DC).

Glossary

Bias: number of voltage levels used when driving an LCD

It is defined as $1 / (\text{number of voltage levels used to drive an LCD display} - 1)$.

Boost circuit: contrast controller circuit

Common: electrical connection terminal connected to several segments (52 segments)

Duty ratio: number defined as $1 / (\text{number of common terminals on a given LCD display})$

Frame: one period of the waveform written to a segment

Frame rate: number of frames per second (number of times the LCD segments are energized per second)

LCD (liquid crystal display): passive display panel with terminals leading directly to a segment

Segment: smallest viewing element (a single bar or dot that is used to help create a character on an LCD display)

19.2 LCD main features

- Highly flexible frame rate control
- Static, 1/2, 1/3, 1/4, and 1/8 duty supported
- Static, 1/2, 1/3, and 1/4 bias supported
- Double-buffered memory allowing data in LCD_RAM registers to be updated at any time by the application firmware without affecting the integrity of the data displayed
 - LCD data RAM of up to 16 x 32-bit registers which contain pixel information (active/inactive)
- Software selectable LCD output voltage (contrast) from V_{LCDmin} to V_{LCDmax}

- No need for external analog components:
 - stepup converter embedded to generate an internal V_{LCD} voltage higher than V_{DD}
 - software selection between external and internal V_{LCD} voltage source. In case of an external source, the internal boost circuit is disabled to reduce power consumption.
 - resistive network embedded to generate intermediate V_{LCD} voltages
 - structure of the resistive network configurable by software to adapt the power consumption to match the capacitive charge required by the LCD panel
 - Integrated voltage output buffers for higher LCD driving capability.
- Contrast that can be adjusted using two different methods:
 - When using the internal step-up converter, the software can adjust V_{LCD} between V_{LCDmin} and V_{LCDmax} .
 - Programmable dead time (up to eight phase periods) between frames
- Full support of low-power modes: The LCD controller can be displayed in Sleep, Low-power run, Low-power sleep, and Stop modes, or can be fully disabled to reduce power consumption.
- Built-in phase inversion for reduced power consumption and EMI (electromagnetic interference)
- Start-of-frame interrupt to synchronize the software when updating the LCD data RAM
- Blink capability:
 - Up to 1, 2, 3, 4, 8, or all pixels which can be programmed to blink at a configurable frequency
 - Software adjustable blink frequency to achieve around 0.5 Hz, 1 Hz, 2 Hz, or 4 Hz
- Used LCD segment and common pins must be configured as GPIO alternate functions, and unused segment and common pins can be used for GPIO or for another peripheral alternate function.

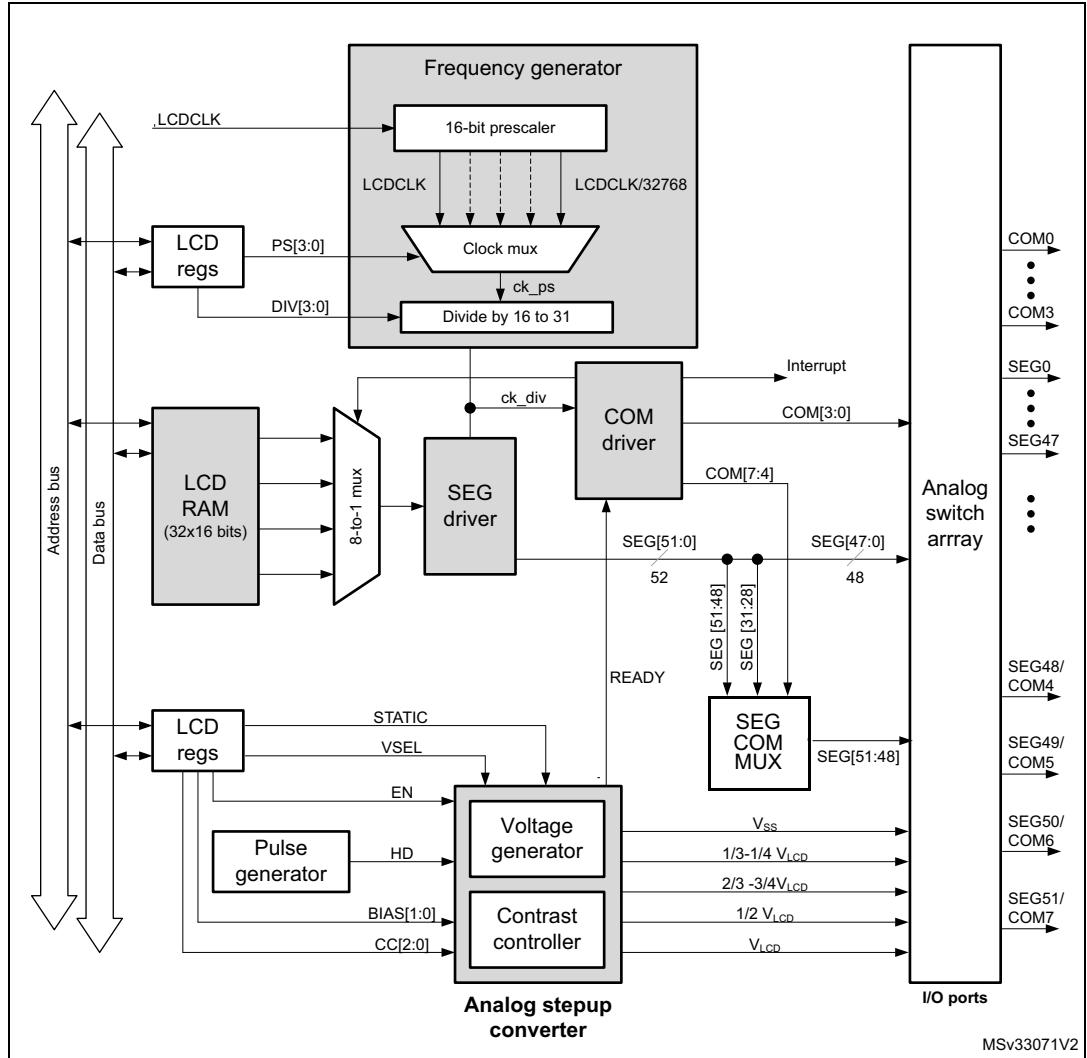
Note: When the LCD relies on the internal stepup converter, the VLCD pin must be connected to V_{SS} with a capacitor. Its typical value is $1 \mu F$ (see C_{EXT} value in the datasheet for details).

19.3 LCD functional description

19.3.1 General description

The LCD controller has five main blocks (see [Figure 74](#)):

Figure 74. LCD controller block diagram



Note: *LCDCLK is the same as RTCCLK. Refer to the RTC/LCD clock description in the RCC section of the reference manual.*

The frequency generator allows the user to achieve various LCD frame rates starting from an LCD input clock frequency (LCDCLK) which can vary from 32 kHz up to 1 MHz.

Three different clock sources can be used to provide the LCD clock (LCDCLK/RTCCLK):

- 32 kHz low-speed external RC (LSE)
- 32 kHz low-speed internal RC (LSI)

19.3.2 Frequency generator

This clock source must be stable in order to obtain accurate LCD timing, and hence to minimize DC voltage offset across LCD segments. The input clock LCDCLK can be divided by any value from 1 to $2^{15} \times 31$ (see [Section 19.6.2](#)). The frequency generator consists of a prescaler (16-bit ripple counter), and a 16 to 31 clock divider.

PS[3:0] in LCD_FCR selects LCDCLK divided by $2^{\text{PS}[3:0]}$. If a finer resolution rate is required, DIV[3:0] in LCD_FCR can be used to divide the clock further by 16 to 31.

In this way, the user can roughly scale the frequency, and then fine-tune it by linearly scaling the clock with the counter. The output of the frequency generator block is $f_{\text{ck_div}}$, which constitutes the time base for the entire LCD controller. The ck_div frequency is equivalent to the LCD phase frequency, rather than the frame frequency (they are equal only in case of static duty). The frame frequency (f_{frame}) is obtained from $f_{\text{ck_div}}$ by dividing it by the number of active common terminals (or by multiplying it for the duty). Thus the relation between the input clock frequency (f_{LCDCLK}) of the frequency generator and its output clock frequency $f_{\text{ck_div}}$ is:

$$f_{\text{ckdiv}} = \frac{f_{\text{LCDCLK}}}{2^{\text{PS}} \times (16 + \text{DIV})}$$

$$f_{\text{frame}} = f_{\text{ckdiv}} \times \text{duty}$$

This makes the frequency generator very flexible. An example of frame rate calculation is shown in the table below.

Table 99. Example of frame rate calculation

| LCDCLK | PS[3:0] | DIV[3:0] | Ratio | Duty | f_{frame} |
|------------|---------|----------|-------|--------|--------------------|
| 32.768 kHz | 3 | 1 | 136 | 1/8 | 30.12 Hz |
| 32.768 kHz | 4 | 1 | 272 | 1/4 | 30.12 Hz |
| 32.768 kHz | 4 | 6 | 352 | 1/3 | 31.03 Hz |
| 32.768 kHz | 5 | 1 | 544 | 1/2 | 30.12 Hz |
| 32.768 kHz | 6 | 1 | 1088 | static | 30.12 Hz |
| 32.768 kHz | 1 | 4 | 40 | 1/8 | 102.40 Hz |
| 32.768 kHz | 2 | 4 | 80 | 1/4 | 102.40 Hz |
| 32.768 kHz | 2 | 11 | 108 | 1/3 | 101.14 Hz |
| 32.768 kHz | 3 | 4 | 160 | 1/2 | 102.40 Hz |
| 32.768 kHz | 4 | 4 | 320 | static | 102.40 Hz |
| 1.00 MHz | 6 | 3 | 1216 | 1/8 | 102.80 Hz |
| 1.00 MHz | 7 | 3 | 2432 | 1/4 | 102.80 Hz |
| 1.00 MHz | 7 | 10 | 3328 | 1/3 | 100.16 Hz |
| 1.00 MHz | 8 | 3 | 4864 | 1/2 | 102.80 Hz |
| 1.00 MHz | 9 | 3 | 9728 | static | 102.80 Hz |

The frame frequency must be selected to be within a range of around ~30 Hz to ~100 Hz. It is a compromise between power consumption and the acceptable refresh rate. In addition, a dedicated blink prescaler selects the blink frequency. This frequency is defined as:

$$f_{BLINK} = f_{ck_div}/2^{(BLINKF + 3)},$$

with $BLINKF[2:0] = 0, 1, 2, \dots, 7$

The blink frequency achieved is in the range of 0.5 Hz, 1 Hz, 2 Hz, or 4 Hz.

19.3.3 Common driver

Common signals are generated by the common driver block (see [Figure 74](#)).

COM signal bias

Each COM signal has identical waveforms, but different phases. It has its max amplitude V_{LCD} or V_{SS} only in the corresponding phase of a frame cycle, while during the other phases, the signal amplitude is:

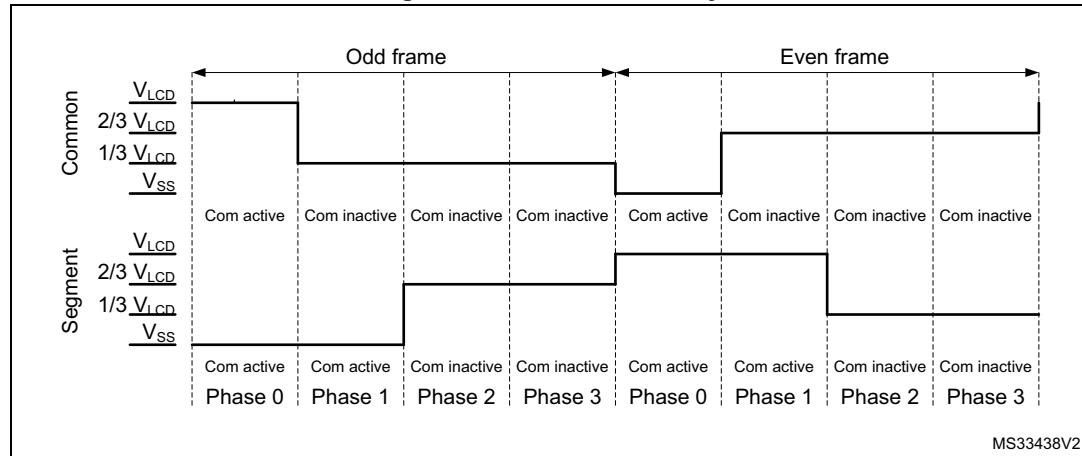
- 1/4 V_{LCD} or 3/4 V_{LCD} in case of 1/4 bias
- 1/3 V_{LCD} or 2/3 V_{LCD} in case of 1/3 bias
- 1/2 V_{LCD} in case of 1/2 bias

The selection between 1/2, 1/3, and 1/4 bias mode can be done through BIAS[1:0] in LCD_CR.

A pixel is activated when both of its corresponding common and segment lines are active during the same phase: when the voltage difference between common and segment is maximum during this phase. Common signals are phase inverted in order to reduce EMI.

As shown in the figure below, with phase inversion, there is a mean voltage of 1/2 V_{LCD} at the end of every odd cycle.

Figure 75. 1/3 bias, 1/4 duty



In case of 1/2 bias (BIAS = 01), the VLCD pin generates an intermediate voltage equal to 1/2 V_{LCD} on node b for odd and even frames (see [Figure 78](#)).

COM signal duty

Depending on DUTY[2:0] in LCD_CR, the COM signals are generated with static duty (see [Figure 77](#)), 1/2 duty (see [Figure 78](#)), 1/3 duty (see [Figure 79](#)), 1/4 duty (see [Figure 80](#)), or 1/8 duty (see [Figure 81](#)).

COM[n] n[0 to 7] is active during phase n in the odd frame, so the COM pin is driven to V_{LCD} .

During phase n of the even frame, the COM pin is driven to V_{SS} .

In the case of 1/3 or 1/4) bias, COM[n] is inactive during phases other than n, so the COM pin is driven to 1/3 (1/4) V_{LCD} during odd frames, and to 2/3 (3/4) V_{LCD} during even frames.

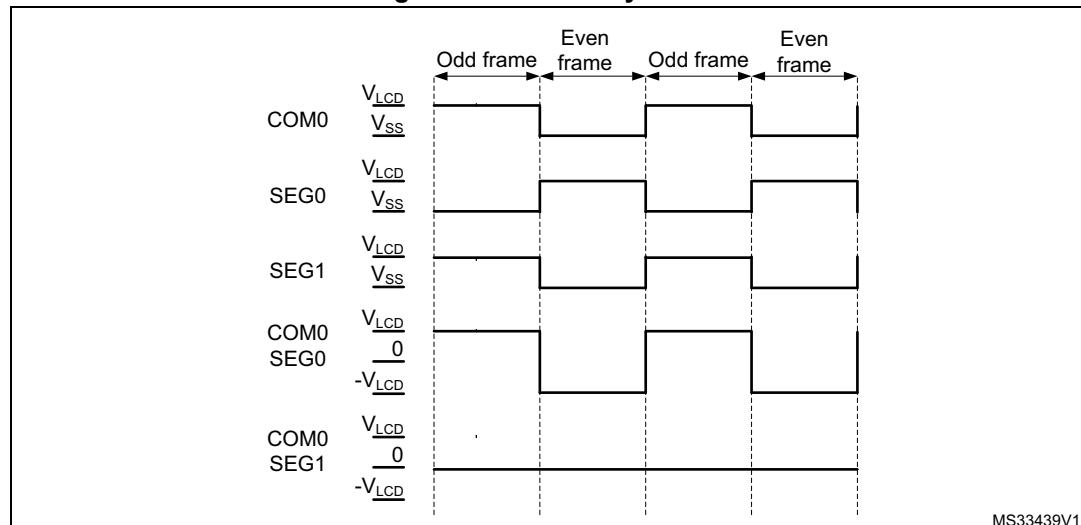
In the case of 1/2 bias, if COM[n] is inactive during phases other than n, the COM pin is always driven (odd and even frame) to 1/2 V_{LCD} .

When static duty is selected, the segment lines are not multiplexed, which means that each segment output corresponds to one pixel. In this way, only up to 51 pixels can be driven. COM[0] is always active while COM[7:1] are not used and are driven to V_{SS} .

When LCDEN in LCD_CR is reset, all common lines are pulled down to V_{SS} , and the ENS flag in LCD_SR becomes 0. Static duty means that COM[0] is always active, and only two voltage levels are used for segment and common lines: V_{LCD} and V_{SS} . A pixel is active if the corresponding SEG line has a voltage opposite to that of the COM, and inactive when the voltages are equal. In this way the LCD has maximum contrast (see [Figure 76](#), [Figure 77](#)).

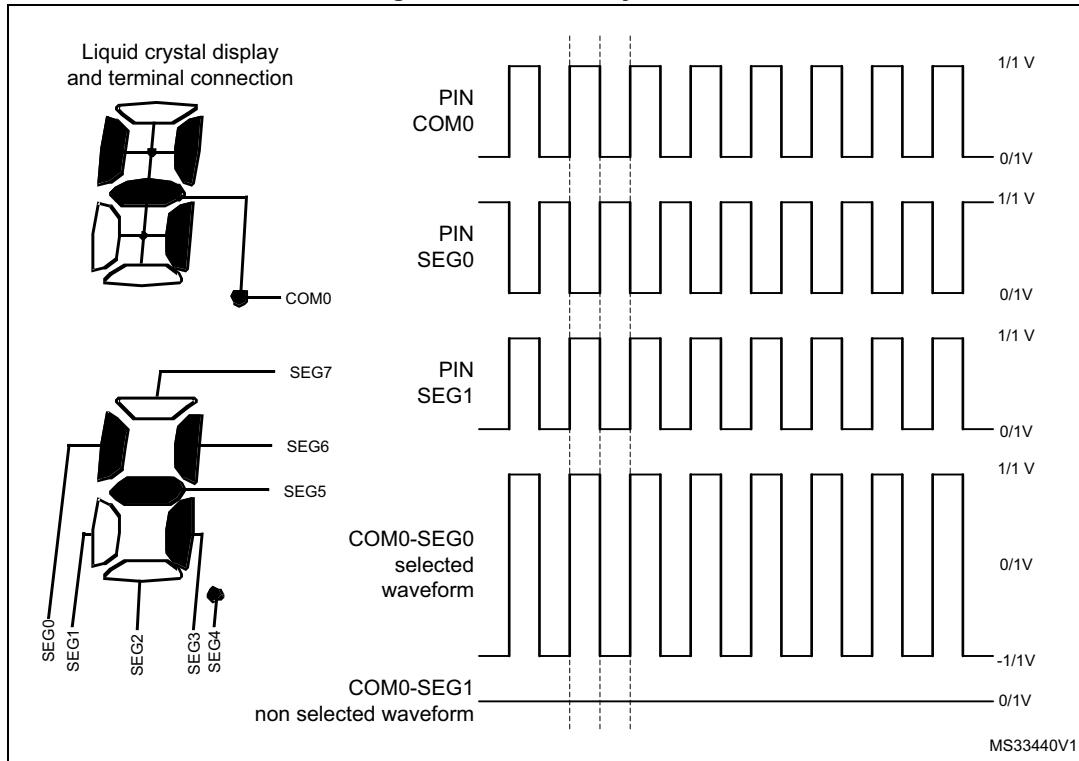
In the figure below, pixel 0 is active while pixel 1 is inactive.

Figure 76. Static duty case 1



In each frame, there is only one phase, this is why f_{frame} is equal to f_{LCD} . If 1/4 duty is selected there are four phases in a frame in which COM[0] is active during phase 0, COM[1] is active during phase 1, COM[2] is active during phase 2, and COM[3] is active during phase 3.

Figure 77. Static duty case 2

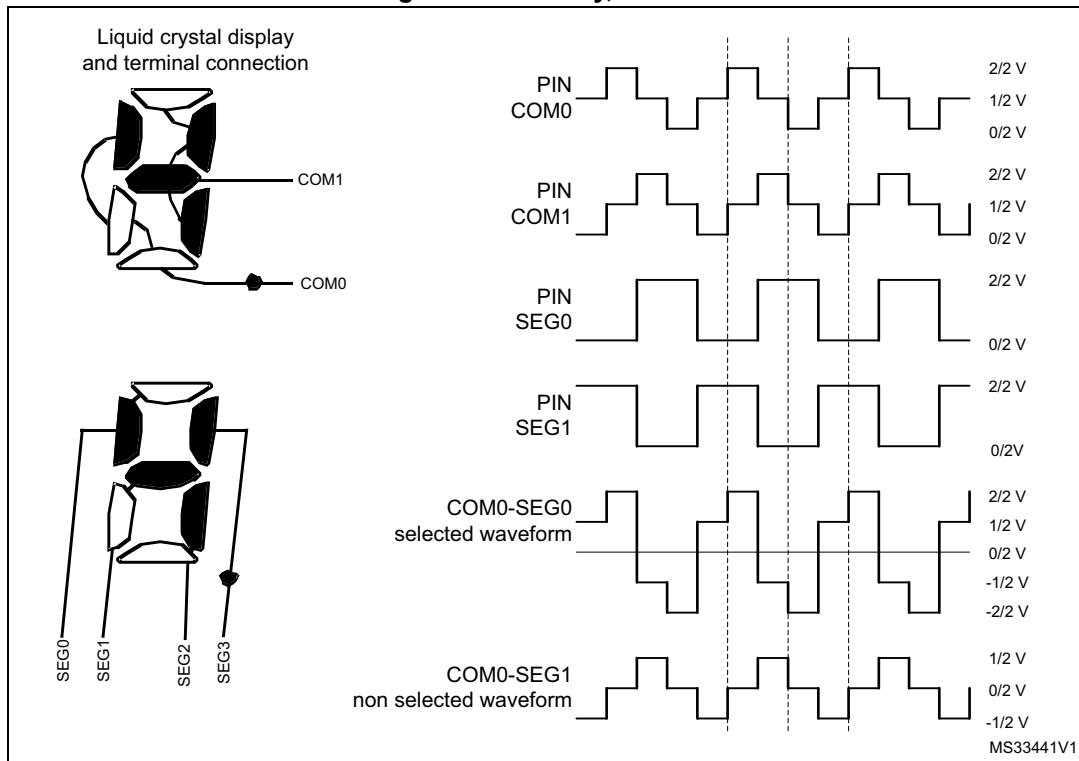


In this mode, the segment terminals are multiplexed, and each of them control four pixels. A pixel is activated only when both of its corresponding SEG and COM lines are active in the same phase. In case of 1/4 duty, to deactivate pixel 0 connected to COM[0], SEG[0] needs to be inactive during the phase 0 when COM[0] is active. To activate pixel 0 connected to COM[1], SEG[0] needs to be active during phase 1 when COM[1] is active (see [Figure 80](#)). To activate pixels from 0 to 51 connected to COM[0], SEG[0:51] needs to be active during phase 0 when COM[0] is active. These considerations can be extended to the other pixels.

Eight-to-one mux

When COM[0] is active the common driver block, it also drives the eight-to-one mux shown in [Figure 74](#) in order to select the content of first two RAM register locations. When COM[7] is active, the output of the eight-to-one mux is the content of the last two RAM locations.

Figure 78. 1/2 duty, 1/2 bias



19.3.4 Segment driver

The segment driver block controls the SEG lines according to the pixel data coming from the eight-to-one mux driven in each phase by the common driver block.

In the case of 1/4 or 1/8 duty

When COM[0] is active, the pixel information (active/inactive) related to the pixel connected to COM[0] (content of the first two LCD_RAM locations) goes through the eight-to-one mux.

The SEG[n] pin n [0 to 51] is driven to V_{SS} (indicating pixel n is active when COM[0] is active) in phase 0 of the odd frame.

The SEG[n] pin is driven to V_{LCD} in phase 0 of the even frame. If pixel n is inactive then the SEG[n] pin is driven to 2/3 (2/4) V_{LCD} in the odd frame or 1/3 (2/4) V_{LCD} in the even frame (current inversion in V_{LCD} pad) (see [Figure 75](#)).

In case of 1/2 bias, if the pixel is inactive the SEG[n] pin is driven to V_{LCD} in the odd and to V_{SS} in the even frame.

When the LCD controller is disabled (LCDEN cleared in LCD_CR), then the SEG lines are pulled down to V_{SS}.

Figure 79. 1/3 duty, 1/3 bias

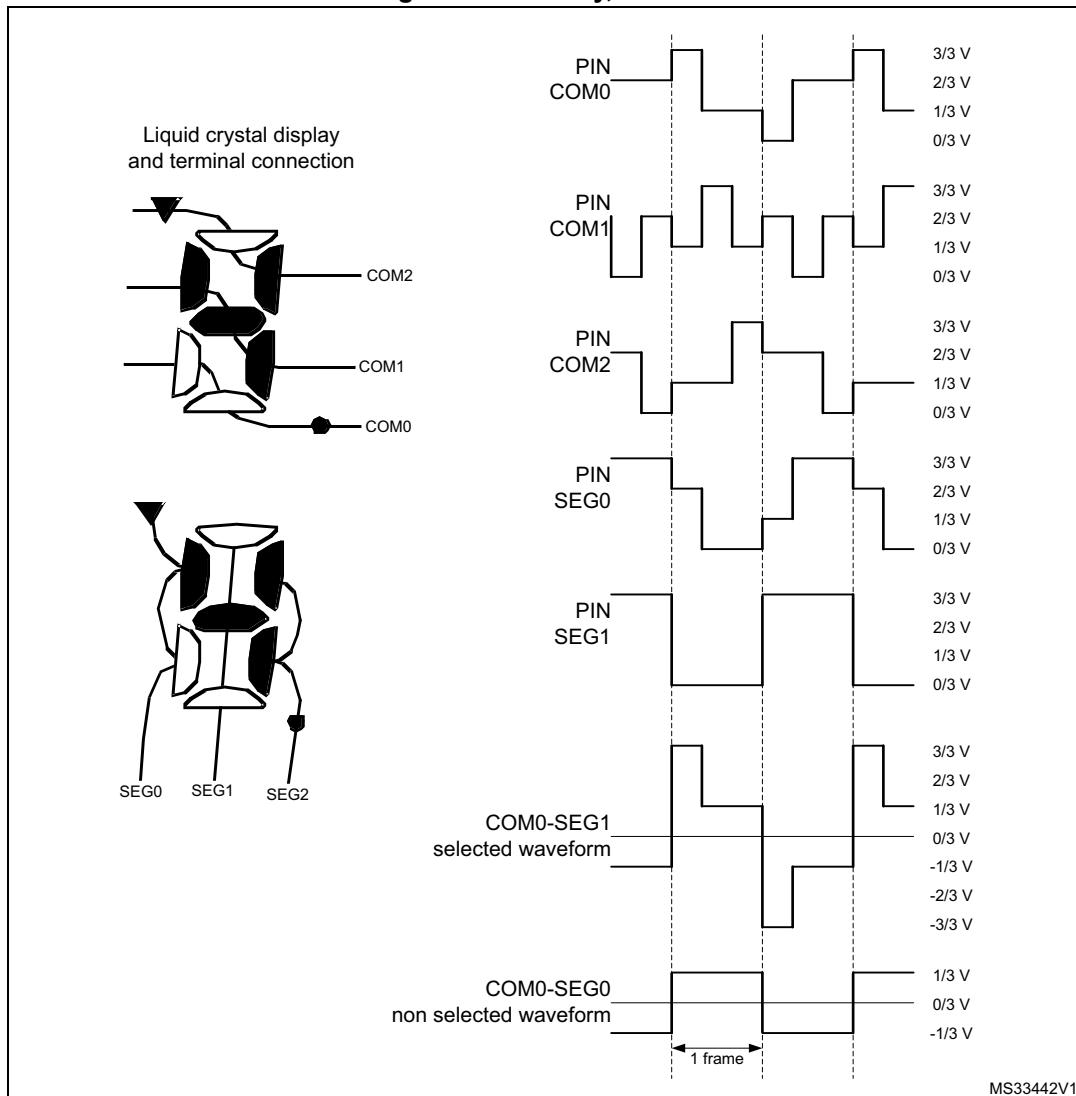


Figure 80. 1/4 duty, 1/3 bias

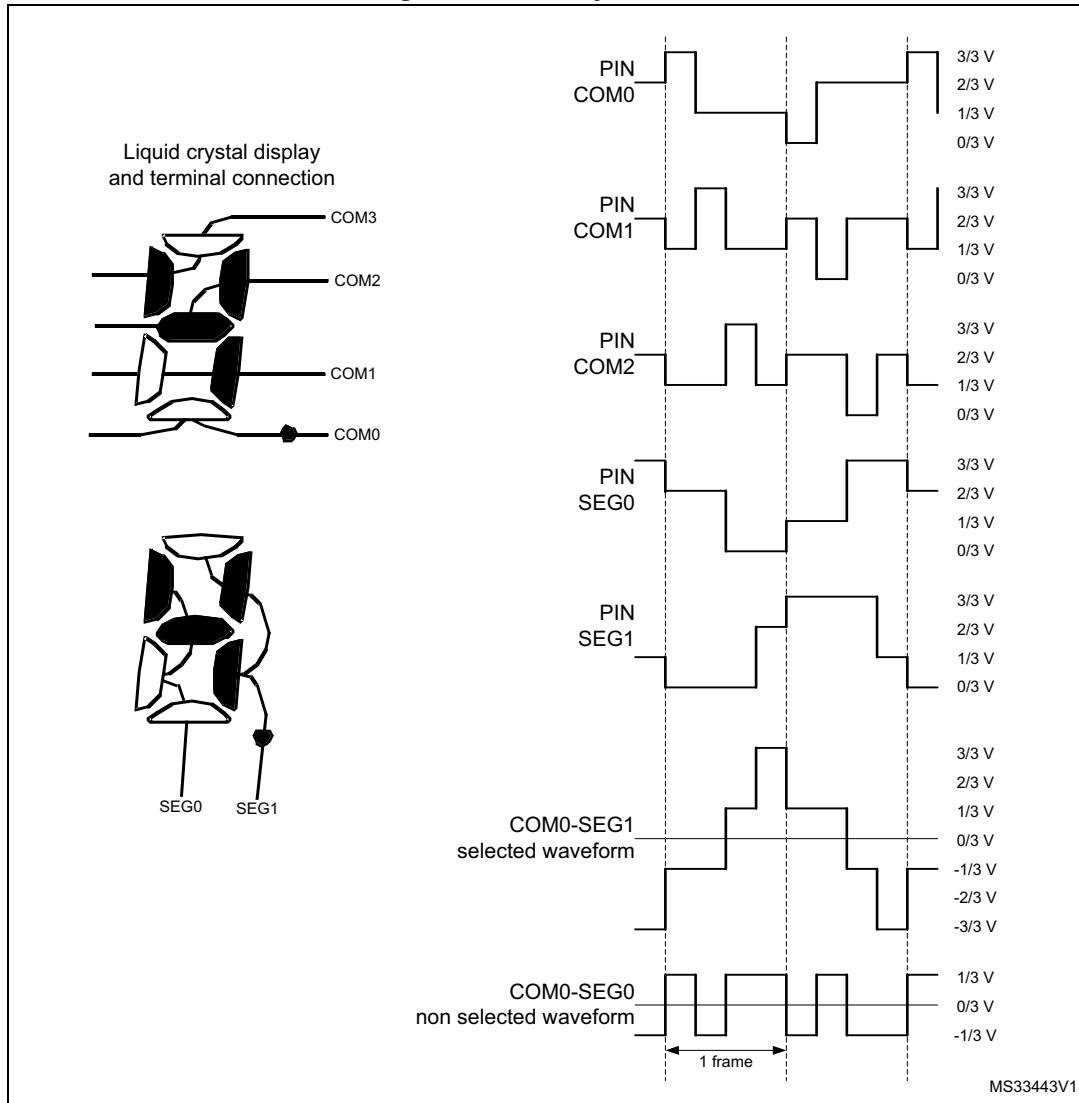
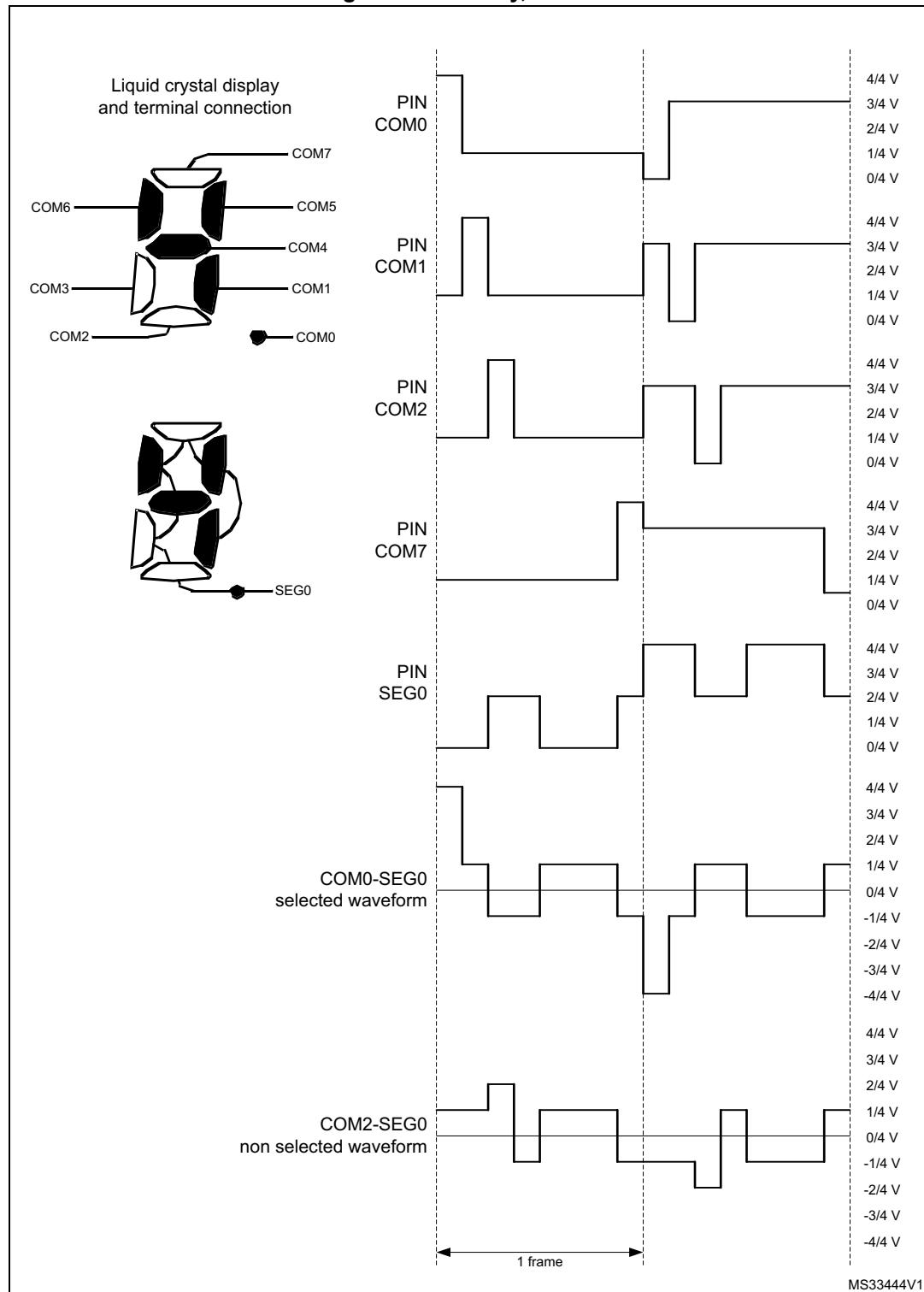


Figure 81. 1/8 duty, 1/4 bias



Blink

The segment driver also implements a programmable blink feature to allow some pixels to continuously switch on at a specific frequency. The blink mode can be configured by BLINK[1:0] in LCD_FCR, making possible to blink up to 1, 2, 4, 8, or all pixels (see [Section 19.6.2](#)). The blink frequency can be selected from eight different values using BLINKF[2:0] in LCD_FCR.

The table below gives examples of different blink frequencies (as a function of ck_div frequency).

Table 100. Blink frequency

| BLINKF[2:0] | | | ck_div frequency (with LCDCLK frequency of 32.768 kHz) | | | |
|-------------|---|---|--|---------|---------|---------|
| | | | 32 Hz | 64 Hz | 128 Hz | 256 Hz |
| 0 | 0 | 0 | 4.0 Hz | N/A | N/A | N/A |
| 0 | 0 | 1 | 2.0 Hz | 4.0 Hz | N/A | N/A |
| 0 | 1 | 0 | 1.0 Hz | 2.0 Hz | 4.0 Hz | N/A |
| 0 | 1 | 1 | 0.5 Hz | 1.0 Hz | 2.0 Hz | 4.0 Hz |
| 1 | 0 | 0 | 0.25 Hz | 0.5 Hz | 1.0 Hz | 2.0 Hz |
| 1 | 0 | 1 | N/A | 0.25 Hz | 0.5 Hz | 1.0 Hz |
| 1 | 1 | 0 | N/A | N/A | 0.25 Hz | 0.5 Hz |
| 1 | 1 | 1 | N/A | N/A | N/A | 0.25 Hz |

19.3.5 Voltage generator and contrast control

LCD supply source

The LCD power supply source comes from either the internal stepup converter, or from an external voltage applied on the VLCD pin. The internal or external voltage source can be selected using VSEL in LCD_CR. In case of external source selected, the internal boost circuit (stepup converter) is disabled to reduce power consumption.

When the stepup converter is selected as V_{LCD} source, the V_{LCD} value can be chosen among a wide set of values from V_{LCDmin} to V_{LCDmax} by means of CC[2:0] (contrast control) in LCD_FCR (see [Section 19.6.2](#)). New values of V_{LCD} takes effect every beginning of a new frame.

When external power source is selected as V_{LCD} source, the V_{LCD} voltage must be chosen in the range of V_{LCDmin} to V_{LCDmax} (see datasheets). The contrast can then be controlled by programming a dead time between frames (see [Deadtime](#)).

LCD intermediate voltages

The LCD intermediate voltage levels are generated through an internal resistor divider network as shown in [Figure 82](#).

The LCD voltage generator issues intermediate voltage levels between V_{SS} and V_{LCD} :

- 1/3 V_{LCD} and 2/3 V_{LCD} in case of 1/3 bias
- 1/4 V_{LCD} , 2/4 V_{LCD} and 3/4 V_{LCD} in case of 1/4 bias

- only 1/2 V_{LCD} in case of 1/2 bias.

LCD drive selection

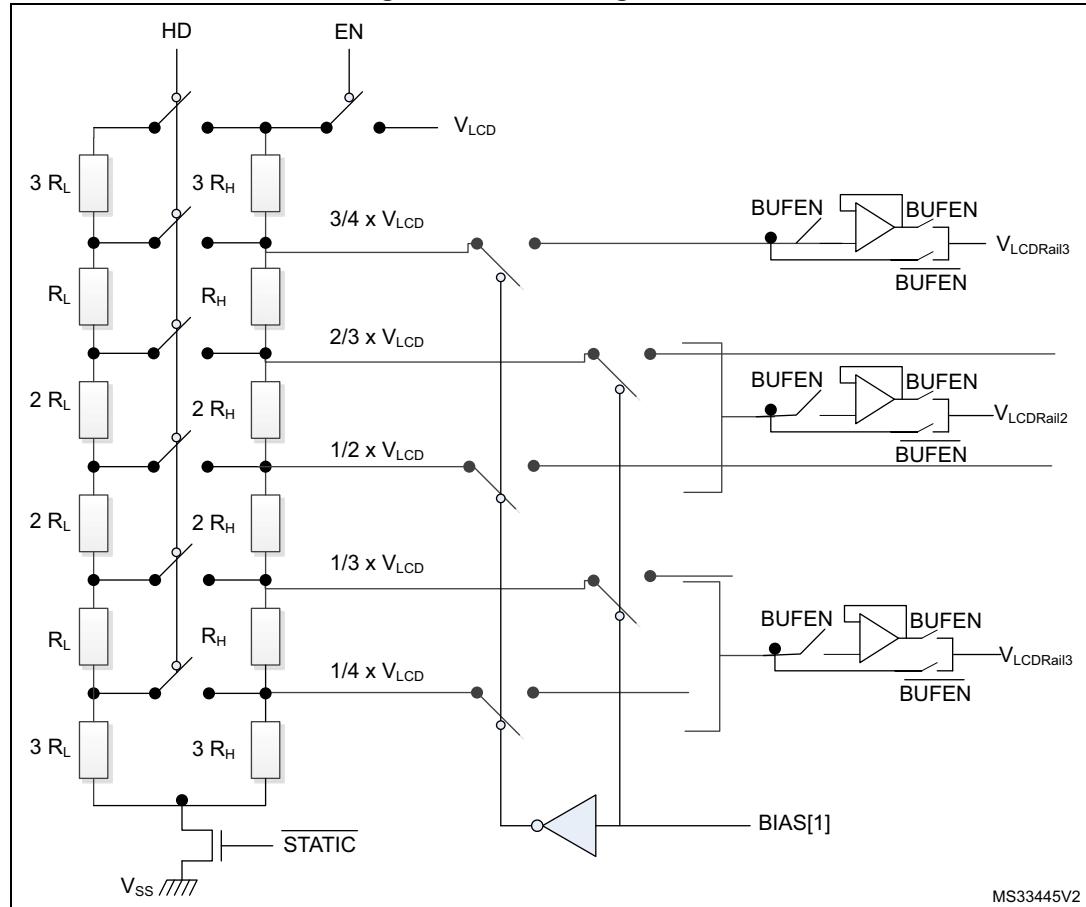
Two resistive networks, one with low-value resistors (R_L), and one with high-value resistors (R_H) are respectively used to increase the current during transitions, and to reduce power consumption in static state.

The EN switch follows the rules described below (see [Figure 82](#)):

- If LCDEN is set in LCD_CR, the EN switch is closed.
- When clearing LCDEN in LCD_CR, the EN switch is open at the end of the even frame in order to avoid a medium voltage level different from 0 considering the entire frame odd plus even.

PON[2:0] (pulse on duration) in LCD_FCR configures the time during which R_L is enabled through the HD (high drive) switch when the levels of the common and segment lines change (see [Figure 82](#)). A short drive time leads to lower power consumption, but displays with high internal resistance may need a longer drive time to achieve satisfactory contrast.

Figure 82. LCD voltage control



MS33445V2

1. R_{LN} and R_{HN} are the low value resistance network and the high value resistance network, respectively.

The R_{LN} divider can be always switched on using HD in LCD_FCR (see [Section 19.6.2](#)).

The HD switch follows the rules described below:

- If HD and PON[2:0] are reset in LCD_FCR, the HD switch is open.
- If HD is reset in LCD_FCR, and PON[2:0] in LCD_FCR are different from 00, the HD switch is closed during the number of pulses defined in PON[2:0].
- If HD = 1 in LCD_FCR, then HD switch is always closed.

After LCDEN is activated, RDY is set in LCD_SR to indicate that voltage levels are stable and the LCD controller can start to work.

Buffered mode

When voltage output buffers are enabled by setting BUFEN in LCD_CR, the LCD driving capability is improved as buffers prevent the LCD capacitive loads from loading the resistor bridge unacceptably and interfering with its voltage generation. As a result, intermediate voltage levels are more stable, which improves RMS voltage applied to the LCD pixels.

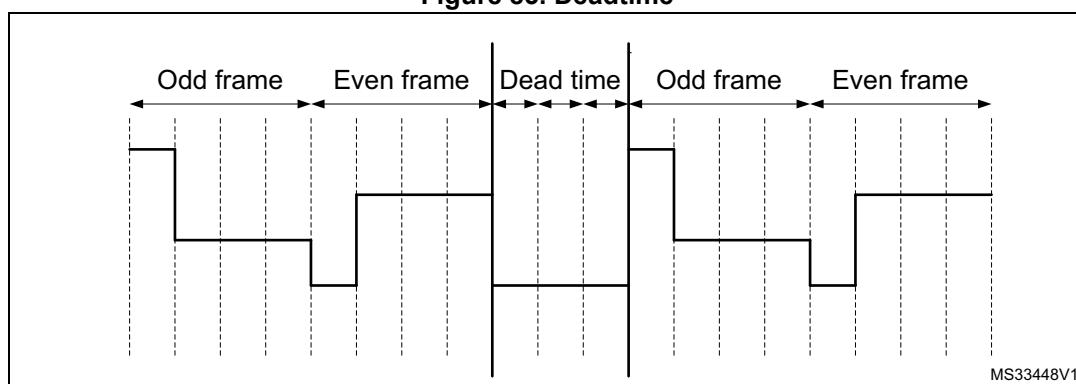
In buffer mode, intermediate voltages are generated by the high value resistor bridge R_{HN} to reduce power consumption. The low value resistor bridge R_{LN} is automatically disabled whatever HD or PON[2:0] configuration.

Buffers can be used independently of the V_{LCD} supply source (internal or external), but they can only be enabled or disabled when the LCD controller is not activated.

Deadtime

In addition to using CC[2:0], the contrast can be controlled by programming a dead time between each frame. During the dead time, the COM and SEG values are put to V_{SS} . DEAD[2:0] in LCD_FCR can be used to program a time of up to eight phase periods. This dead time reduces the contrast without modifying the frame rate.

Figure 83. Deadtime



19.3.6 Double-buffer memory

Using its double-buffer memory, the LCD controller ensures the coherency of the displayed information without having to use interrupts to control the LCD_RAM modification.

The application software can access the first buffer level (LCD_RAM) through the APB interface. Once it has modified the LCD_RAM, it sets UDR in LCD_SR. This UDR flag (update display request) requests the updated information to be moved into the second buffer level (LCD_DISPLAY).

This operation is done synchronously with the frame (at the beginning of the next frame). Until the update is completed, the LCD_RAM is write protected, and the UDR flag stays high. Once the update is completed, another flag (UDD, update display done) is set and generates an interrupt if UDDIE is set in LCD_FCR.

The time it takes to update LCD_DISPLAY is, in the worst case, one odd and one even frame.

The update does not occur (UDR = 1 and UDD = 0) until the display is enabled (LCDEN = 1)

19.3.7 COM and SEG multiplexing

Output pins versus duty modes

The output pins consists of:

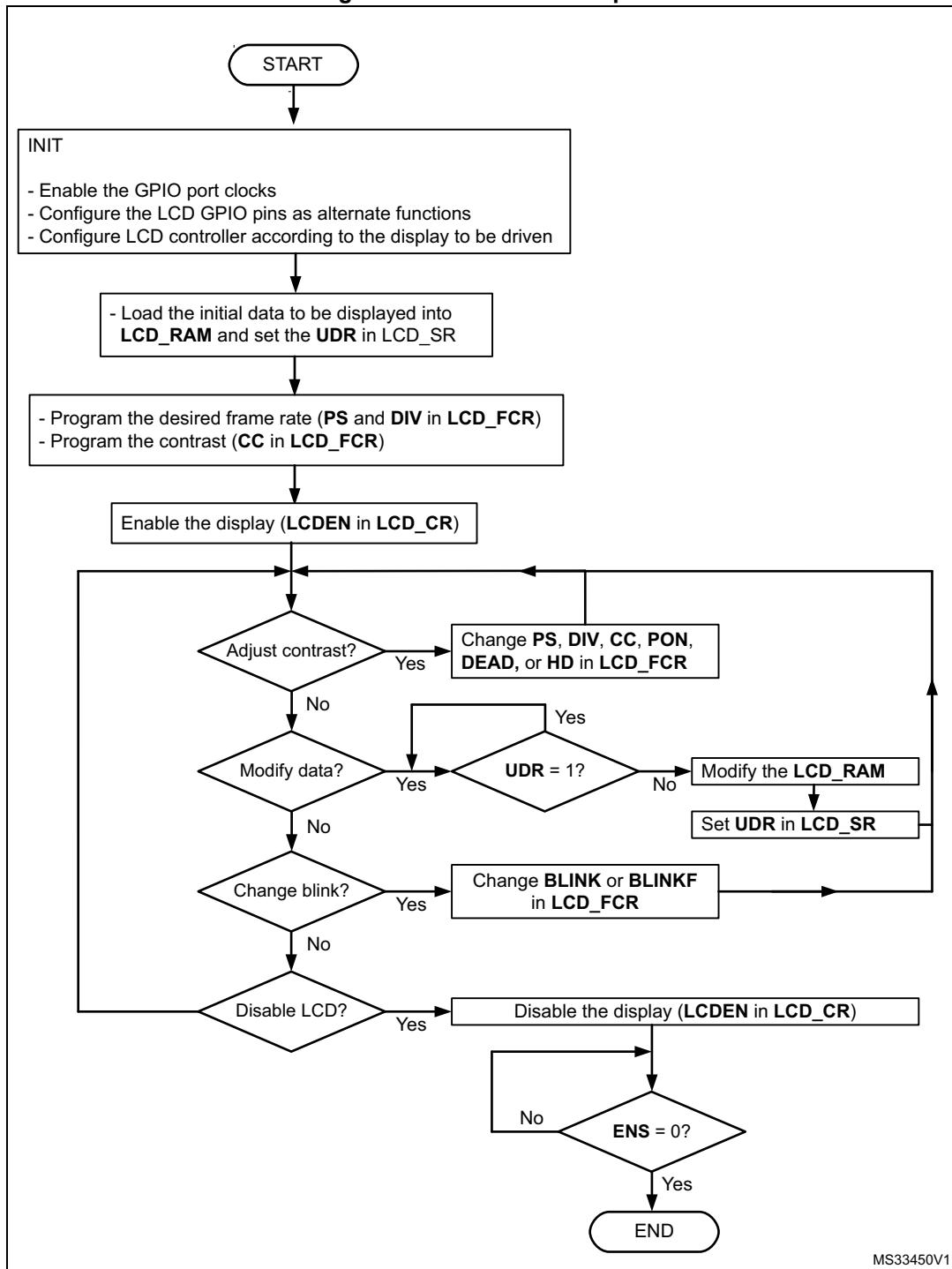
- SEG[51:0]
- COM[3:0]

Depending on the duty configuration, the COM and SEG output pins can have different functions:

- In static, 1/2, 1/3, and 1/4 duty modes, there are up to 52 SEG pins and respectively 1, 2, 3, and 4 COM pins
- In 1/8 duty mode (DUTY[2:0] = 100), COM[7:4] outputs are available on SEG[51:48]. This allows reducing the number of available segments.

19.3.8 Flowchart

Figure 84. Flowchart example



MS33450V1

19.4 LCD low-power modes

The LCD controller can be displayed in Stop mode, or can be fully disabled to reduce power consumption.

19.5 LCD interrupts

The table below gives the list of LCD interrupt requests.

Table 101. LCD interrupt requests

| Interrupt event | Event flag | Event flag Interrupt clearing method | Interrupt enable control bit |
|---------------------------|------------|---|---------------------------------|
| Start of frame (SOF) | SOF | Write SOFC = 1 | SOFIE |
| Update display done (UDD) | UDD | Write UDDC = 1 | UDDIE |

Start of frame (SOF)

The LCD start of frame interrupt is executed if SOFIE (start of frame interrupt enable) is set in LCD_FCR. SOF is cleared by writing SOFC to 1 in LCD_CLR when executing the corresponding interrupt handling vector.

Update display done (UDD)

The LCD update display interrupt is executed if UDDIE (update display done interrupt enable) is set in LCD_FCR. UDD is cleared by writing th UDDC to 1 in LCD_CLR when executing the corresponding interrupt handling vector.

Depending on the product implementation, all these interrupts events can either share the same interrupt vector (LCD global interrupt), or be grouped into two interrupt vectors (LCD SOF interrupt and LCD UDD interrupt). Refer to the for details.

To enable the LCD interrupts, the following sequence is required:

1. Configure and enable the LCD IRQ channel in the NVIC.
2. Configure the LCD to generate interrupts.

19.6 LCD registers

These registers are accessed by words (32-bit).

19.6.1 LCD control register (LCD_CR)

Address offset: 0x00

Reset value: 0x0000 0000

VSEL, MUX_SEG, BIAS, DUTY, and BUFEN bitfields are write-protected when the LCD is enabled
(ENS =1 in LCD_SR).

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

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

Bit 8 **BUFEN**: Voltage output buffer enable

This bit is used to enable/disable the voltage output buffer for higher driving capability.

0: Output buffer disabled

1: Output buffer enabled

Bit 7 **MUX_SEG**: Mux segment enable

This bit is used to enable SEG pin remapping. Four SEG pins can be multiplexed with SEG[31:28]. See [Section 19.3.7](#).

0: SEG pin multiplexing disabled

1: SEG[31:28] multiplexed with SEG[43:40]

Bits 6:5 **BIAS[1:0]**: Bias selector

These bits determine the bias used. Value 11 is forbidden.

00: Bias 1/4

01: Bias 1/2

10: Bias 1/3

11: Reserved

Bits 4:2 **DUTY[2:0]**: Duty selection

These bits determine the duty cycle. Values 101, 110 and 111 are forbidden.

000: Static duty

001: 1/2 duty

010: 1/3 duty

011: 1/4 duty

100: 1/8 duty

Others: Reserved

Bit 1 **VSEL**: Voltage source selection

This bit determines the voltage source for the LCD.

0: Internal source (voltage stepup converter)

1: External source (VLCD pin)

Bit 0 **LCDEN**: LCD controller enable

This bit is set by software to enable the LCD controller/driver. It is cleared by software to turn off the LCD at the beginning of the next frame. When the LCD is disabled, all COM and SEG pins are driven to V_{SS}.

0: LCD controller disabled
1: LCD controller enabled

19.6.2 LCD frame control register (LCD_FCR)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be updated any time, but the new values are applied only at the beginning of the next frame (except for UDDIE, SOFIE that affect the device behavior immediately). The new value of CC[2:0] is also applied immediately, but its effect on the device is delayed at the beginning of the next frame by the voltage generator.

Reading this register gives the last value written in the register, and not the configuration used to display the current frame.

When BUFEN is set in LCD_CR, the low resistor divider network is automatically disabled whatever HD or PON[2:0] configuration.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------------|------|------|---------|------|------|-----------|----|----|----------|----------|----|-------|------|------------|----|
| Res. | Res. | Res. | Res. | Res. | Res. | PS[3:0] | | | | DIV[3:0] | | | | BLINK[1:0] | |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BLINKF[2:0] | | | CC[2:0] | | | DEAD[2:0] | | | PON[2:0] | | | UDDIE | Res. | SOFIE | HD |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw |

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:22 **PS[3:0]**: PS 16-bit prescaler

These bits are written by software to define the division factor of the PS 16-bit prescaler.
 $ck_{ps} = LCDCLK/(2^{PS[3:0]})$. See [Section 19.3.2](#).

0000: ck_ps = LCDCLK
 0001: ck_ps = LCDCLK/2
 0002: ck_ps = LCDCLK/4
 ...
 1111: ck_ps = LCDCLK/32768

Bits 21:18 **DIV[3:0]**: DIV clock divider

These bits are written by software to define the division factor of the DIV divider
 (see [Section 19.3.2](#).)

0000: ck_div = ck_ps/16
 0001: ck_div = ck_ps/17
 0002: ck_div = ck_ps/18
 ...
 1111: ck_div = ck_ps/31

Bits 17:16 **BLINK[1:0]**: Blink mode selection

- 00: Blink disabled
- 01: Blink enabled on SEG[0], COM[0] (1 pixel)
- 10: Blink enabled on SEG[0], all COMs (up to 8 pixels depending on the programmed duty)
- 11: Blink enabled on all SEGs and all COMs (all pixels)

Bits 15:13 **BLINKF[2:0]**: Blink frequency selection

- 000: $f_{LCD}/8$
- 001: $f_{LCD}/16$
- 010: $f_{LCD}/32$
- 011: $f_{LCD}/64$
- 100: $f_{LCD}/128$
- 101: $f_{LCD}/256$
- 110: $f_{LCD}/512$
- 111: $f_{LCD}/1024$

Bits 12:10 **CC[2:0]**: Contrast control

These bits specify one of the V_{LCD} maximum voltages (independent of V_{DD}). It ranges from 2.60 V to 3.51V.

- 000: V_{LCD0}
- 001: V_{LCD1}
- 010: V_{LCD2}
- 011: V_{LCD3}
- 100: V_{LCD4}
- 101: V_{LCD5}
- 110: V_{LCD6}
- 111: V_{LCD7}

Note: Refer to the datasheet for the V_{LCDx} values.

Bits 9:7 **DEAD[2:0]**: Dead time duration

These bits are written by software to configure the length of the dead time between frames. During the dead time the COM and SEG voltage levels are held at 0 V to reduce the contrast without modifying the frame rate.

- 000: No dead time
- 001: 1 phase period dead time
- 010: 2 phase period dead time
-
- 111: 7 phase period dead time

Bits 6:4 PON[2:0]: Pulse ON duration

These bits are written by software to define the pulse duration in terms of ck_ps pulses. A short pulse leads to lower power consumption, but displays with high internal resistance may need a longer pulse to achieve satisfactory contrast.

Note that the pulse is never longer than one half prescaled LCD clock period.

000: 0

001: 1/ck_ps

010: 2/ck_ps

011: 3/ck_ps

100: 4/ck_ps

101: 5/ck_ps

110: 6/ck_ps

111: 7/ck_ps

PON duration example with LCDCLK = 32.768 kHz and PS=0x03:

000: 0 µs

001: 244 µs

010: 488 µs

011: 782 µs

100: 976 µs

101: 1.22 ms

110: 1.46 ms

111: 1.71 ms

Bit 3 UDDIE: Update display done interrupt enable

This bit is set and cleared by software.

0: LCD update display done interrupt disabled

1: LCD update display done interrupt enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 SOFIE: Start of frame interrupt enable

This bit is set and cleared by software.

0: LCD start-of-frame interrupt disabled

1: LCD start-of-frame interrupt enabled

Bit 0 HD: High drive enable

This bit is written by software to enable a low resistance divider. Displays with high internal resistance may need a longer drive time to achieve satisfactory contrast. This bit is useful in this case if some additional power consumption can be tolerated.

0: Permanent high drive disabled

1: Permanent high drive enabled. When HD = 1, PON[2:0] must be programmed to 001.

19.6.3 LCD status register (LCD_SR)

Address offset: 0x08

Reset value: 0x0000 0020

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|-------|------|------|------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | FCRSF | RDY | UDD | UDR | SOF | ENS |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 FCRSF: LCD frame control register synchronization flag

This bit is set by hardware each time the LCD_FCR register is updated in the LCDCLK domain. It is cleared by hardware when writing to the LCD_FCR register.

0: LCD frame control register not yet synchronized

1: LCD frame control register synchronized

Bit 4 RDY: Ready flag

This bit is set and cleared by hardware. It indicates the status of the stepup converter.

0: Not ready

1: Stepup converter enabled and ready to provide the correct voltage

Bit 3 UDD: Update display done

This bit is set by hardware. It is cleared by writing 1 to UDDC in LCD_CLR. The bit set has priority over the clear.

0: No event

1: Update display request done. A UDD interrupt is generated if UDDIE = 1 in LCD_FCR.

Note: If the device is in Stop mode (PCLK not provided), UDD does not generate an interrupt even if UDDIE = 1. If the display is not enabled, the UDD interrupt never occurs.

Bit 2 UDR: Update display request

Each time software modifies the LCD_RAM, it must set this bit to transfer the updated data to the second level buffer. This bit stays set until the end of the update. During this time, the LCD_RAM is write protected.

0: No effect

1: Update display request

Note: When the display is disabled, the update is performed for all LCD_DISPLAY locations.

When the display is enabled, the update is performed only for locations for which commons are active (depending on DUTY). For example if DUTY = 1/2, only the LCD_DISPLAY of COM0 and COM1 are updated.

Writing 0 on this bit or writing 1 when it is already 1 has no effect. This bit can be cleared by hardware only. It can be cleared only when LCDEN = 1

Bit 1 SOF: Start-of-frame flag

This bit is set by hardware at the beginning of a new frame, at the same time as the display data is updated. It is cleared by writing a 1 to SOFC in LCD_CLR. The bit clear has priority over the set.

0: No event

1: Start-of-frame event occurred. An LCD SOF interrupt is generated if SOFIE is set.

Bit 0 ENS: LCD enabled status

This bit is set and cleared by hardware. It indicates the LCD controller status.

0: LCD controller disabled

1: LCD controller enabled

Note: This bit is set immediately when LCDEN in LCD_CR goes from 0 to 1. On deactivation, it reflects the real LCD status. It becomes 0 at the end of the last displayed frame.

19.6.4 LCD clear register (LCD_CLR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit 3 **UDDC**: Update display done clear

This bit is written by software to clear UDD in LCD_SR.

0: No effect

1: Clear UDD flag.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SOFC**: Start-of-frame flag clear

This bit is written by software to clear SOF in LCD_SR.

0: No effect

1: Clear SOF flag.

Bit 0 Reserved, must be kept at reset value.

19.6.5 LCD display memory (LCD_RAMx)

Address offset: 0x14 + 0x4 * x, (x = 0, 2, 4, 6, 8, 10, 12, 14)

Reset value: 0x0000 0000

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

Bits 31:0 **SEGMENT_DATA[31:0]**:

Each bit corresponds to one pixel of the LCD display.

0: Pixel inactive

1: Pixel active

19.6.6 LCD display memory (LCD_RAMx)

Address offset: $0x14 + 0x4 * x$, ($x = 1, 3, 5, 7$)

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. | SEGMENT_DATA[51:48] | | | |
| | | | | | | | | | | | | | | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| SEGMENT_DATA[47:32] | | | | | | | | | | | | | | | | | |
| 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 **SEGMENT_DATA[51:32]**:

Each bit corresponds to one pixel of the LCD display.

0: Pixel inactive

1: Pixel active

19.6.7 LCD display memory (LCD_RAMx)

Address offset: $0x14 + 0x4 * x$, ($x = 9, 11, 13, 15$)

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SEGMENT_DATA[47:32] | | | | | | | | | | | | | | | |
| 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 **SEGMENT_DATA[47:32]**:

Each bit corresponds to one pixel of the LCD display.

0: Pixel inactive

1: Pixel active

19.6.8 LCD register map

Table 102. LCD register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------|---------------|------|------|------|------|------|------|------|------|------|------|------|------|------------|---------|----|----|
| 0x00 | LCD_CR | Res. | BUFEN | 8 | 0 |
| | Reset value | | | | | | | | | | | | | | MUX_SEG | 7 | 0 |
| | | | | | | | | | | | | | | BIAS[1:0] | 6 | 0 | |
| | | | | | | | | | | | | | | DUTY [2:0] | 5 | 0 | |
| | | | | | | | | | | | | | | VSEL | 3 | 0 | |
| | | | | | | | | | | | | | | LCDEN | 2 | 0 | |

Table 102. LCD register map and reset values (continued)

Table 102. LCD register map and reset values (continued)

| Offset | Register name | 0x3C | 0x40 | 0x44 | 0x48 | 0x4C | 0x50 |
|--------|----------------|------|------|------|------|------|------|
| | LCD_RAM (COM5) | | | | | | |
| | LCD_RAM (COM6) | | | | | | |
| | LCD_RAM (COM7) | | | | | | |
| Res. | S31 | Res. | 0 | S31 | Res. | 0 | S31 |
| Res. | S30 | Res. | 0 | S30 | Res. | 0 | S30 |
| Res. | S29 | Res. | 0 | S29 | Res. | 0 | S29 |
| Res. | S28 | Res. | 0 | S28 | Res. | 0 | S28 |
| Res. | S27 | Res. | 0 | S27 | Res. | 0 | S27 |
| Res. | S26 | Res. | 0 | S26 | Res. | 0 | S26 |
| Res. | S25 | Res. | 0 | S25 | Res. | 0 | S25 |
| Res. | S24 | Res. | 0 | S24 | Res. | 0 | S24 |
| Res. | S23 | Res. | 0 | S23 | Res. | 0 | S23 |
| Res. | S22 | Res. | 0 | S22 | Res. | 0 | S22 |
| Res. | S21 | Res. | 0 | S21 | Res. | 0 | S21 |
| Res. | S20 | Res. | 0 | S20 | Res. | 0 | S20 |
| Res. | S19 | Res. | 0 | S19 | Res. | 0 | S19 |
| Res. | S18 | Res. | 0 | S18 | Res. | 0 | S18 |
| Res. | S17 | Res. | 0 | S17 | Res. | 0 | S17 |
| Res. | S16 | Res. | 0 | S16 | Res. | 0 | S16 |
| 0 | S47 | 0 | S15 | 0 | S47 | 0 | S15 |
| 0 | S46 | 0 | S14 | 0 | S46 | 0 | S14 |
| 0 | S45 | 0 | S13 | 0 | S45 | 0 | S13 |
| 0 | S44 | 0 | S12 | 0 | S44 | 0 | S12 |
| 0 | S43 | 0 | S11 | 0 | S43 | 0 | S11 |
| 0 | S42 | 0 | S10 | 0 | S42 | 0 | S10 |
| 0 | S41 | 0 | S09 | 0 | S41 | 0 | S09 |
| 0 | S40 | 0 | S08 | 0 | S40 | 0 | S08 |
| 0 | S39 | 0 | S07 | 0 | S39 | 0 | S07 |
| 0 | S38 | 0 | S06 | 0 | S38 | 0 | S06 |
| 0 | S37 | 0 | S05 | 0 | S37 | 0 | S05 |
| 0 | S36 | 0 | S04 | 0 | S36 | 0 | S04 |
| 0 | S35 | 0 | S03 | 0 | S35 | 0 | S03 |
| 0 | S34 | 0 | S02 | 0 | S34 | 0 | S02 |
| 0 | S33 | 0 | S01 | 0 | S33 | 0 | S01 |
| 0 | S32 | 0 | S00 | 0 | S32 | 0 | S00 |

Refer to [Section 2.2](#) for the register boundary addresses table.

20 Touch sensing controller (TSC)

20.1 Introduction

The touch sensing controller provides a simple solution for adding capacitive sensing functionality to any application. Capacitive sensing technology is able to detect finger presence near an electrode that is protected from direct touch by a dielectric (for example glass, plastic). The capacitive variation introduced by the finger (or any conductive object) is measured using a proven implementation based on a surface charge transfer acquisition principle.

The touch sensing controller is fully supported by the STMtouch touch sensing firmware library, which is free to use and allows touch sensing functionality to be implemented reliably in the end application.

20.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 21 capacitive sensing channels
- Up to 7 capacitive sensing channels can be acquired in parallel offering a very good response time
- Spread spectrum feature to improve system robustness in noisy environments
- Full hardware management of the charge transfer acquisition sequence
- Programmable charge transfer frequency
- Programmable sampling capacitor I/O pin
- Programmable channel I/O pin
- Programmable max count value to avoid long acquisition when a channel is faulty
- Dedicated end of acquisition and max count error flags with interrupt capability
- One sampling capacitor for up to 3 capacitive sensing channels to reduce the system components
- Compatible with proximity, touchkey, linear and rotary touch sensor implementation
- Designed to operate with STMtouch touch sensing firmware library

Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to I/O availability.

20.3 TSC implementation

Table 103. TSC implementation

| TSC features | STM32U031xx | STM32U0x3xx |
|-----------------------------|------------------|-------------|
| Number of analog I/O groups | 6 ⁽¹⁾ | 7 |
| Number of channels | 18 | 21 |

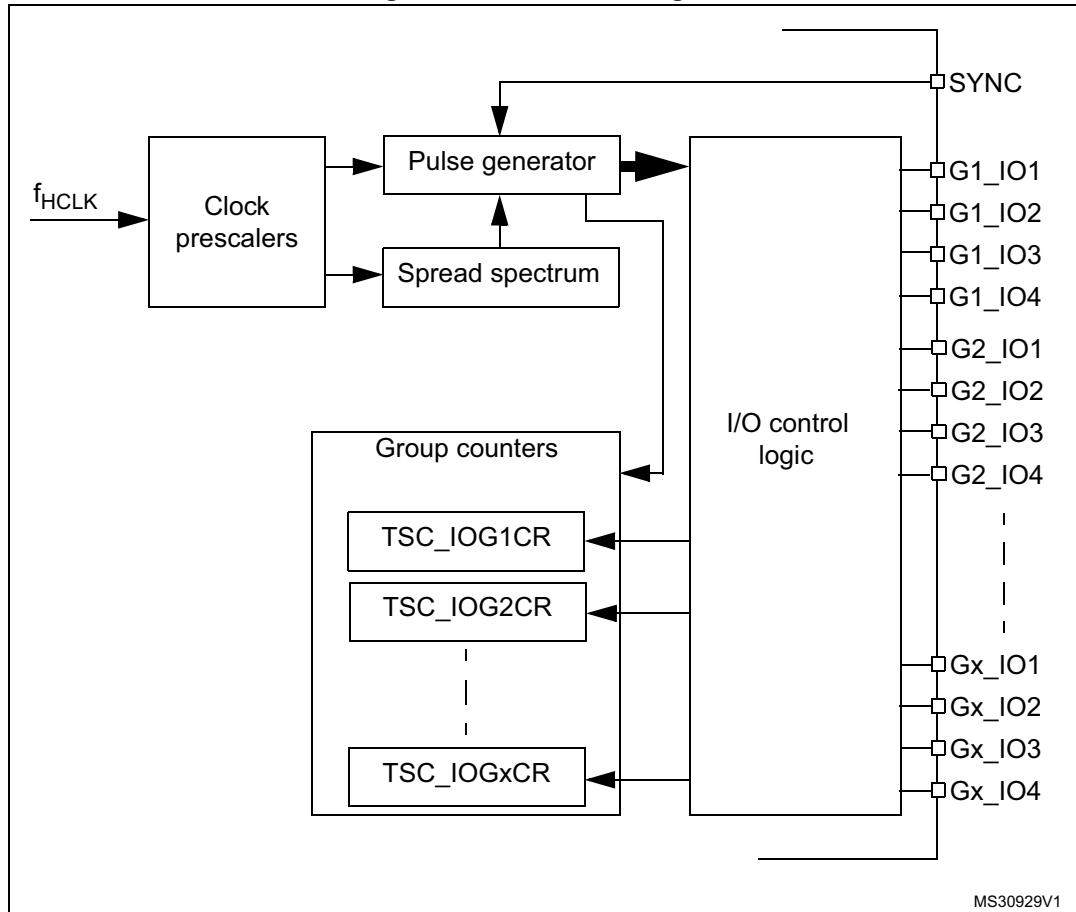
1. Only Gx, x = 1 to 6 applies for this category of devices. G7 related information does not apply and corresponding bits are reserved.

20.4 TSC functional description

20.4.1 TSC block diagram

The block diagram of the touch sensing controller is shown in [Figure 85](#).

Figure 85. TSC block diagram



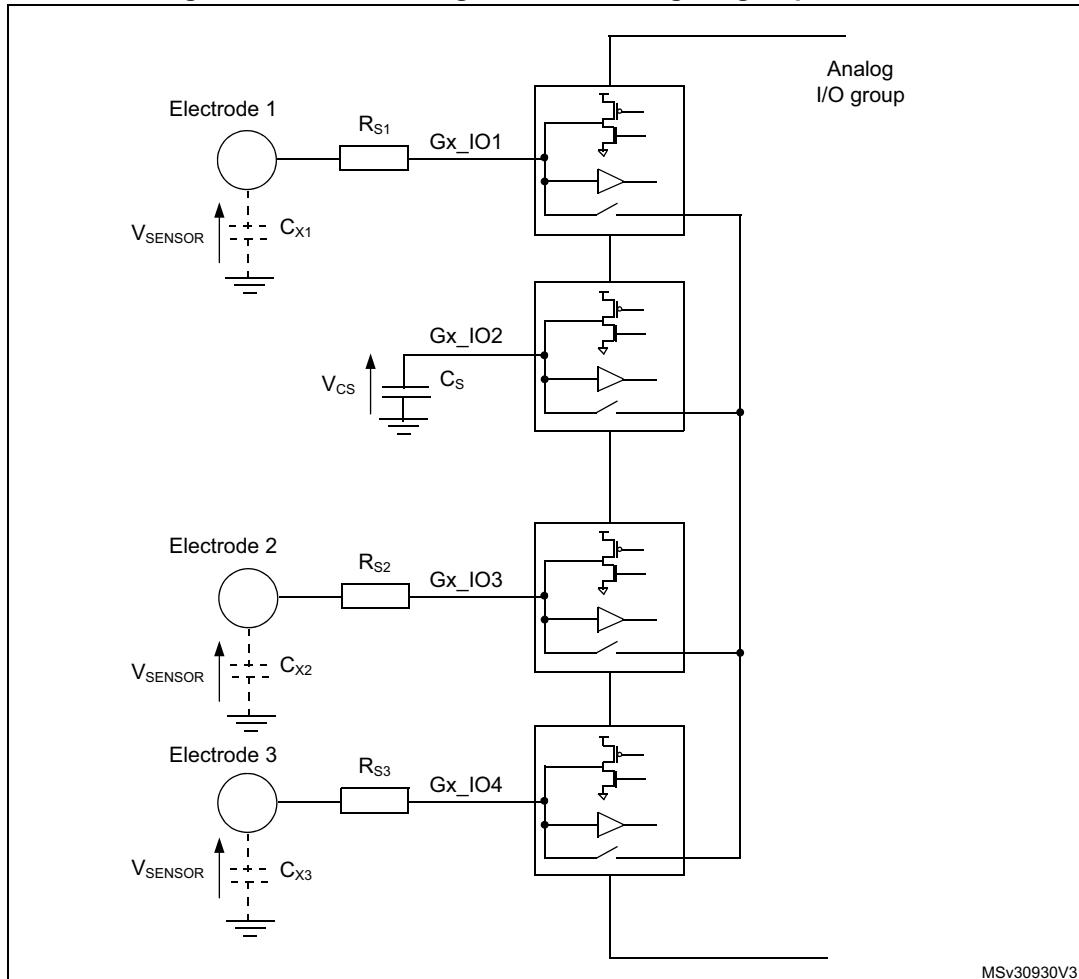
20.4.2 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum number of external components to operate with a single ended electrode type. This acquisition is designed around an analog I/O group composed of up to four GPIOs (see [Figure 86](#)). Several analog I/O groups are available to allow the acquisition of several capacitive sensing channels simultaneously and to support a larger number of capacitive sensing channels. Within a same analog I/O group, the acquisition of the capacitive sensing channels is sequential.

One of the GPIOs is dedicated to the sampling capacitor C_S . Only one sampling capacitor I/O per analog I/O group must be enabled at a time.

The remaining GPIOs are dedicated to the electrodes and are commonly called channels. For some specific needs (such as proximity detection), it is possible to simultaneously enable more than one channel per analog I/O group.

Figure 86. Surface charge transfer analog I/O group structure



Note: *Gx_x_IOy where x is the analog I/O group number and y the GPIO number within the selected group.*

The surface charge transfer acquisition principle consists of charging an electrode capacitance (C_X) and transferring a part of the accumulated charge into a sampling capacitor (C_S). This sequence is repeated until the voltage across C_S reaches a given threshold (V_{IH} in our case). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

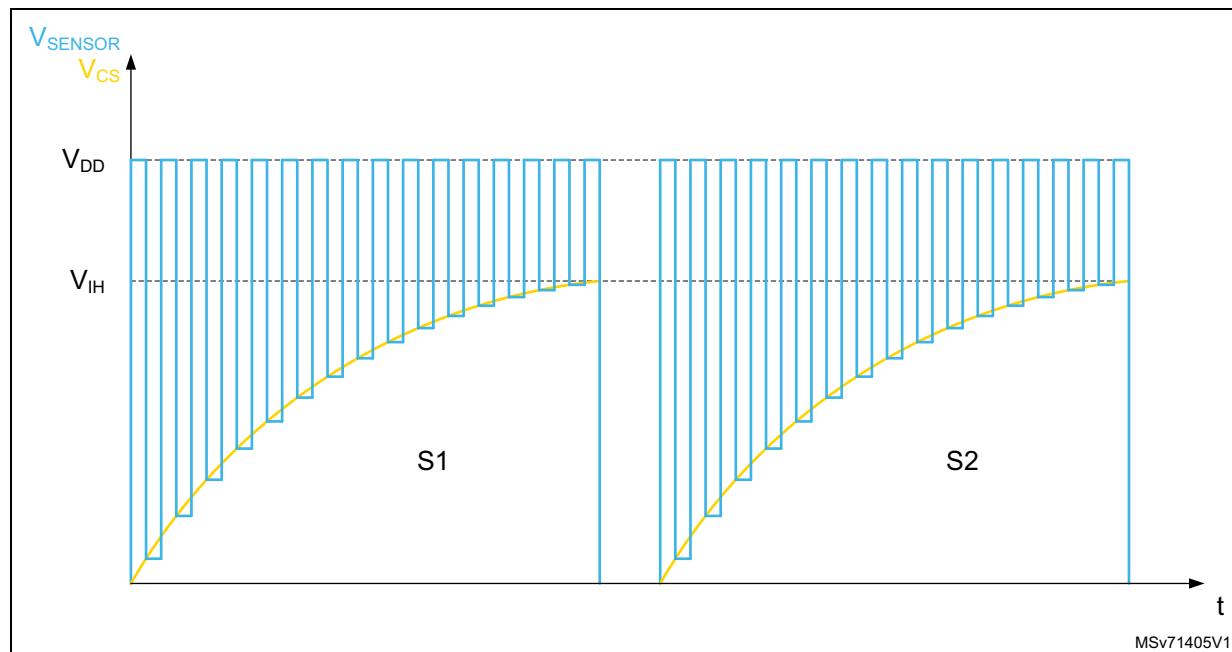
[Table 104](#) details the charge transfer acquisition sequence of the capacitive sensing channel 1. States 3 to 7 are repeated until the voltage across C_S reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor R_S improves the ESD immunity of the solution.

Table 104. Acquisition sequence summary

| State | Gx_IO1 (channel) | Gx_IO2 (sampling) | Gx_IO3 (channel) | Gx_IO4 (channel) | State description | | |
|-------|--|---|--|---------------------|--|--|--|
| #1 | Input floating with analog switch closed | Output open-drain low with analog switch closed | Input floating with analog switch closed | | Discharge all C_X and C_S | | |
| #2 | Input floating | | | | Dead time | | |
| #3 | Output push-pull high | Input floating | | | Charge C_{X1} | | |
| #4 | Input floating | | | | Dead time | | |
| #5 | Input floating with analog switch closed | Input floating | | | Charge transfer from C_{X1} to C_S | | |
| #6 | Input floating | | | | Dead time | | |
| #7 | Input floating | | | | Measure C_S voltage | | |

Note: Gx_IOy where x is the analog I/O group number and y the GPIO number within the selected group.

The voltage variation over the time on the sampling capacitor C_S is detailed below (refer to [Figure 86](#) for V_{SENSOR} and V_{CS} definition):

Figure 87. Sampling capacitor voltage variation

20.4.3 Reset and clocks

The TSC clock source is the AHB clock (HCLK). Two programmable prescalers are used to generate the pulse generator and the spread spectrum internal clocks:

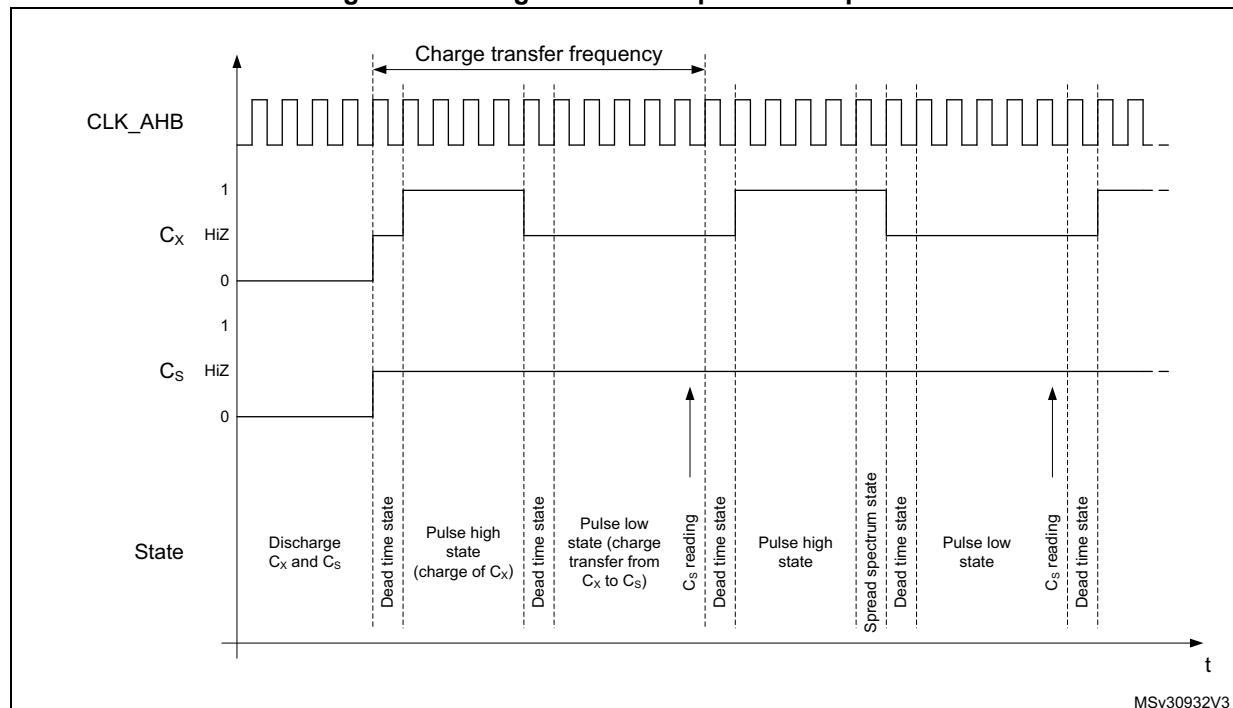
- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC_CR register

The reset and clock controller (RCC) provides dedicated bits to enable the touch sensing controller clock and to reset this peripheral. For more information, refer to [Section 5: Reset and clock control \(RCC\)](#).

20.4.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in [Figure 88](#).

Figure 88. Charge transfer acquisition sequence



MSv30932V3

For higher flexibility, the charge transfer frequency is fully configurable. Both the pulse high state (charge of C_X) and the pulse low state (transfer of charge from C_X to C_S) duration can be defined using the CTPH[3:0] and CTPL[3:0] bits in the TSC_CR register. The standard range for the pulse high and low states duration is 500 ns to 2 μ s. To ensure a correct measurement of the electrode capacitance, the pulse high state duration must be set to ensure that C_X is always fully charged.

A dead time where both the sampling capacitor I/O and the channel I/O are in input floating state is inserted between the pulse high and low states to ensure an optimum charge transfer acquisition sequence. This state duration is 1 period of HCLK.

At the end of the pulse high state and if the spread spectrum feature is enabled, a variable number of periods of the SSCLK clock are added.

The reading of the sampling capacitor I/O, to determine if the voltage across C_S has reached the given threshold, is performed at the end of the pulse low state.

Note: The following TSC control register configurations are forbidden:

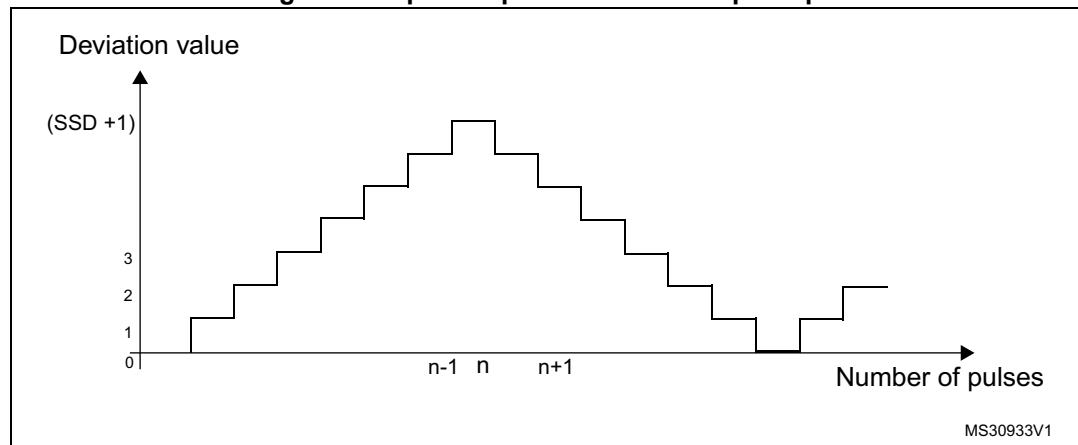
- bits PGPSC are set to 000 and bits CTPL are set to 0001
- bits PGPSC are set to 001 and bits CTPL are set to 0000

20.4.5 Spread spectrum feature

The spread spectrum feature generates a variation of the charge transfer frequency. This is done to improve the robustness of the charge transfer acquisition in noisy environments and also to reduce the induced emission. The maximum frequency variation is in the range of 10% to 50% of the nominal charge transfer period. For instance, for a nominal charge transfer frequency of 250 kHz (4 μ s), the typical spread spectrum deviation is 10% (400 ns) which leads to a minimum charge transfer frequency of ~227 kHz.

In practice, the spread spectrum consists of adding a variable number of SSCLK periods to the pulse high state using the principle shown below:

Figure 89. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

Table 105. Spread spectrum deviation versus AHB clock frequency

| f_{HCLK} | Spread spectrum step | Maximum spread spectrum deviation |
|------------|----------------------|-----------------------------------|
| 48 MHz | 20.8 ns | 5333.3 ns |

The spread spectrum feature can be disabled/enabled using the SSE bit in the TSC_CR register. The frequency deviation is also configurable to accommodate the device HCLK clock frequency and the selected charge transfer frequency through the SSPSC and SSD[6:0] bits in the TSC_CR register.

20.4.6 Max count error

The max count error prevents long acquisition times resulting from a faulty capacitive sensing channel. It consists of specifying a maximum count value for the analog I/O group counters. This maximum count value is specified using the MCV[2:0] bits in the TSC_CR register. As soon as an acquisition group counter reaches this maximum value, the ongoing acquisition is stopped and the end of acquisition (EOAF bit) and max count error (MCEF bit) flags are both set. An interrupt can also be generated if the corresponding end of acquisition (EOAIE bit) or/and max count error (MCEIE bit) interrupt enable bits are set.

20.4.7 Sampling capacitor I/O and channel I/O mode selection

To allow the GPIOs to be controlled by the touch sensing controller, the corresponding alternate function must be enabled through the standard GPIO registers and the GPIOxAFR registers.

The GPIOs modes controlled by the TSC are defined using the TSC_IOSCR and TSC_IOCCR register.

When there is no ongoing acquisition, all the I/Os controlled by the touch sensing controller are in default state. While an acquisition is ongoing, only unused I/Os (neither defined as sampling capacitor I/O nor as channel I/O) are in default state. The IODEF bit in the TSC_CR register defines the configuration of the I/Os which are in default state. The table below summarizes the configuration of the I/O depending on its mode.

Table 106. I/O state depending on its mode and IODEF bit value

| IODEF bit | Acquisition status | Unused I/O mode | Channel I/O mode | Sampling capacitor I/O mode |
|-----------------------------|--------------------|----------------------|----------------------|-----------------------------|
| 0 (output push-pull low) | No | Output push-pull low | Output push-pull low | Output push-pull low |
| 0 (output push-pull low) | Ongoing | Output push-pull low | - | - |
| 1 (input floating) | No | Input floating | Input floating | Input floating |
| 1 (input floating) | Ongoing | Input floating | - | - |

Unused I/O mode

An unused I/O corresponds to a GPIO controlled by the TSC peripheral but not defined as an electrode I/O nor as a sampling capacitor I/O.

Sampling capacitor I/O mode

To allow the control of the sampling capacitor I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output open drain mode and then the corresponding Gx_IoY bit in the TSC_IOSCR register must be set.

Only one sampling capacitor per analog I/O group must be enabled at a time.

Channel I/O mode

To allow the control of the channel I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output push-pull mode and the corresponding Gx_IOy bit in the TSC_IOCCR register must be set.

For proximity detection where a higher equivalent electrode surface is required or to speed-up the acquisition process, it is possible to enable and simultaneously acquire several channels belonging to the same analog I/O group.

Note: *During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR or TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.*

20.4.8 Acquisition mode

The touch sensing controller offers two acquisition modes:

- Normal acquisition mode: the acquisition starts as soon as the START bit in the TSC_CR register is set.
- Synchronized acquisition mode: the acquisition is enabled by setting the START bit in the TSC_CR register but only starts upon the detection of a falling edge or a rising edge and high level on the SYNC input pin. This mode is useful for synchronizing the capacitive sensing channels acquisition with an external signal without additional CPU load.

The GxE bits in the TSC_IOGCSR registers specify which analog I/O groups are enabled (corresponding counter is counting). The C_S voltage of a disabled analog I/O group is not monitored and this group does not participate in the triggering of the end of acquisition flag. However, if the disabled analog I/O group contains some channels, they are pulsed.

When the C_S voltage of an enabled analog I/O group reaches the given threshold, the corresponding GxS bit of the TSC_IOGCSR register is set. When the acquisition of all enabled analog I/O groups is complete (all GxS bits of all enabled analog I/O groups are set), the EOAF flag in the TSC_ISR register is set. An interrupt request is generated if the EOAIE bit in the TSC_IER register is set.

In the case that a max count error is detected, the ongoing acquisition is stopped and both the EOAF and MCEF flags in the TSC_ISR register are set. Interrupt requests can be generated for both events if the corresponding bits (EOAIE and MCEIE bits of the TSCIER register) are set. Note that when the max count error is detected the remaining GxS bits in the enabled analog I/O groups are not set.

To clear the interrupt flags, the corresponding EOAIIC and MCEIIC bits in the TSC_ICR register must be set.

The analog I/O group counters are cleared when a new acquisition is started. They are updated with the number of charge transfer cycles generated on the corresponding channel(s) upon the completion of the acquisition.

20.4.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller is able to take the control of the Schmitt trigger hysteresis and analog switch of each Gx_IOy. This control is available whatever the I/O control mode is (controlled by standard GPIO registers or other peripherals) assuming that the touch sensing controller is enabled. This may be useful to perform a different acquisition sequence or for other purposes.

In order to improve the system immunity, the Schmitt trigger hysteresis of the GPIOs controlled by the TSC must be disabled by resetting the corresponding Gx_IOy bit in the TSC_IHCR register.

20.5 TSC low-power modes

Table 107. Effect of low-power modes on TSC

| Mode | Description |
|-----------------|--|
| Sleep | No effect. Peripheral interrupts cause the device to exit Sleep mode. |
| Low power run | No effect. |
| Low power sleep | No effect. Peripheral interrupts cause the device to exit Low-power sleep mode. |
| Stop 0 / Stop 1 | Peripheral registers content is kept. |
| Stop 2 | |
| Standby | Powered-down. The peripheral must be reinitialized after exiting Standby or Shutdown mode. |
| Shutdown | |

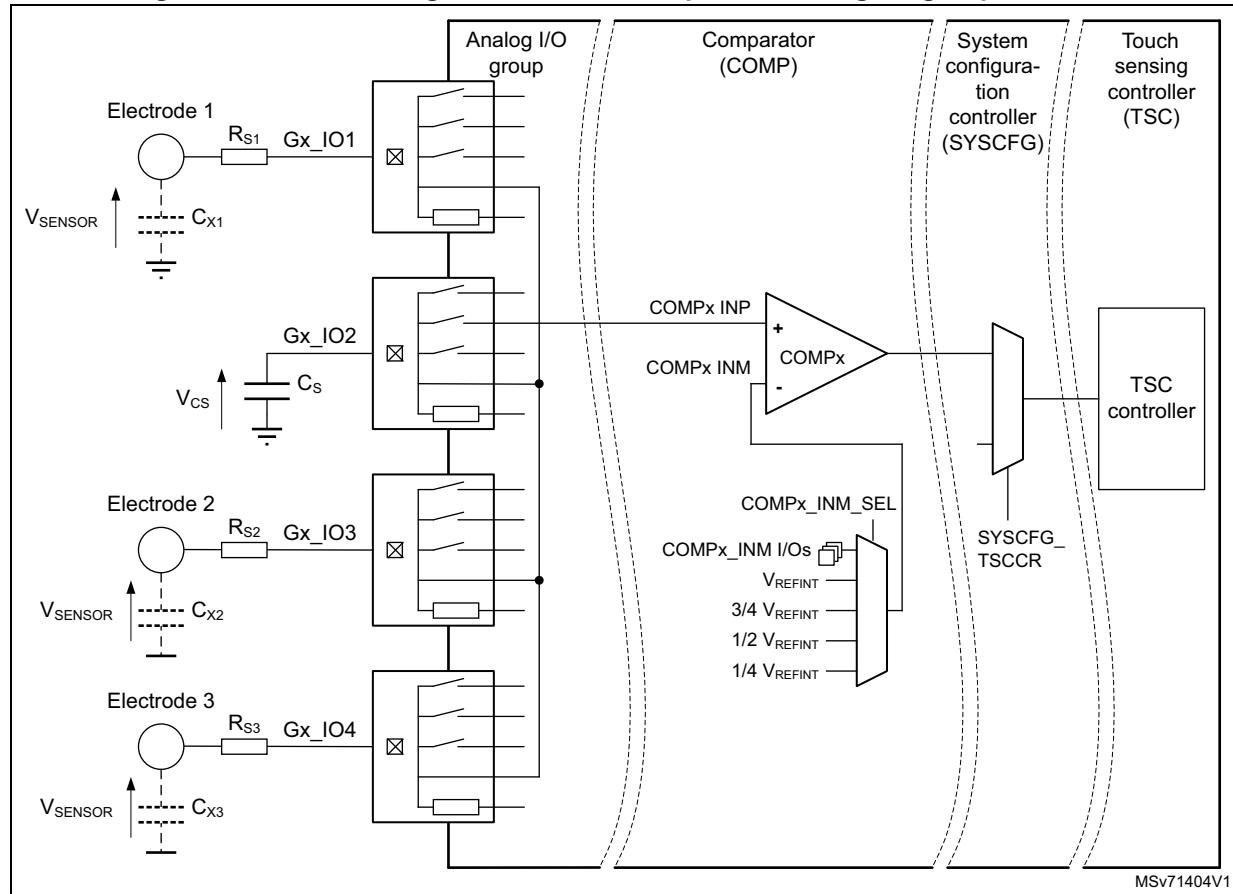
20.5.1 Comparator usage overview

This low-power mode requires that the GPIO dedicated to the sampling capacitor C_S has both the TSC alternate function (TSC_Gx_IOy) and the COMP alternate function (COMPx_INP) available, see product datasheet.

In this case, the TSC peripheral has the possibility to use the microcontroller internal comparator to achieve lower power consumption.

The method uses the same principle as the surface charge transfer acquisition that is the acquisition of the different sensors is automatically stopped when the voltage on the sampling capacitor reaches a certain threshold.

Figure 90. Surface charge transfer with comparator analog I/O group structure



Unlike charge transfer acquisition (where the threshold is fixed = V_{IH}), here the threshold is programmable. The sampling capacitor is internally redirected to the comparator input in order to offer a programmable voltage threshold value V_{COMP} based on V_{REFINT} . The possible configurations are about 0.3, 0.6, 0.9 or 1.2 V (corresponding to $1/4V_{REFINT}$, $1/2V_{REFINT}$, $3/4V_{REFINT}$ or V_{REFINT}).

V_{REFINT} is the bandgap voltage (~1.2 V).

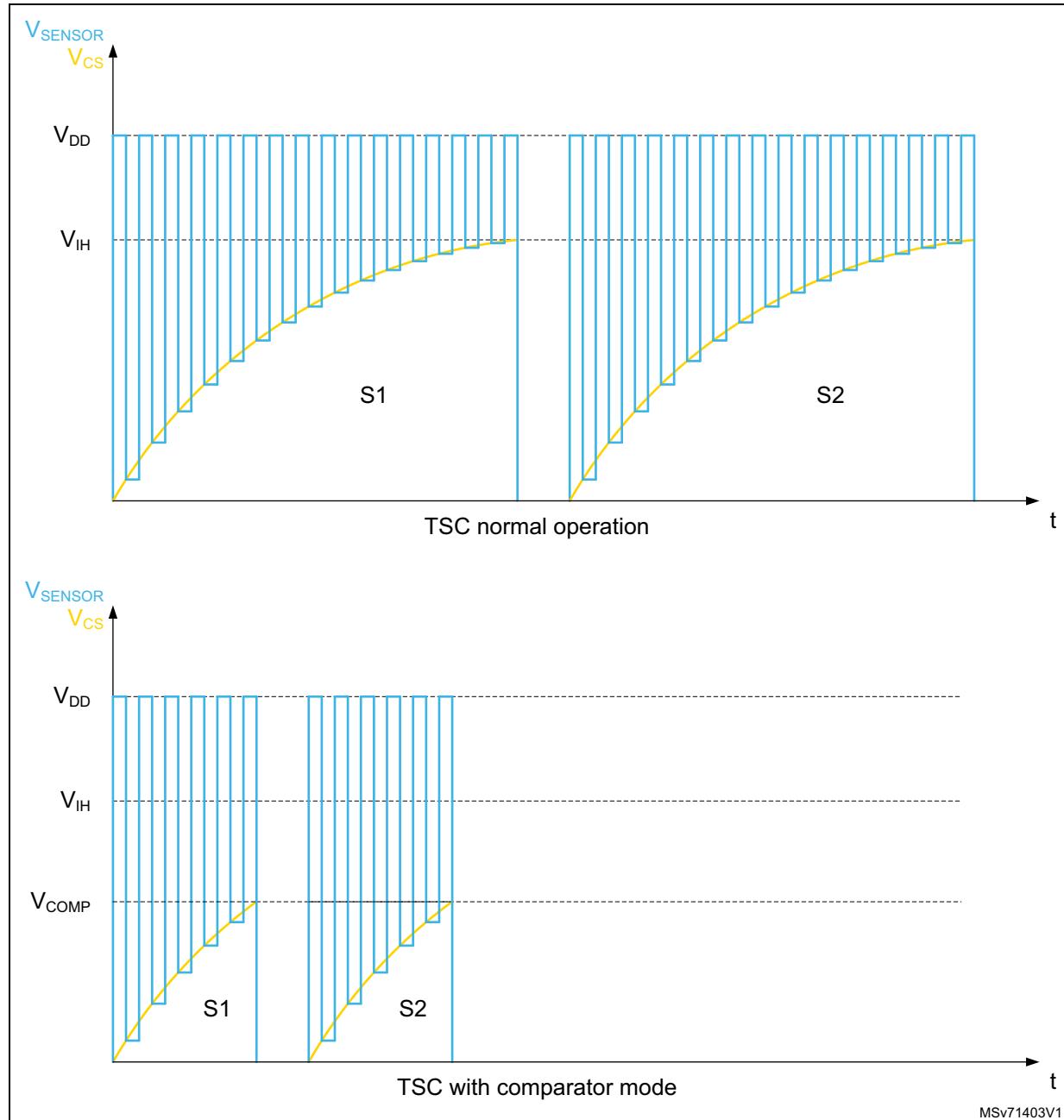
By reducing the voltage threshold, the acquisition time is shortened which in turn reduces the power consumption.

The selection of the threshold value is a compromise between sensor acquisition time (the lowest the threshold the best) and sensor sensitivity (the highest the threshold the best).

[Figure 91](#) below compares the normal mode and the comparator mode (see [Figure 90](#) for V_{SENSOR} and V_{CS} definition).

The SYSCFG_TSCCR register should be set appropriately in order to configure the MUX allowing the use of the comparator mode instead of the usual TSC mode (see device reference manual).

Figure 91. Sensor voltage variation for both normal and comparator mode



Using the internal comparator, the acquisition time is consequently reduced. This implies that the number of counts needed to reach the threshold is also reduced, thus, the TSC peripheral achieves lower power consumption.

The COMPx_INM_SEL and COMPx_INP_SEL bitfields used to configure the comparator are available in [Section 17.6.1: Comparator 1 control and status register \(COMP1_CSR\)](#) and [Section 17.6.2: Comparator 2 control and status register \(COMP2_CSR\)](#).

20.6 TSC interrupts

Table 108. Interrupt control bits

| Interrupt event | Enable control bit | Event flag | Clear flag bit | Exit the Sleep mode | Exit the Stop mode | Exit the Standby mode |
|--------------------|--------------------|------------|----------------|---------------------|--------------------|-----------------------|
| End of acquisition | EOAIE | EOAIF | EOAIC | Yes | No | No |
| Max count error | MCEIE | MCEIF | MCEIC | Yes | No | No |

20.7 TSC 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).

20.7.1 TSC control register (TSC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | SSE |
|-----------|------------|----|----|-----------|------|------|------|----------|----|----|-------|----------|----|-------|------|-----|
| CTPH[3:0] | | | | CTPL[3:0] | | | | SSD[6:0] | | | | | | | | |
| rw | 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 | |
| SSPSC | PGPSC[2:0] | | | Res. | Res. | Res. | Res. | MCV[2:0] | | | IODEF | SYNC POL | AM | START | TSCE | |
| rw | rw | rw | rw | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:28 **CTPH[3:0]**: Charge transfer pulse high

These bits are set and cleared by software. They define the duration of the high state of the charge transfer pulse (charge of C_X).

0000: 1x t_{PGCLK}
0001: 2x t_{PGCLK}

...
1111: 16x t_{PGCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Bits 27:24 **CTPL[3:0]**: Charge transfer pulse low

These bits are set and cleared by software. They define the duration of the low state of the charge transfer pulse (transfer of charge from C_X to C_S).

0000: 1x t_{PGCLK}
0001: 2x t_{PGCLK}
...
1111: 16x t_{PGCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Note: Some configurations are forbidden. Refer to the [Section 20.4.4: Charge transfer acquisition sequence](#) for details.

Bits 23:17 **SSD[6:0]**: Spread spectrum deviation

These bits are set and cleared by software. They define the spread spectrum deviation which consists in adding a variable number of periods of the SSCLK clock to the charge transfer pulse high state.

0000000: 1x t_{SSCLK}
0000001: 2x t_{SSCLK}
...
1111111: 128x t_{SSCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Bit 16 **SSE**: Spread spectrum enable

This bit is set and cleared by software to enable/disable the spread spectrum feature.

0: Spread spectrum disabled
1: Spread spectrum enabled

Note: This bit must not be modified when an acquisition is ongoing.

Bit 15 **SSPSC**: Spread spectrum prescaler

This bit is set and cleared by software. It selects the AHB clock divider used to generate the spread spectrum clock (SSCLK).

0: f_{HCLK}
1: $f_{HCLK}/2$

Note: This bit must not be modified when an acquisition is ongoing.

Bits 14:12 **PGPSC[2:0]**: Pulse generator prescaler

These bits are set and cleared by software. They select the AHB clock divider used to generate the pulse generator clock (PGCLK).

000: f_{HCLK}
 001: $f_{HCLK} /2$
 010: $f_{HCLK} /4$
 011: $f_{HCLK} /8$
 100: $f_{HCLK} /16$
 101: $f_{HCLK} /32$
 110: $f_{HCLK} /64$
 111: $f_{HCLK} /128$

Note: These bits must not be modified when an acquisition is ongoing.

Note: Some configurations are forbidden. Refer to the [Section 20.4.4: Charge transfer acquisition sequence](#) for details.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **MCV[2:0]**: Max count value

These bits are set and cleared by software. They define the maximum number of charge transfer pulses that can be generated before a max count error is generated.

000: 255
 001: 511
 010: 1023
 011: 2047
 100: 4095
 101: 8191
 110: 16383
 111: reserved

Note: These bits must not be modified when an acquisition is ongoing.

Bit 4 **IODEF**: I/O Default mode

This bit is set and cleared by software. It defines the configuration of all the TSC I/Os when there is no ongoing acquisition. When there is an ongoing acquisition, it defines the configuration of all unused I/Os (not defined as sampling capacitor I/O or as channel I/O).

0: I/Os are forced to output push-pull low
 1: I/Os are in input floating

Note: This bit must not be modified when an acquisition is ongoing.

Bit 3 **SYNCPOL**: Synchronization pin polarity

This bit is set and cleared by software to select the polarity of the synchronization input pin.

0: Falling edge only
 1: Rising edge and high level

Bit 2 **AM**: Acquisition mode

This bit is set and cleared by software to select the acquisition mode.

0: Normal acquisition mode (acquisition starts as soon as START bit is set)

1: Synchronized acquisition mode (acquisition starts if START bit is set and when the selected signal is detected on the SYNC input pin)

Note: This bit must not be modified when an acquisition is ongoing.

Bit 1 **START**: Start a new acquisition

This bit is set by software to start a new acquisition. It is cleared by hardware as soon as the acquisition is complete or by software to cancel the ongoing acquisition.

0: Acquisition not started

1: Start a new acquisition

Bit 0 **TSC_E**: Touch sensing controller enable

This bit is set and cleared by software to enable/disable the touch sensing controller.

0: Touch sensing controller disabled

1: Touch sensing controller enabled

Note: When the touch sensing controller is disabled, TSC registers settings have no effect.

20.7.2 TSC interrupt enable register (TSC_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | MCEIE | EOAIE |
| | | | | | | | | | | | | | | rw | rw |

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

Bit 1 **MCEIE**: Max count error interrupt enable

This bit is set and cleared by software to enable/disable the max count error interrupt.

0: Max count error interrupt disabled

1: Max count error interrupt enabled

Bit 0 **EOAIE**: End of acquisition interrupt enable

This bit is set and cleared by software to enable/disable the end of acquisition interrupt.

0: End of acquisition interrupt disabled

1: End of acquisition interrupt enabled

20.7.3 TSC interrupt clear register (TSC_ICR)

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

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

Bit 1 **MCEIC**: Max count error interrupt clear

This bit is set by software to clear the max count error flag and it is cleared by hardware when the flag is reset. Writing a 0 has no effect.

0: No effect

1: Clears the corresponding MCEF of the TSC_ISR register

Bit 0 **EOAIC**: End of acquisition interrupt clear

This bit is set by software to clear the end of acquisition flag and it is cleared by hardware when the flag is reset. Writing a 0 has no effect.

0: No effect

1: Clears the corresponding EOAF of the TSC_ISR register

20.7.4 TSC interrupt status register (TSC_ISR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit 1 **MCEF**: Max count error flag

This bit is set by hardware as soon as an analog I/O group counter reaches the max count value specified. It is cleared by software writing 1 to the bit MCEIC of the TSC_ICR register.

0: No max count error (MCE) detected

1: Max count error (MCE) detected

Bit 0 **EOAF**: End of acquisition flag

This bit is set by hardware when the acquisition of all enabled group is complete (all GxS bits of all enabled analog I/O groups are set or when a max count error is detected). It is cleared by software writing 1 to the bit EOAIC of the TSC_ICR register.

0: Acquisition is ongoing or not started

1: Acquisition is complete

20.7.5 TSC I/O hysteresis control register (TSC_IOHCR)

Address offset: 0x10

Reset value: 0xFFFF FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Res. | Res. | Res. | Res. | G7_IO4 | G7_IO3 | G7_IO2 | G7_IO1 | G6_IO4 | G6_IO3 | G6_IO2 | G6_IO1 | G5_IO4 | G5_IO3 | G5_IO2 | G5_IO1 |
| | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| G4_IO4 | G4_IO3 | G4_IO2 | G4_IO1 | G3_IO4 | G3_IO3 | G3_IO2 | G3_IO1 | G2_IO4 | G2_IO3 | G2_IO2 | G2_IO1 | G1_IO4 | G1_IO3 | G1_IO2 | G1_IO1 |
| rw |

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

Bits 27:0 **Gx_IOy**: Gx_IOy Schmitt trigger hysteresis mode

These bits are set and cleared by software to enable/disable the Gx_IOy Schmitt trigger hysteresis.

0: Gx_IOy Schmitt trigger hysteresis disabled

1: Gx_IOy Schmitt trigger hysteresis enabled

Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).

20.7.6 TSC I/O analog switch control register (TSC_IOASCR)

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. | G7_IO4 | G7_IO3 | G7_IO2 | G7_IO1 | G6_IO4 | G6_IO3 | G6_IO2 | G6_IO1 | G5_IO4 | G5_IO3 | G5_IO2 | G5_IO1 |
| | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| G4_IO4 | G4_IO3 | G4_IO2 | G4_IO1 | G3_IO4 | G3_IO3 | G3_IO2 | G3_IO1 | G2_IO4 | G2_IO3 | G2_IO2 | G2_IO1 | G1_IO4 | G1_IO3 | G1_IO2 | G1_IO1 |
| rw |

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

Bits 27:0 **Gx_IOy**: Gx_IOy analog switch enable

These bits are set and cleared by software to enable/disable the Gx_IOy analog switch.

0: Gx_IOy analog switch disabled (opened)

1: Gx_IOy analog switch enabled (closed)

Note: These bits control the I/O analog switch whatever the I/O control mode is (even if controlled by standard GPIO registers).

20.7.7 TSC I/O sampling control register (TSC_IOSCR)

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. | G7_IO4 | G7_IO3 | G7_IO2 | G7_IO1 | G6_IO4 | G6_IO3 | G6_IO2 | G6_IO1 | G5_IO4 | G5_IO3 | G5_IO2 | G5_IO1 |
| | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| G4_IO4 | G4_IO3 | G4_IO2 | G4_IO1 | G3_IO4 | G3_IO3 | G3_IO2 | G3_IO1 | G2_IO4 | G2_IO3 | G2_IO2 | G2_IO1 | G1_IO4 | G1_IO3 | G1_IO2 | G1_IO1 |
| rw |

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

Bits 27:0 **Gx_IOy**: Gx_IOy sampling mode

These bits are set and cleared by software to configure the Gx_IOy as a sampling capacitor I/O. Only one I/O per analog I/O group must be defined as sampling capacitor.

0: Gx_IOy unused

1: Gx_IOy used as sampling capacitor

Note: These bits must not be modified when an acquisition is ongoing.

During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

20.7.8 TSC I/O channel control register (TSC_IOCCR)

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. | G7_IO4 | G7_IO3 | G7_IO2 | G7_IO1 | G6_IO4 | G6_IO3 | G6_IO2 | G6_IO1 | G5_IO4 | G5_IO3 | G5_IO2 | G5_IO1 |
| | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| G4_IO4 | G4_IO3 | G4_IO2 | G4_IO1 | G3_IO4 | G3_IO3 | G3_IO2 | G3_IO1 | G2_IO4 | G2_IO3 | G2_IO2 | G2_IO1 | G1_IO4 | G1_IO3 | G1_IO2 | G1_IO1 |
| rw |

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

Bits 27:0 **Gx_IOy**: Gx_IOy channel mode

These bits are set and cleared by software to configure the Gx_IOy as a channel I/O.

0: Gx_IOy unused

1: Gx_IOy used as channel

Note: These bits must not be modified when an acquisition is ongoing.

During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

20.7.9 TSC I/O group control status register (TSC_IOGCSR)

Address offset: 0x30

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|
| Res. | G7S | G6S | G5S | G4S | G3S | G2S | G1S |
| | | | | | | | | | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | G7E | G6E | G5E | G4E | G3E | G2E | G1E |
| | | | | | | | | | rw |

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **GxS**: Analog I/O group x status

These bits are set by hardware when the acquisition on the corresponding enabled analog I/O group x is complete. They are cleared by hardware when a new acquisition is started.

0: Acquisition on analog I/O group x is ongoing or not started

1: Acquisition on analog I/O group x is complete

Note: When a max count error is detected the remaining GxS bits of the enabled analog I/O groups are not set.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **GxE**: Analog I/O group x enable

These bits are set and cleared by software to enable/disable the acquisition (counter is counting) on the corresponding analog I/O group x.

0: Acquisition on analog I/O group x disabled

1: Acquisition on analog I/O group x enabled

20.7.10 TSC I/O group x counter register (TSC_IOGxCR)

x represents the analog I/O group number.

Address offset: 0x30 + 0x04 * x, (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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | | | | | | | | CNT[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 **CNT[13:0]**: Counter value

These bits represent the number of charge transfer cycles generated on the analog I/O group x to complete its acquisition (voltage across C_S has reached the threshold).

20.7.11 TSC register map

Table 109. TSC register map and reset values

Table 109. TSC 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 |
|--------|---------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 0x003C | TSC_IOG3CR | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| | Reset value | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| 0x0040 | TSC_IOG4CR | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| | Reset value | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| 0x0044 | TSC_IOG5CR | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| | Reset value | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| 0x0048 | TSC_IOG6CR | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| | Reset value | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| 0x004C | TSC_IOG7CR | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| | Reset value | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

21 True random number generator (RNG)

21.1 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 is a NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

21.2 RNG main features

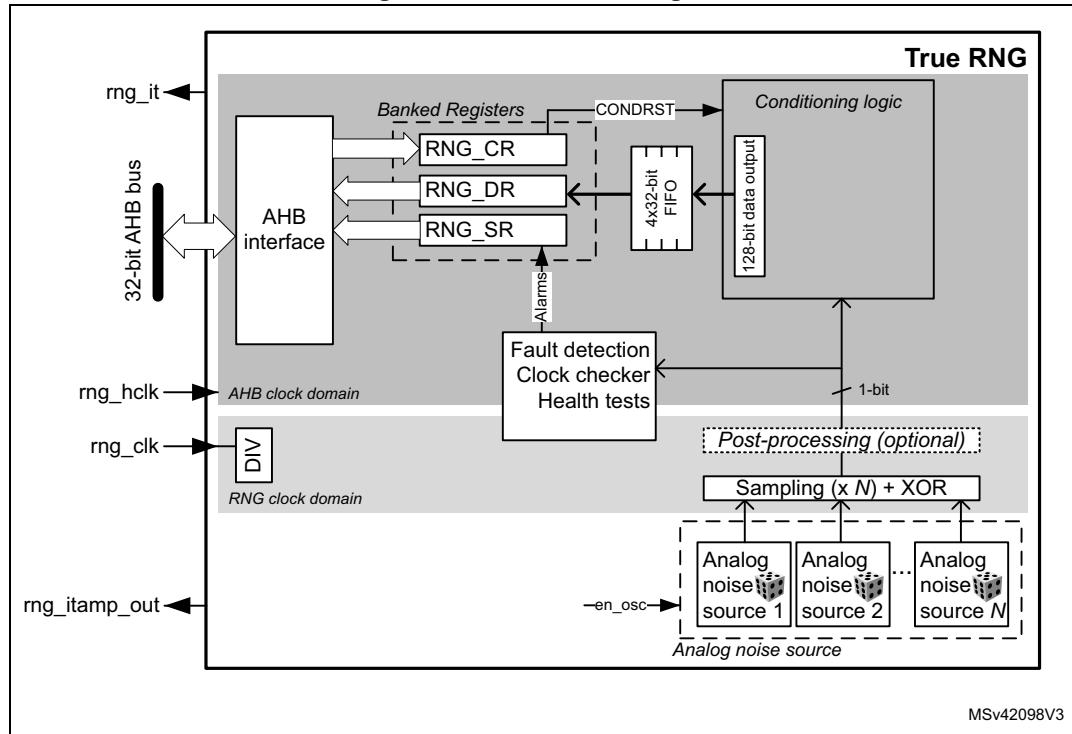
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a nondeterministic random bit generator (NDRBG).
- In the NIST configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- It embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- 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).

21.3 RNG functional description

21.3.1 RNG block diagram

Figure 92 shows the RNG block diagram.

Figure 92. RNG block diagram



21.3.2 RNG internal signals

Table 110 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 110. RNG internal input/output signals

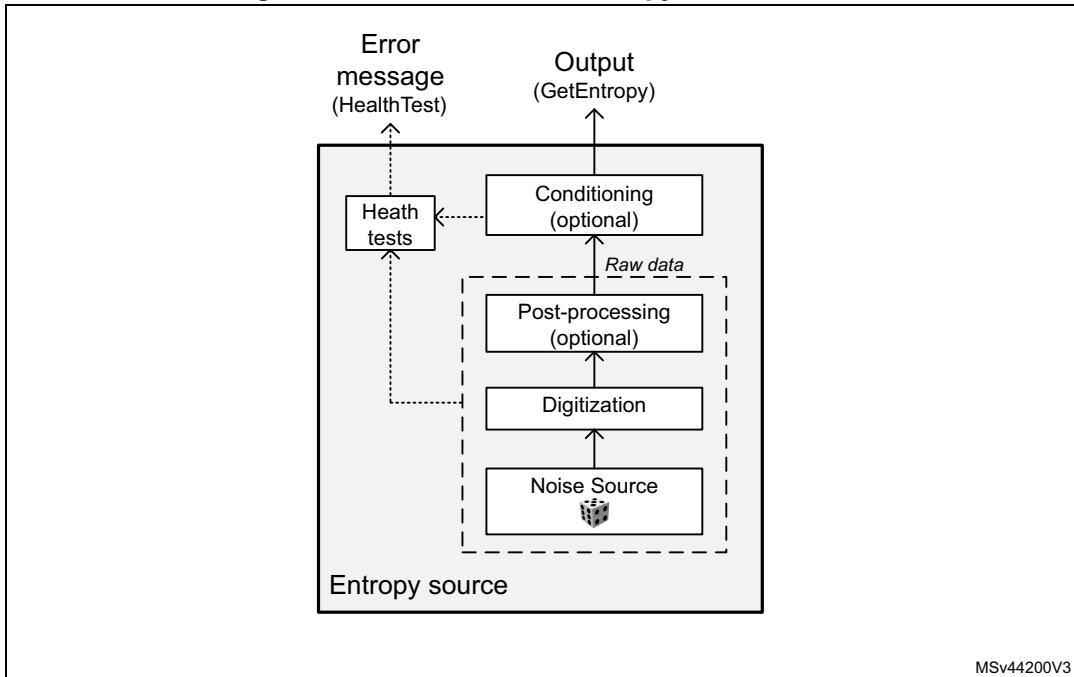
| Signal name | Signal type | Description |
|---------------|----------------|--|
| rng_it | Digital output | RNG global interrupt request |
| rng_hclk | Digital input | AHB clock |
| rng_clk | Digital input | RNG dedicated clock, asynchronous to rng_hclk |
| rng_itamp_out | digital output | RNG internal tamper event signal to TAMP (XORed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples, requiring a reset and a new initialization to be usable again. |

21.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary RNG integrates all the required NIST components depicted on [Figure 93](#). Those components are an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

Figure 93. NIST SP800-90B 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. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x6), 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 21.3.8: RNG low-power use](#). Multiple oscillators are also disabled for the configuration A (see [Table 113: RNG configurations](#)).
- The XORing of all the noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

Note: In [Section 21.6: RNG entropy source validation](#) the recommended RNG clock frequencies and associated divider value are given.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In non-NIST configuration B (as defined in [Section 21.6.2](#)) a normalization debiasing is applied, that is half of the bits are taken from the sampled noise source, half of the bits are taken from the inverted sampled noise source.

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). The NIST SP800-90B target is full entropy on the output (128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in [Section 21.5: RNG processing time](#).

Output buffer

A data output buffer can store up to four 32-bit words that 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 after a number of clock cycles specified in [Section 21.5: RNG processing time](#).

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 in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (configuration A in [Section 21.6.2](#)).

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
 - Repetition count test, flagging an error when the noise source has provided more than 42 consecutive bits at a constant value (0 or 1).
 - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 711 times.
 - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source:
 - Repetition count test, similar to the one running in startup tests.
 - Adaptive proportion test, similar to the one running in startup tests.
3. Vendor specific continuous tests
 - Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrences of 2-bit patterns (01 or 10).
 - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than AHB clock cycle divided by 32.
4. On-demand test of digitized noise source (raw data)
 - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in [Section 21.3.4: RNG initialization](#)). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in [Section 21.3.7: Error management](#).

Note: An interrupt can be generated when an error is detected.

Above the health test thresholds are modified by changing the value in the RNG_HTCR register. See [Section 21.6: RNG entropy source validation](#) for details.

21.3.4 RNG initialization

The RNG simplified state machine is pictured on [Figure 94](#).

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 cycles of RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering startup behavior test on the raw data samples and known-answer tests.
3. When startup health tests are completed. During this time, three 128-bit noise source samples are used.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in [Section 21.5: RNG processing time](#).

Figure 94. RNG initialization overview

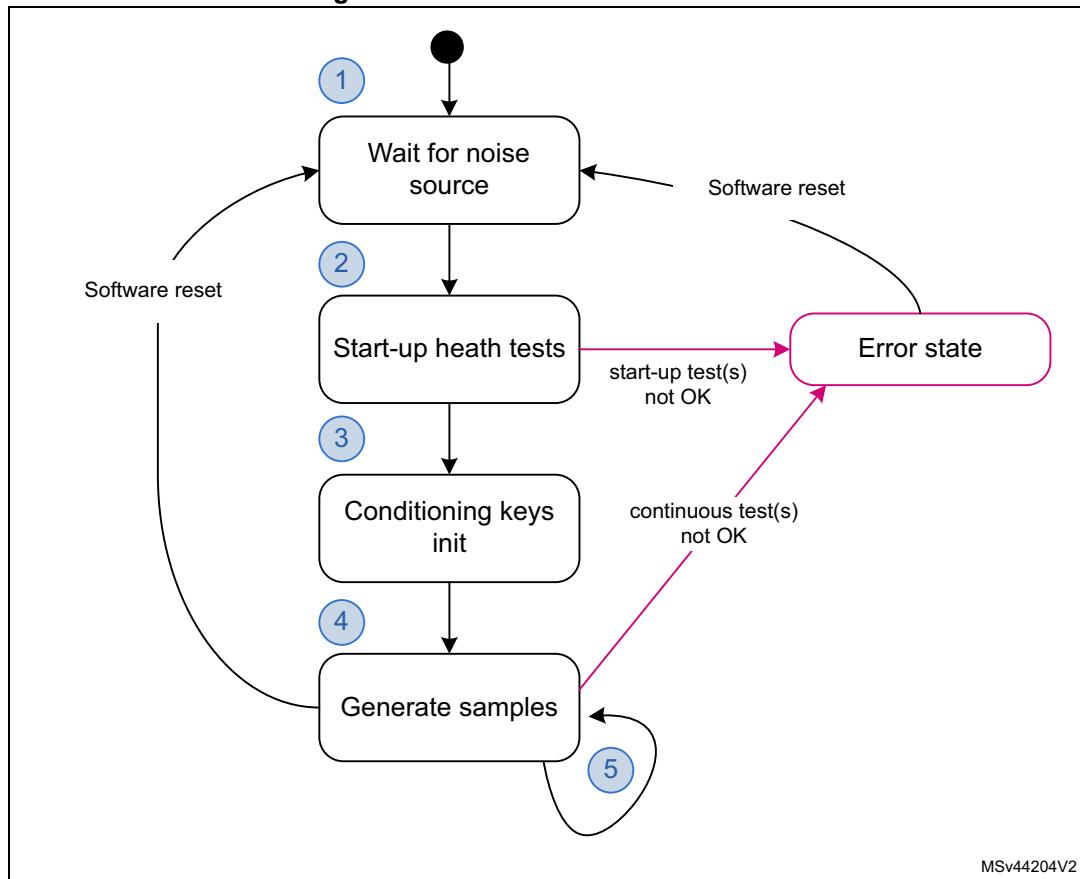


Figure 94 also highlights a possible software reset sequence, implemented by:

1. Writing bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
2. Then writing RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
3. Wait for random number to be ready, after initialization completes.

Note: When the RNG peripheral is reset through RCC (hardware reset), the RNG configuration for optimal randomness is lost in the RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

21.3.5 RNG operation

Normal operations

To run the RNG using interrupts, the following steps are recommended:

1. Consult [Section 21.6: RNG entropy source validation](#) and verify if a specific RNG configuration is required for the application.
 - If it is the case, write in the RNG_CR register the bit CONDRST = 1 together with the correct RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0, the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.
 - If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE = 1 and the RNG enable 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 21.3.7](#) must be used.

To run the RNG in polling mode following steps are recommended:

1. Consult [Section 21.6: RNG entropy source validation](#) and verify if a specific RNG configuration is required for the application.
 - If it is the case write in the RNG_CR register the bit CONDRST = 1 together with the correction RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
 - If it is not the case only enable the RNG 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 21.3.7](#) 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).*

If the random number generation period is a concern to the application and if NIST compliance is not required it is possible to select a faster RNG configuration by using the RNG configuration “B”, described in [Section 21.6: RNG entropy source validation](#). The gain in random number generation speed is summarized in [Section 21.5: RNG processing time](#).

Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in [Section 21.3.8: RNG low-power use](#).

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in [Section 21.3.3: Random number generation](#).

21.3.6 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, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in [Section 21.6: RNG entropy source validation](#).

Note: *When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider **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 21.3.1: RNG block diagram](#) for details (AHB and RNG clock domains).

21.3.7 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 21.3.6: 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.

The following sequence must be used to fully recover from a seed error:

1. Software reset by writing CONDRST at 1 and at 0 (see bitfield description for details). This step is needed only if SECS is set. Indeed, when SEIS is set and SECS is cleared it means RNG performed the reset automatically (auto-reset). In this case application must clear the SEIS bit interrupt flag.
2. If SECS was set in step 1 (no auto-reset) wait for CONDRST to be cleared in the RNG_CR register, then confirm that SEIS is cleared in the RNG_SR register. Otherwise, just clear the SEIS bit in the RNG_SR register.
3. If SECS was set in step 1 (no auto-reset), wait for SECS to be cleared by RNG. The random number generation is now back to normal.

Note: *After a seed error RNG restarts generating random numbers when SECS is cleared.*

When the application sets the ARDIS bit in the RNG_CR register, the auto-reset is disabled. CONDRST must be used in step 1.

RNG tamper errors

When an unexpected error is found by the RNG an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral either using the central reset management or the global SoC reset. Then a proper initialization of the RNG is required, again.

21.3.8 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 RNG must be reenabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode) and a complete conditioning round is done. Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in [Section 21.5: RNG processing time on page 494](#).

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, unless the application resets the conditioning logic using CONDRST bit in the RNG_CR register.

When the application wants to disable the RNG clock it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and disabling the RNG kernel clock using the RCC.

Also, when application needs to enter a power mode where RNG is de-activated, it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and entering the low power mode using the PWR.

In the two cases above, to avoid unexpected consumption when RNG analog oscillators stay active, application can set the bit 13 in RNG_CR register. Setting this bit adds some marginal power consumption while RNGEN bit is set (RNG activated).

Note: *The power modes where RNG is deactivated (that is retained or not available) can be found in the PWR section.*

21.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see [Section 21.3.7: Error management](#)
- Clock error, see [Section 21.3.7: Error management](#)

Dedicated interrupt enable control bits are available as shown in [Table 111](#).

Table 111. 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 CONDRST to 1 then to 0 unless ARDIS is cleared (see Section 21.3.7: Error management) |
| | 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.*

21.5 RNG processing time

In recommended configuration A or C described in [Table 113](#), the time between two sets of four 32-bit data is either:

- $203 \times N$ AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- $128 \times N$ RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

With $f_{threshold} = 1.6 \times f_{RNG}$, for instance 77 MHz if $f_{RNG} = 48$ MHz. Value N is defined in [Section 21.6: RNG entropy source validation](#).

Note: When $CLKDIV$ is different from zero, f_{RNG} must take into account the internal divider ratio.

If configuration B is selected the performance figures become:

- 203 AHB cycles if $f_{AHB} < f_{threshold}$ or
- 32 RNG cycles $f_{AHB} \geq f_{threshold}$

with $f_{threshold} = 6.5 \times f_{RNG}$.

Initialization time

More time is needed for the first set of random numbers after the device exits reset (see [Section 1.3.4: RNG initialization](#)). Table below gives details on how to compute the time spent in each initialization step.

Table 112. RNG initialization times

| Initialization step | Configuration A or C, reset value | Configuratton B |
|--|--|--|
| Wait for noise source, then startup health tests | Max((wait_noise ⁽¹⁾ + 11 + 1024 x CLKDIV) RNG_cycles, 13 x 203 AHB_cycles) | Max(16 + 1035 RNG_cycles, 13 x 203 AHB_cycles) |
| Conditioning keys initialization | Max((1+2xN) x 128 x CLKDIV) RNG_cycles + 203 AHB_cycles, (128 x CLKDIV) RNG_cycles + (2xN+2) x 203 AHB_cycles) | Max (3 x 32 RNG_cycles + 203 AHB_cycles, 32 RNG_cycles + 4 x 203 AHB_cycles) |

1. 192 RNG_cycles (configuration A or C), 16 RNG_cycles (reset value)

As an example, if AHB clock= 54 MHz and RNG clock= 48 MHz then the initialization time is around 79 μ s in the configuration C (CLKDIV=1, N=2) and 66 μ s in the configuration B.

21.6 RNG entropy source validation

21.6.1 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), and NIST SP800-90B test suite.

21.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock `rng_clk` = 48 MHz
- RNG configurations are described in [Table 113](#). Note that only configuration A can be certified NIST SP800-90B. Refer to [Table 114](#) to select the best configuration for the application.
- Configuration C can be used when configuration A is not flagged as certified.

Table 113. RNG configurations

| Configuration | RNG_CR bits | | | | | | Loop number (N) | RNG_HTCR register | RNG_NCSR register |
|---------------|---|-------------------|--------------------|----------------------------------|----------------------------------|---------|-----------------|---------------------|-------------------|
| | NISTC bit | RNG_CONFIG1 [5:0] | CLKDIV [3:0] | RNG_CONFIG2 [2:0] ⁽¹⁾ | RNG_CONFIG3 [3:0] ⁽²⁾ | CED bit | | | |
| A | Refer to <i>NIST compliant RNG configuration</i> table in AN4230 available from www.st.com . This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on https://csrc.nist.gov/projects/cryptographic-module-validation-program . | | | | | | | | |
| B | 1 | 0x18 | 0x0 ⁽³⁾ | 0x0 | 0x0 | 0 | 1 | 0x0000 | default |
| C | 0 | 0x0F | | 0x0 | 0xD | 0 | 2 | AAC7 ⁽⁴⁾ | default |

1. 0x1 value is recommended instead of 0x0 for `RNG_CONFIG2[2:0]`, when RNG power consumption is critical. See the end of [Section 21.3.8: RNG low-power use](#) for details.
2. `RNG_CONFIG3[1:0]` defines the loop number N: 0x0 corresponds to N=1, 0x1 to N=2, 0x2 to N=3, 0x3 to N=4
3. The noise source sampling must be 48 MHz or less. Hence, if the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in [Section 21.7.1](#) for details.
4. This value can be fixed in the RNG driver (it doesn't depend on the STM32 family).

Table 114. Configuration selection

| Section criteria | Config A | Config B | Config C |
|--|-----------|----------|-----------|
| Suitable to generate NIST compliant cryptographic keys | Yes | No | |
| Entropy ⁽¹⁾ | Certified | Good | Very good |
| Speed ⁽²⁾ | Baseline | Faster | Baseline |

1. For configurations B and C entropy is verified using AIS-31 test suite (T0 to T8).
2. When speed is not enough for application a NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to “STM32 microcontrollers random number generation validation using NIST statistical test suite” application note (AN4230) available from www.st.com.

21.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

21.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------------------|-------------|------|-------|------------------|------|------------------|----|-------|------|-----|------|----|-------|-------------|------|
| CONF1 GLOCK | COND RST | Res. | Res. | Res. | Res. | RNG_CONFIG1[5:0] | | | | | | | | CLKDIV[3:0] | |
| rs | 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 |
| RNG_CONFIG2[2:0] | | | NISTC | RNG_CONFIG3[3:0] | | | | ARDIS | Res. | CED | Res. | IE | RNGEN | Res. | Res. |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | | rw | rw | | |

Bit 31 **CONFIGLOCK**: RNG Config lock

0: Writes to the RNG_NSSCR, RNG_HTCR and RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSSCR, RNG_HTCR and RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

Bit 30 **COND_RST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_CR, RNG_NSSCR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that set any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

When CONDRST is set to 0 by the software, its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.

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

Bits 25:20 **RNG_CONFIG1[5:0]**: RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in [Section 21.6: RNG entropy source validation](#).

Writing any bit of RNG_CONFIG1 is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0). 0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing these bits is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 15:13 **RNG_CONFIG2[2:0]**: RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Bit 13 can be set when RNG power consumption is critical. See [Section 21.3.8: RNG low-power use](#). Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration per 128-bit output two conditioning loops are performed and 256 bits of noise source are used.
1: Custom values for NIST compliant RNG. See [Section 21.6: RNG entropy source validation](#) for recommended configuration.
Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 11:8 **RNG_CONFIG3[3:0]**: RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details. If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.
0: Auto-reset enabled
1: Auto-reset disabled
Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in [Section 21.3.7: Error management](#). Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 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.
Writing this bit is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

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.

21.7.2 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 (unless CONDRST is used). 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 clock before the internal divider 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:

- Runtime repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0 or 1, or more than 32 consecutive occurrence of two bits patterns 01 or 10)
- Startup or continuous adaptive proportion test on noise source failed.
- Startup post-processing/conditioning sanity check failed.

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.

21.7.3 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. 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).

21.7.4 RNG noise source control register (RNG_NSSCR)

Address offset: 0x00C

Reset value: 0x0003 FFFF

Writing in RNG_NSSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK = 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. | EN_OSC6[2:1] |
| | | | | | | | | | | | | | | | rw rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EN_OSC6[0] | EN_OSC5[2:0] | | | EN_OSC4[2:0] | | | EN_OSC3[2:0] | | | EN_OSC2[2:0] | | | EN_OSC1[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 17:15 **EN_OSC6[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 6. The bitfield has no effect otherwise.

Bits 14:12 **EN_OSC5[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 5. The bitfield has no effect otherwise.

Bits 11:9 EN_OSC4[2:0]:

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 4. The bitfield has no effect otherwise.

Bits 8:6 EN_OSC3[2:0]:

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 EN_OSC2[2:0]:

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 EN_OSC1[2:0]:

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

21.7.5 RNG health test control register (RNG_HTCR)

Address offset: 0x010

Reset value: 0x0000 72AC

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK=1.

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

Bits 31:0 HTCFG[31:0]: health test configuration

This configuration is used by RNG to configure the health tests. See [Section 21.6: RNG entropy source validation](#) for the recommended value.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

21.7.6 RNG register map

Table 115. RNG register map and reset map

Refer to [Section 2.2: Memory organization](#) for the register boundary addresses.

22 AES hardware accelerator (AES)

22.1 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).

22.2 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 write-only 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

22.3 AES implementation

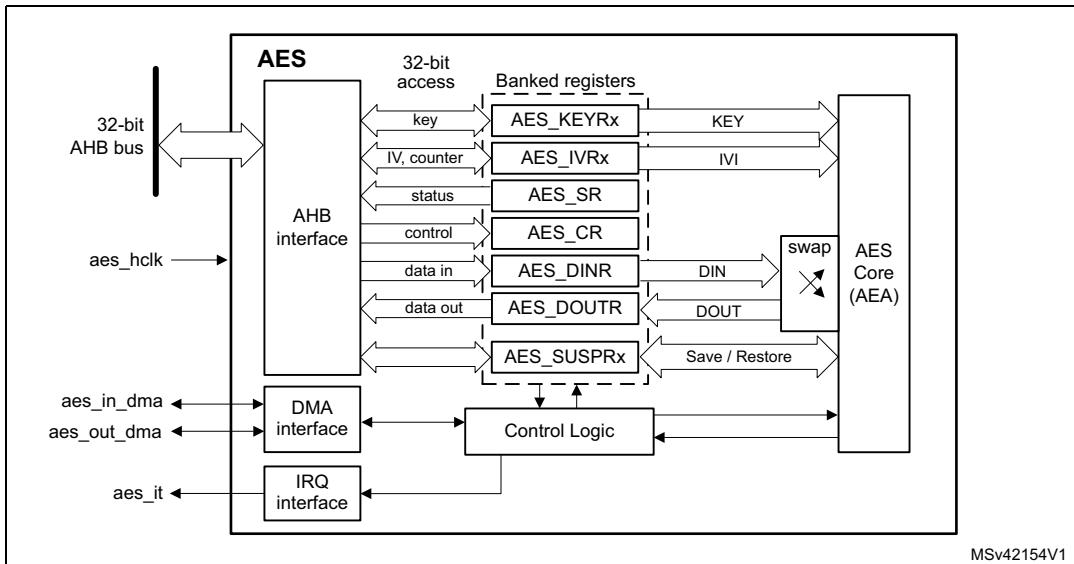
The devices have one AES peripheral, implemented as per the following table.

22.4 AES functional description

22.4.1 AES block diagram

Figure 95 shows the block diagram of AES.

Figure 95. AES block diagram



MSv42154V1

22.4.2 AES internal signals

Table 116 describes the user relevant internal signals interfacing the AES peripheral.

Table 116. 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 |

22.4.3 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.

Note: *Mode 2 is only used when performing ECB and CBC decryption.*

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 22.4.4: AES procedure to perform a cipher operation on page 509](#).

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)

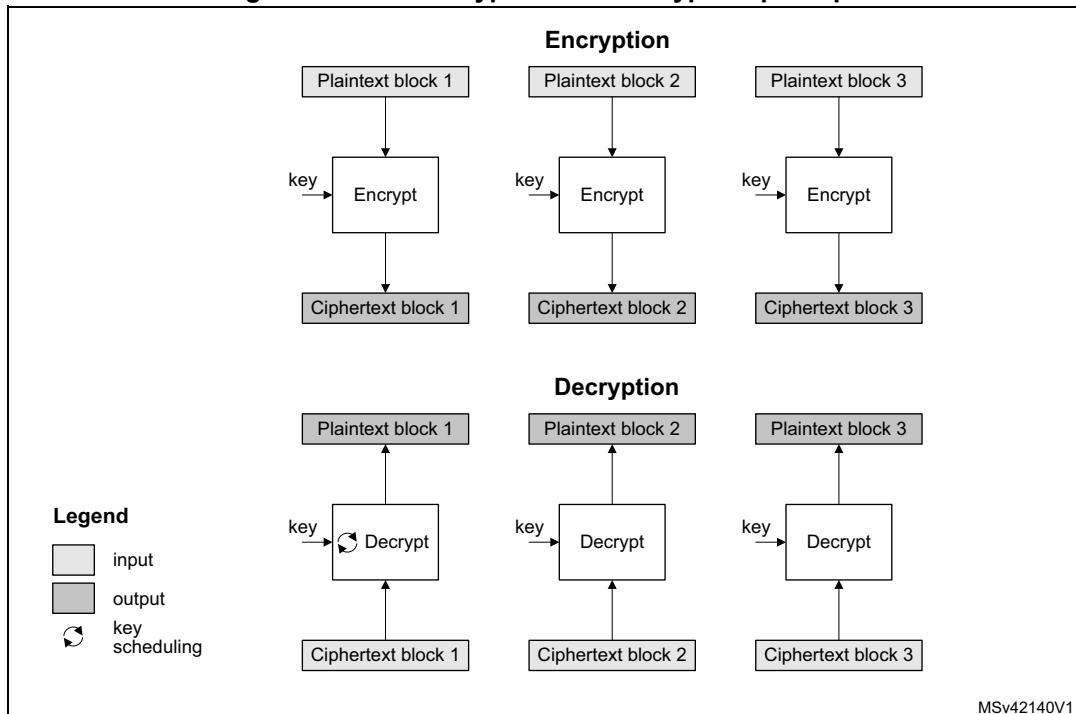
Note: *The chaining mode may be changed only when AES is disabled (bit EN of the AES_CR register cleared).*

Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from [Section 22.4.8: AES basic chaining modes \(ECB, CBC\)](#).

Electronic codebook (ECB) mode

Figure 96. ECB encryption and decryption principle

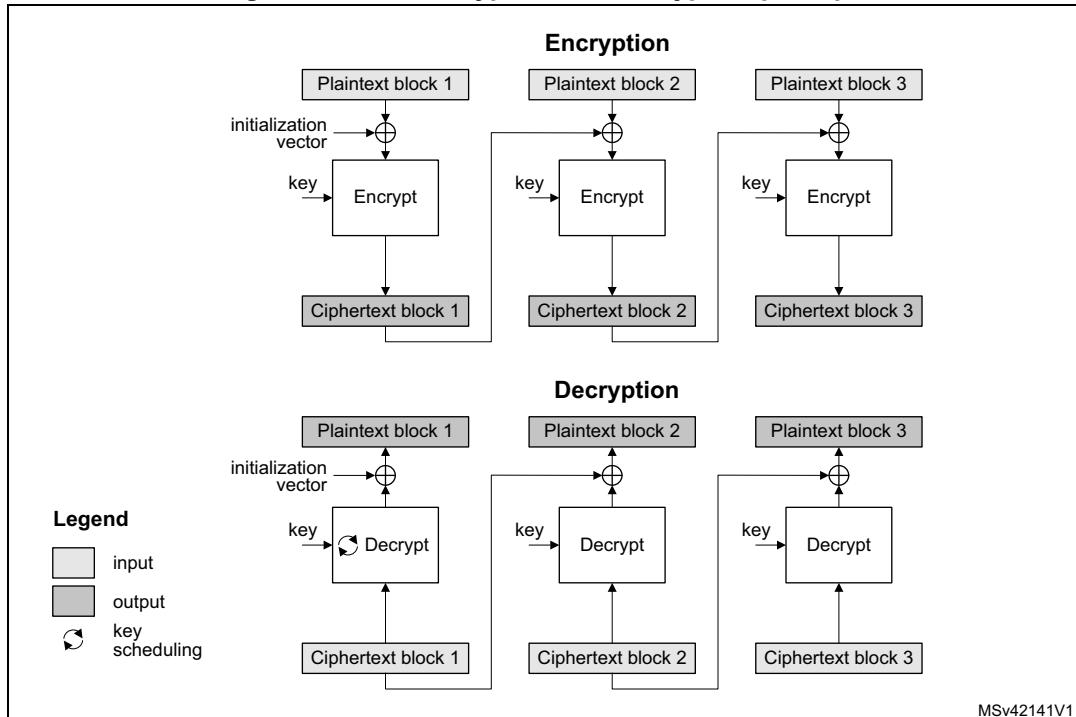


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.

Cipher block chaining (CBC) mode

Figure 97. CBC encryption and decryption principle

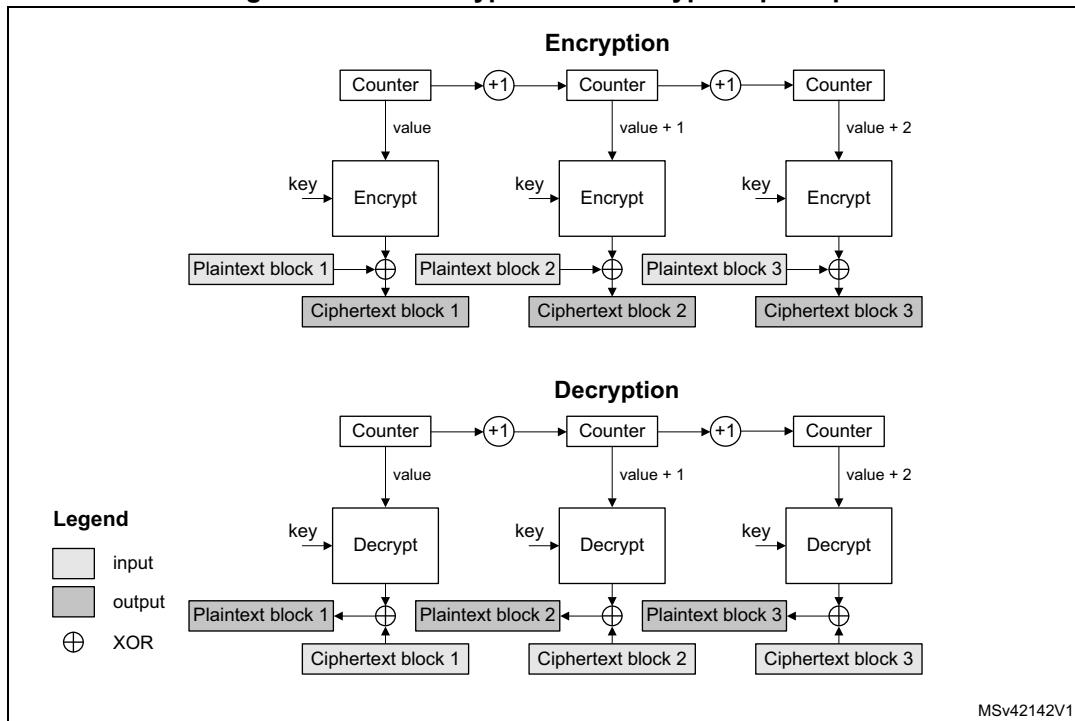


In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

Note: For decryption, a special key scheduling is required before processing the first block.

Counter (CTR) mode

Figure 98. CTR encryption and decryption principle

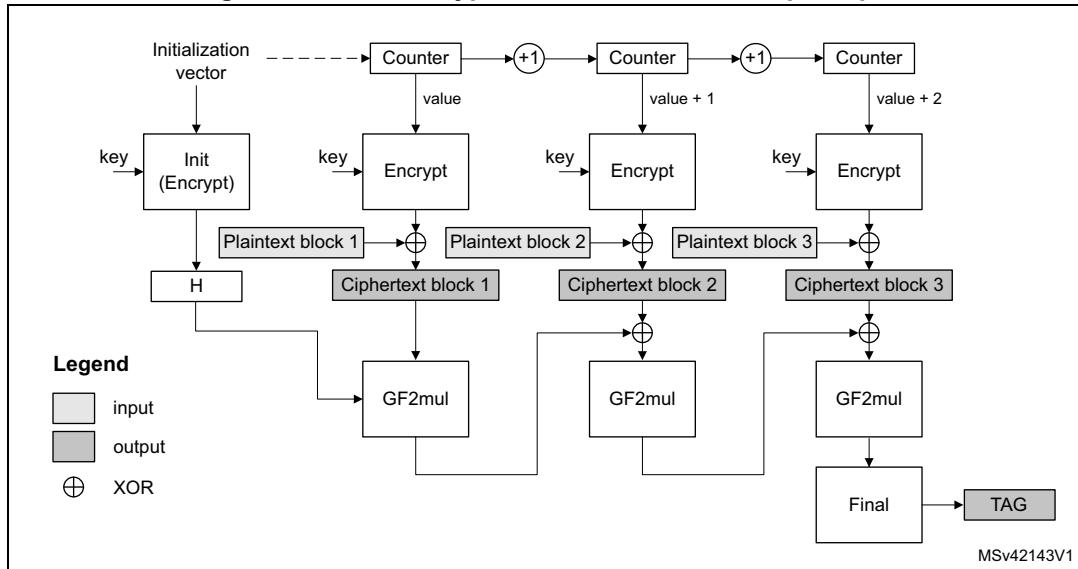


The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext as specified in NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation*.

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 99. GCM encryption and authentication principle

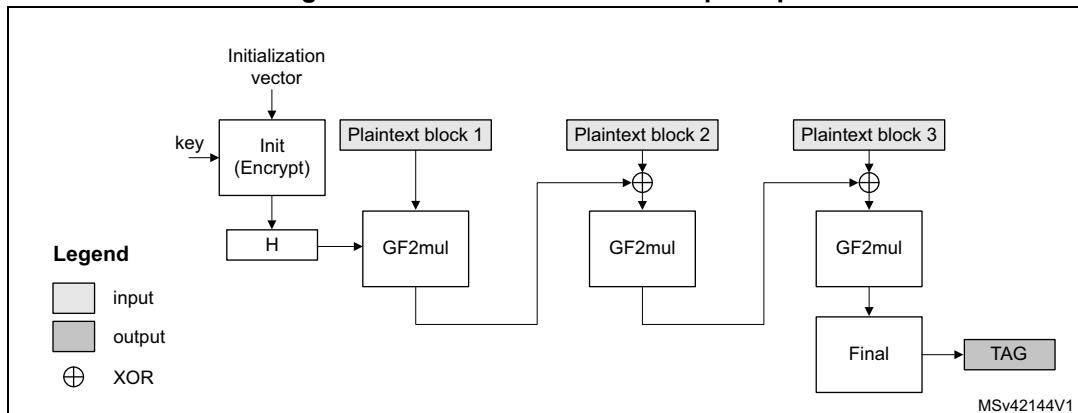


In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in *NIST Special Publication 800-38D, Recommendation for Block 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 100. GMAC authentication principle

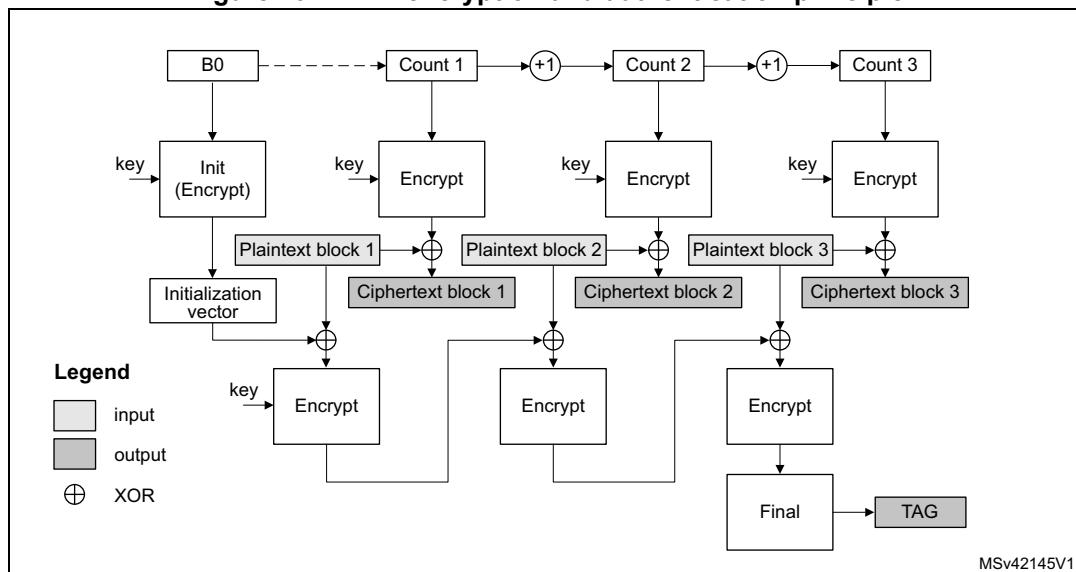


Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in *NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

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 101. 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.

22.4.4 AES procedure to perform a cipher operation

Introduction

A typical cipher operation is explained below. Detailed information is provided in sections starting from [Section 22.4.8: 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 22.4.5: 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 22.4.6: 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 CCF bit of the AES_ICR 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_IER 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 CCF bit of the AES_ICR register.
 - c) If the data block just processed is the second-last block of a 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 22.4.16: 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.

22.4.5 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. 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.

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.

Note:

The operation of the key preparation lasts 59 or 82 clock cycles, depending on the key size (128- or 256-bit).

22.4.6 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 22.4.4: AES procedure to perform a cipher operation on page 509](#) 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 22.4.13: AES data registers and data swapping](#) for details).

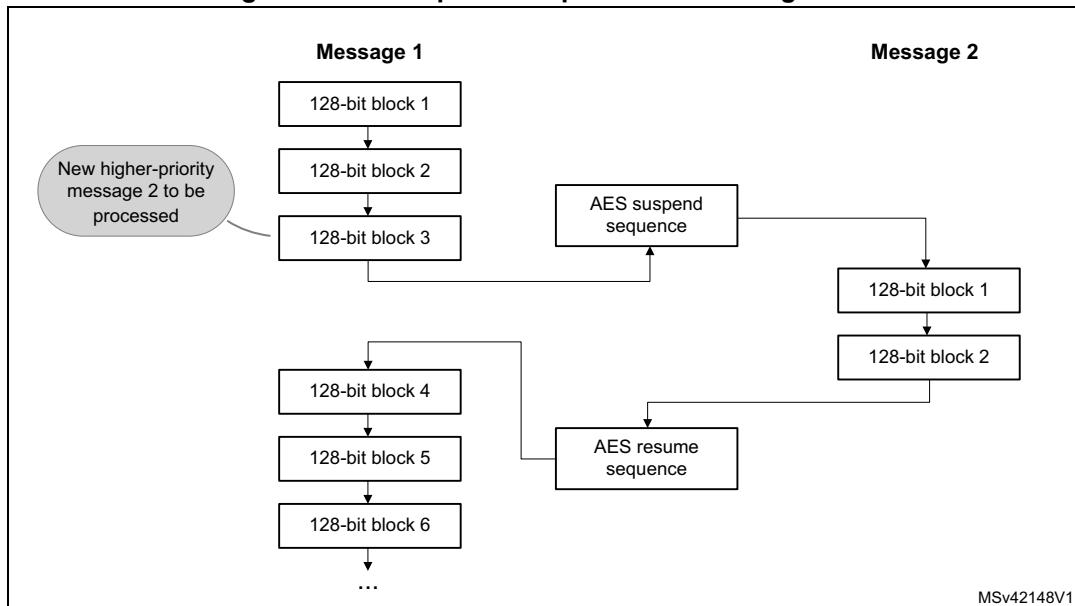
22.4.7 AES task suspend and resume

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 102 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 102. Example of suspend mode management



A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

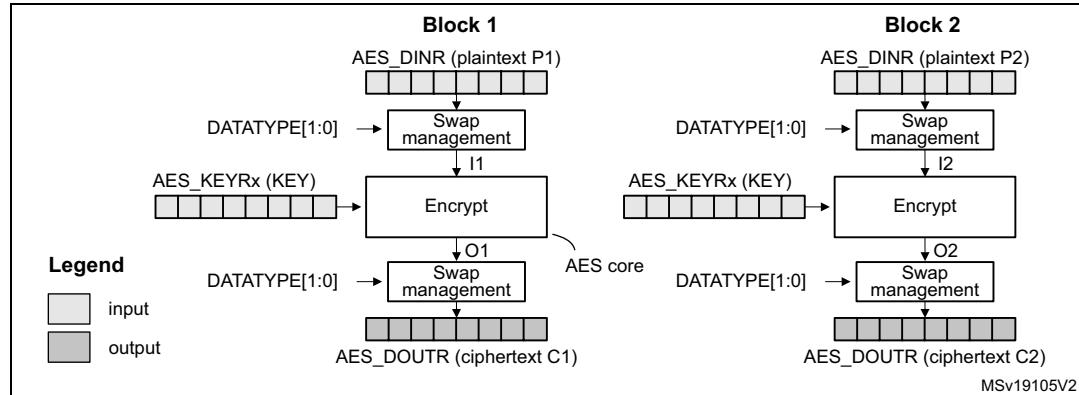
22.4.8 AES basic chaining modes (ECB, CBC)

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 103 illustrates the electronic codebook (ECB) encryption.

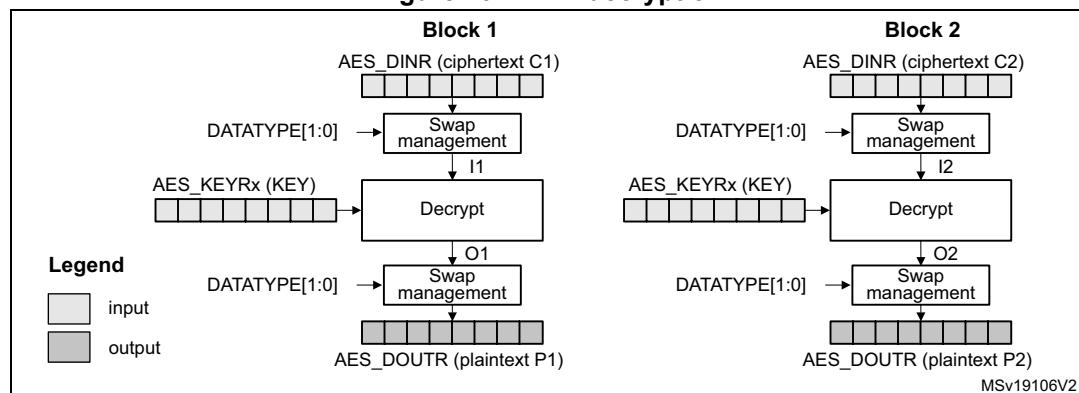
Figure 103. ECB encryption



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 Cx. The ECB encryption continues in this way until the last complete plaintext block is encrypted.

Figure 104 illustrates the electronic codebook (ECB) decryption.

Figure 104. ECB decryption

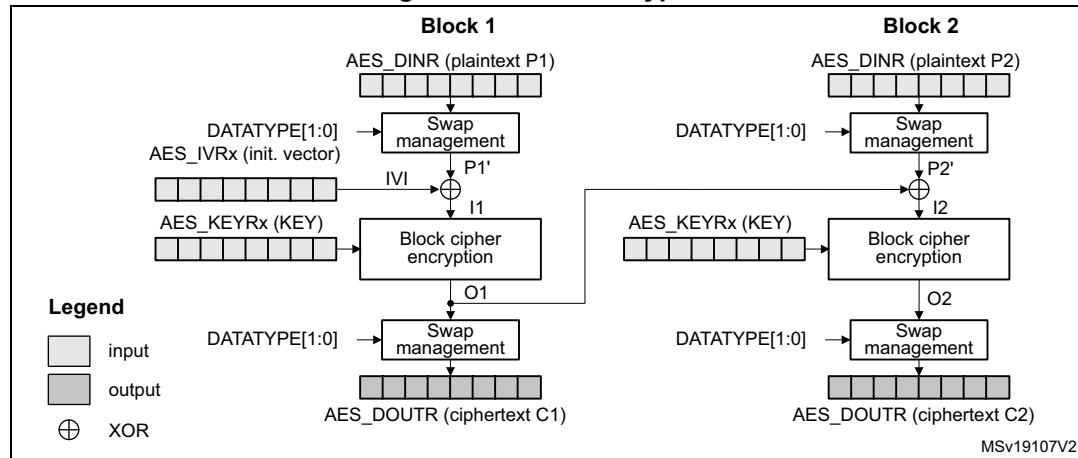


To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported 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 105](#) illustrates the cipher block chaining (CBC) encryption.

Figure 105. CBC encryption

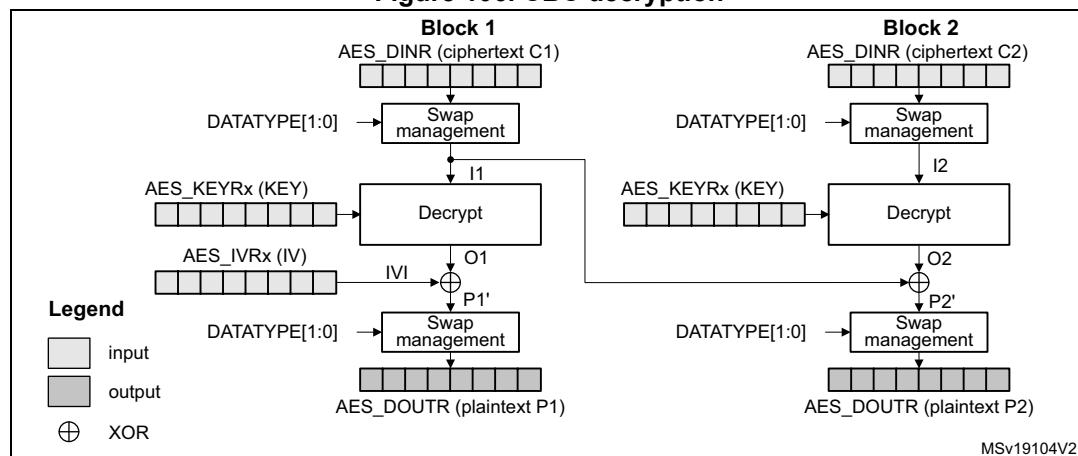


In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping ($P1'$), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the $I1$ input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block $O1$, 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 22.4.6: AES ciphertext stealing and data padding](#).

[Figure 106](#) illustrates the cipher block chaining (CBC) decryption.

Figure 106. 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 22.4.6: AES ciphertext stealing and data padding](#).

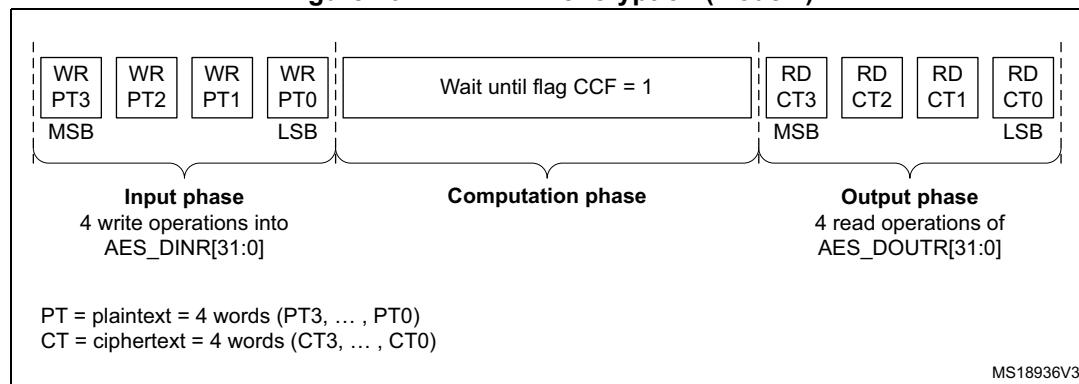
For more information on data swapping, refer to [Section 22.4.13: AES data registers and data swapping](#).

ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in [Section 22.4.4](#)):

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 107](#).
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 107](#). Then clear the CCF flag by setting the CCF bit of the AES_ICR register.
9. Repeat steps 6-7-8 to process all the blocks with the same encryption key.

Figure 107. ECB/CBC encryption (Mode 1)



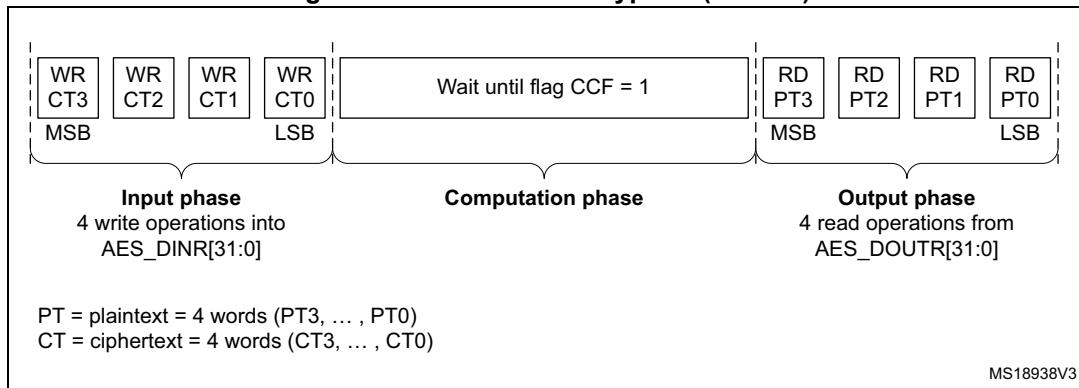
ECB/CBC decryption sequence

The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in [Section 22.4.4](#)).

1. Follow the steps described in [Section 22.4.5: 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 108](#).
 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 108](#). Then clear the CCF flag by setting the CCF bit of the AES_ICR register.
 8. Repeat steps [5-6-7](#) to process all the blocks encrypted with the same key.

Figure 108. 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 CCF bit of the AES_ICR 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).

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.
4. Prepare the decryption key as described in [Section 22.4.5: AES decryption round key preparation](#) (only required for ECB or CBC decryption).
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.

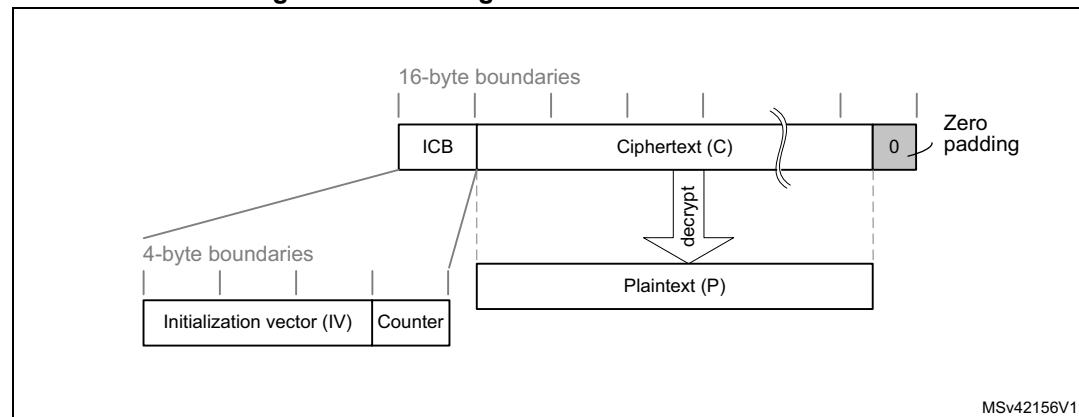
22.4.9 AES counter (CTR) mode

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 109](#).

Figure 109. 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 110](#) and [Figure 111](#) 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 110. CTR encryption

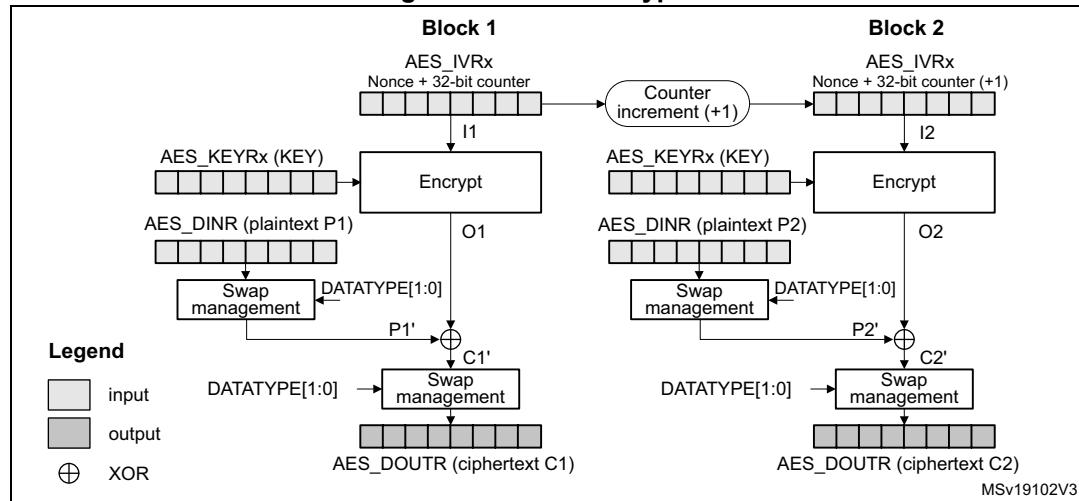
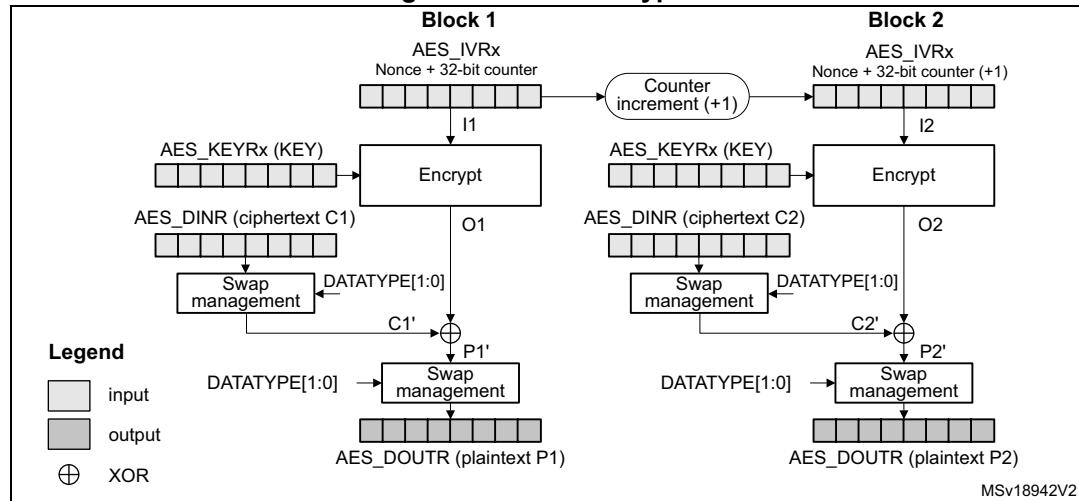


Figure 111. CTR decryption



In CTR mode, the cryptographic core output (also called keystream) O_x is XOR-ed with relevant input block (P_x' for encryption, C_x' for decryption), to produce the correct output block (C_x for encryption, P_x for decryption). Initialization vectors in AES must be initialized as shown in [Table 117](#).

Table 117. CTR mode initialization vector definition

| AES_IVR3[31:0] | AES_IVR2[31:0] | AES_IVR1[31:0] | AES_IVR0[31:0] |
|----------------|----------------|----------------|--------------------------------------|
| IVI[127:96] | IVI[95:64] | IVI[63:32] | IVI[31:0] 32-bit counter = 0x0001 |

Unlike in CBC mode that uses the AES_IVRx registers only once when processing the first data block, in CTR mode AES_IVRx registers are used for processing each data block, and the AES peripheral increments the counter bits of the initialization vector (leaving the nonce bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that is then XOR-ed with the plaintext (CTR encryption) or ciphertext (CTR decryption) input. In CTR mode, the MODE[1:0] bitfield setting 01 (key derivation) is forbidden and all the other settings default to encryption mode.

The sequence of events to perform an encryption or a decryption in CTR chaining mode:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select CTR chaining mode by setting to 010 the CHMOD[2:0] bitfield of the AES_CR register. Set MODE[1:0] bitfield to any value other than 01.
3. Initialize the AES_KEYRx registers, and load the AES_IVRx registers as described in [Table 117](#).
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 22.4.4: 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 22.4.8: AES basic chaining modes \(ECB, CBC\)](#).

Note: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.

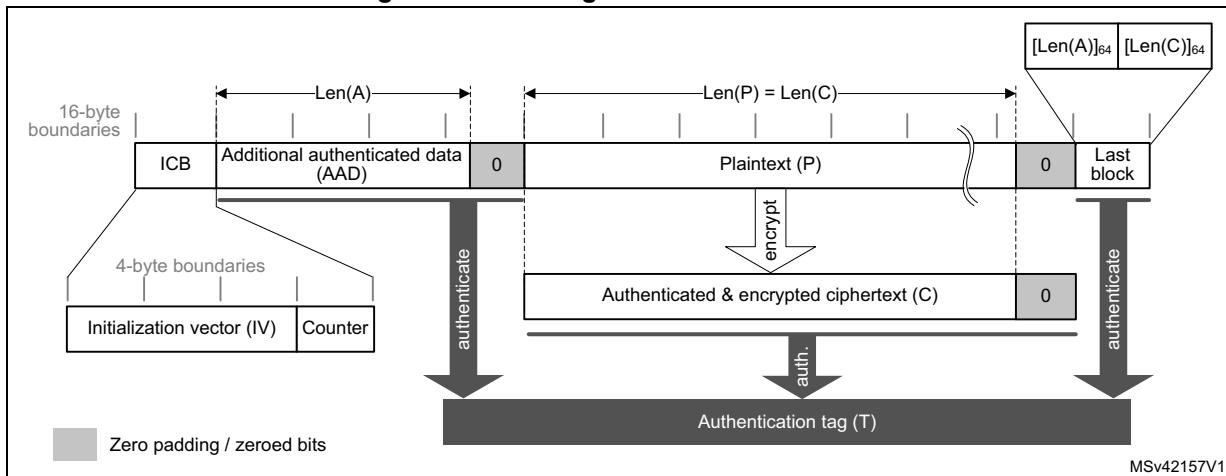
22.4.10 AES Galois/counter mode (GCM)

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 112](#).

Figure 112. 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 known 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 118](#).

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.

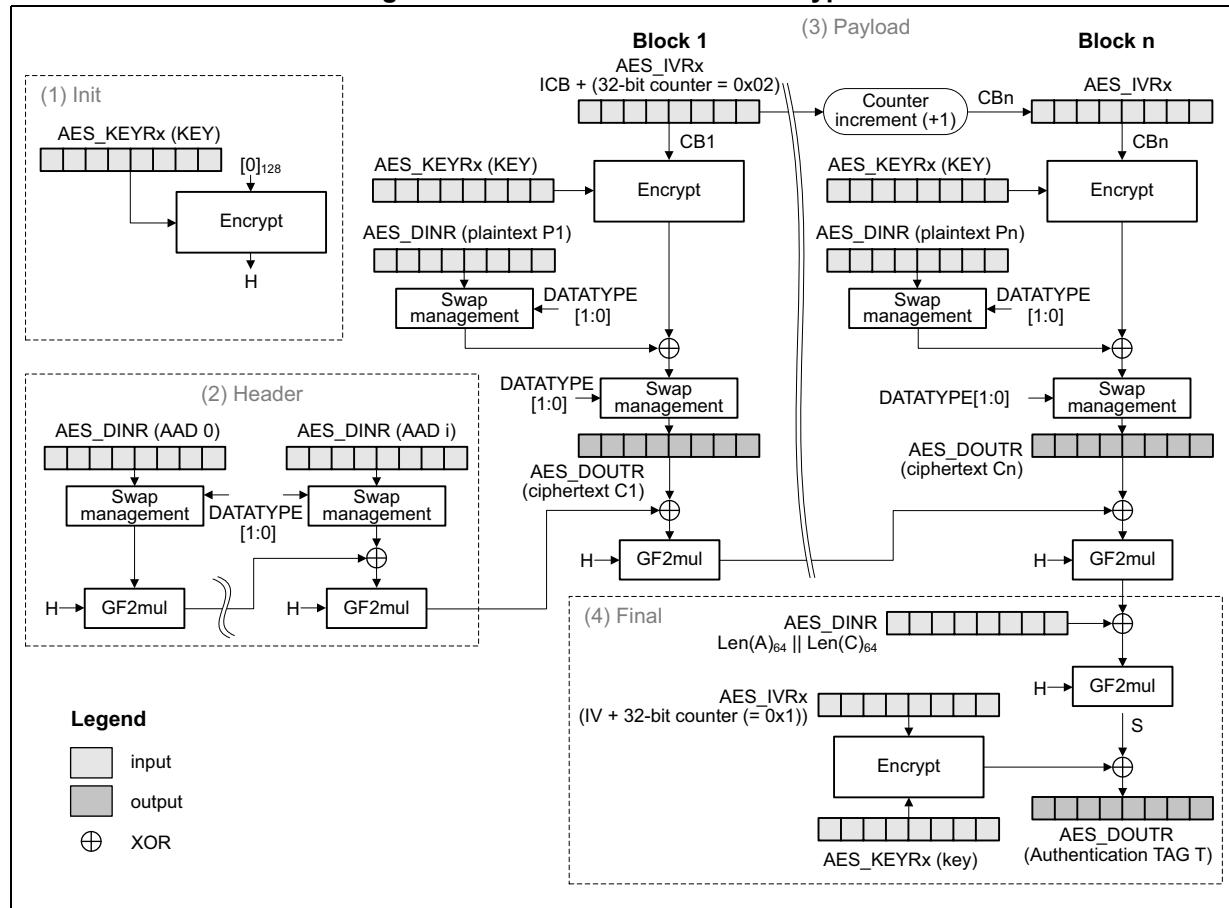
Table 118. GCM last block definition

| 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 113](#) 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 113. 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 119](#)).

Table 119. Initialization of AES_IVRx registers in GCM mode

| AES_IVR3[31:0] | AES_IVR2[31:0] | AES_IVR1[31:0] | AES_IVR0[31:0] |
|----------------|----------------|----------------|--------------------------------------|
| ICB[127:96] | ICB[95:64] | ICB[63:32] | ICB[31:0] 32-bit counter = 0x0002 |

Note: In this mode, the setting 01 of the MODE[1:0] bitfield (key derivation) is 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 119](#).
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 CCF bit of the AES_ICR 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 22.4.4: 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 22.4.4: AES procedure to perform a cipher operation on page 509](#), 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 118](#). 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 CCF bit of the AES_ICR 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 CCF bit of the AES_ICR 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.

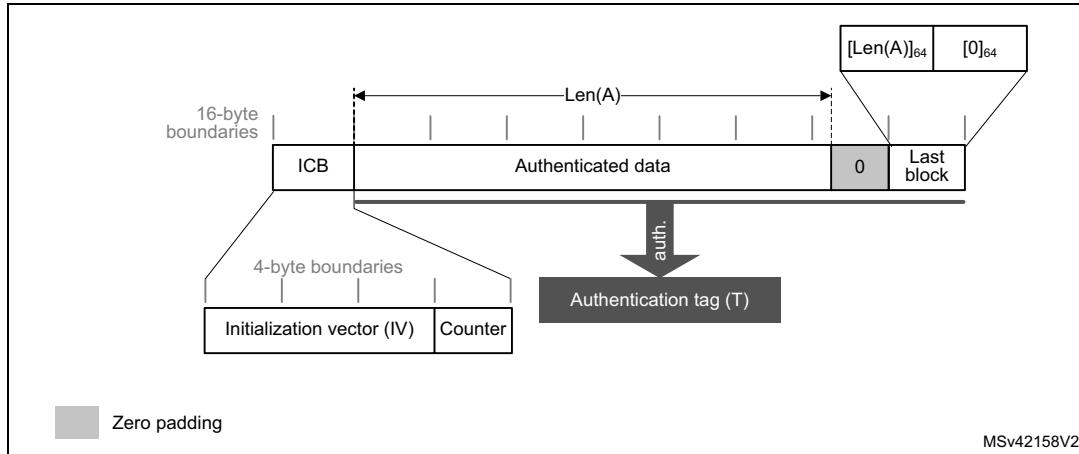
22.4.11 AES Galois message authentication code (GMAC)

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 114](#).

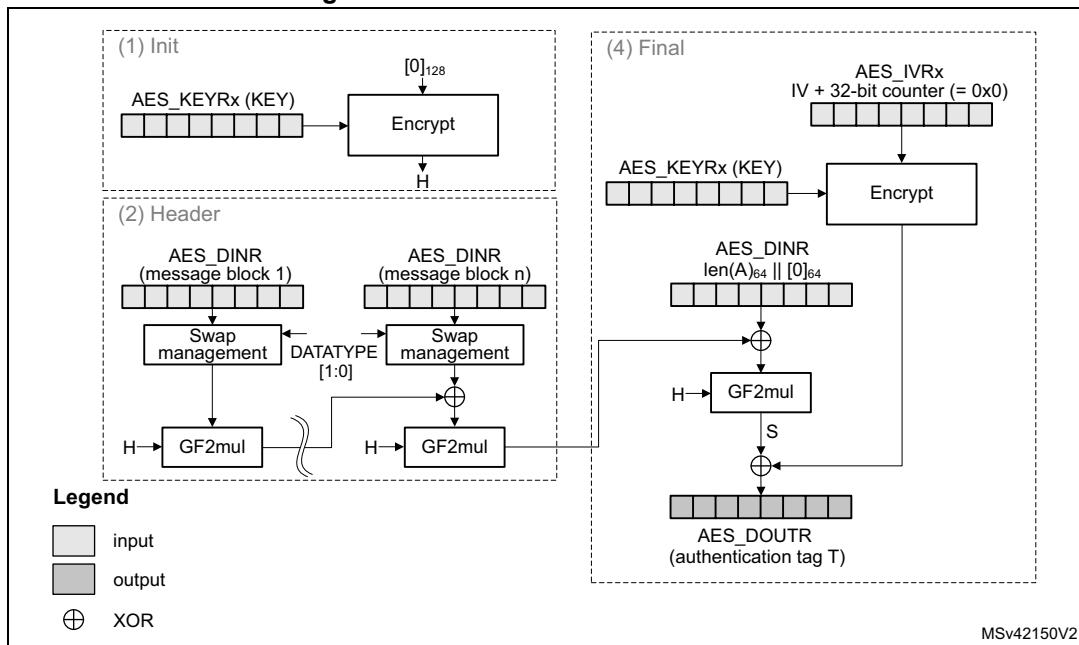
Figure 114. Message construction in GMAC mode



AES GMAC processing

[Figure 115](#) 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 115. 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.

22.4.12 AES counter with CBC-MAC (CCM)

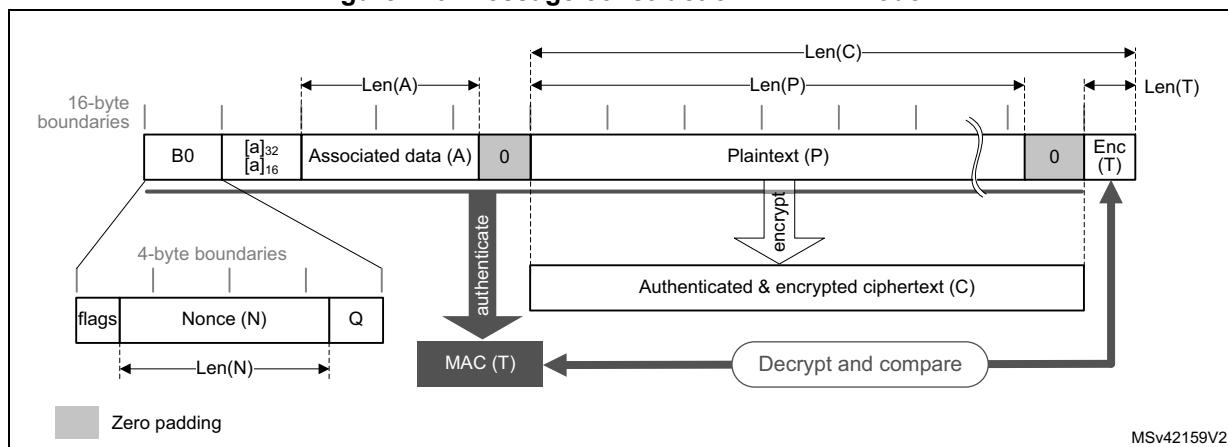
Overview

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 116](#).

Figure 116. 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 116](#)). 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 $0xff \parallel 0xfe \parallel [a]_{32}$, that is, on six bytes.
- If $2^{32} < a < 2^{64}$, then it is encoded as $0xff \parallel 0xff \parallel [a]_{64}$, 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 116](#)).
- **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):

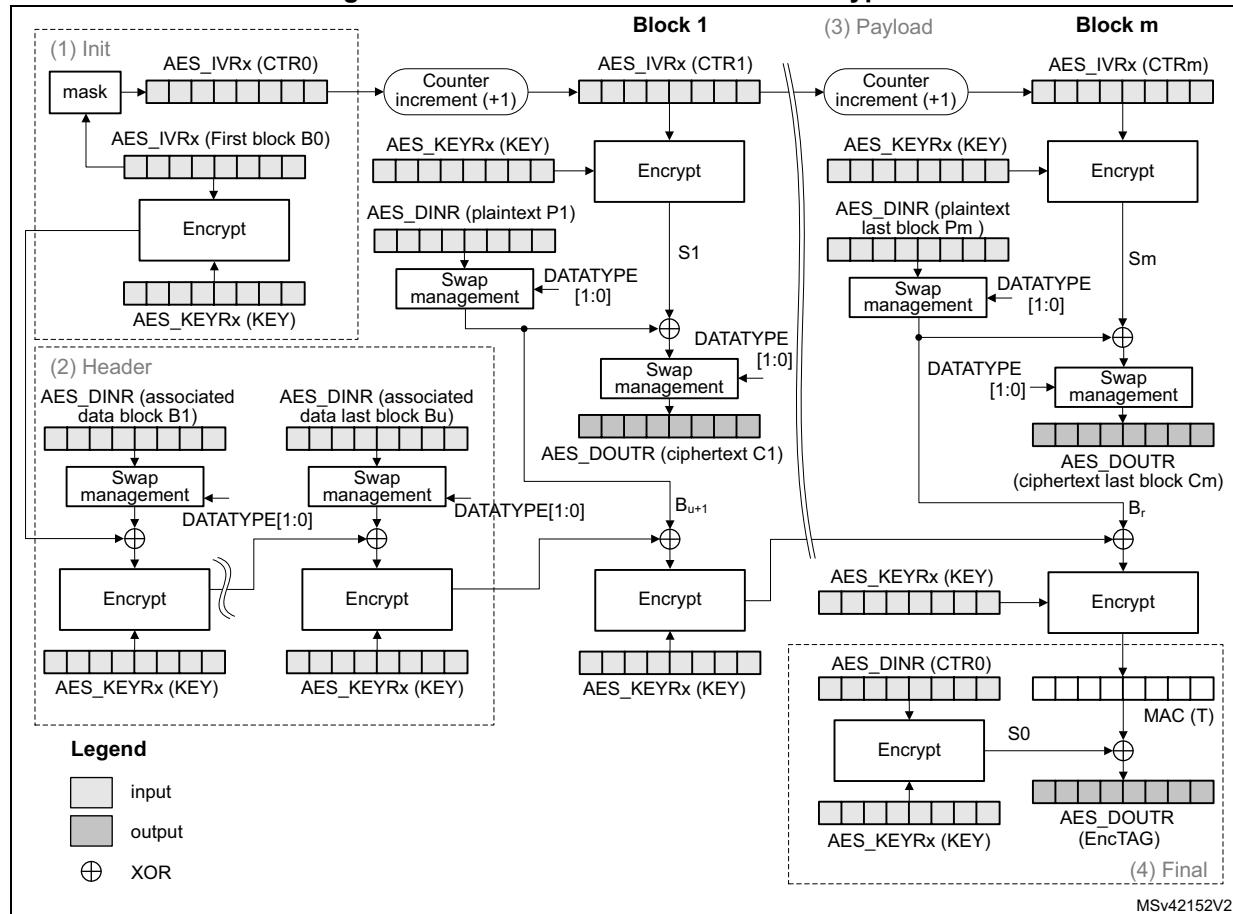
N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)
A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)
P: 20212223 (Len(P) = 32 bits or 4 bytes)
T: 6084341B (Len(T) = 32 bits or t = 4)
B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000
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 117](#) 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 117. 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 120](#) 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 120. 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 setting 01 of the MODE[1:0] bitfield (key derivation) is 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 120](#).
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 CCF bit of the AES_ICR 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 22.4.4: 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 GCMFH[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 22.4.4: AES procedure to perform a cipher operation on page 509](#), 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 GCMFH[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 CCF bit of the AES_ICR 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 CCF bit of the AES_ICR 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.

22.4.13 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 22.4.16: AES DMA interface](#).

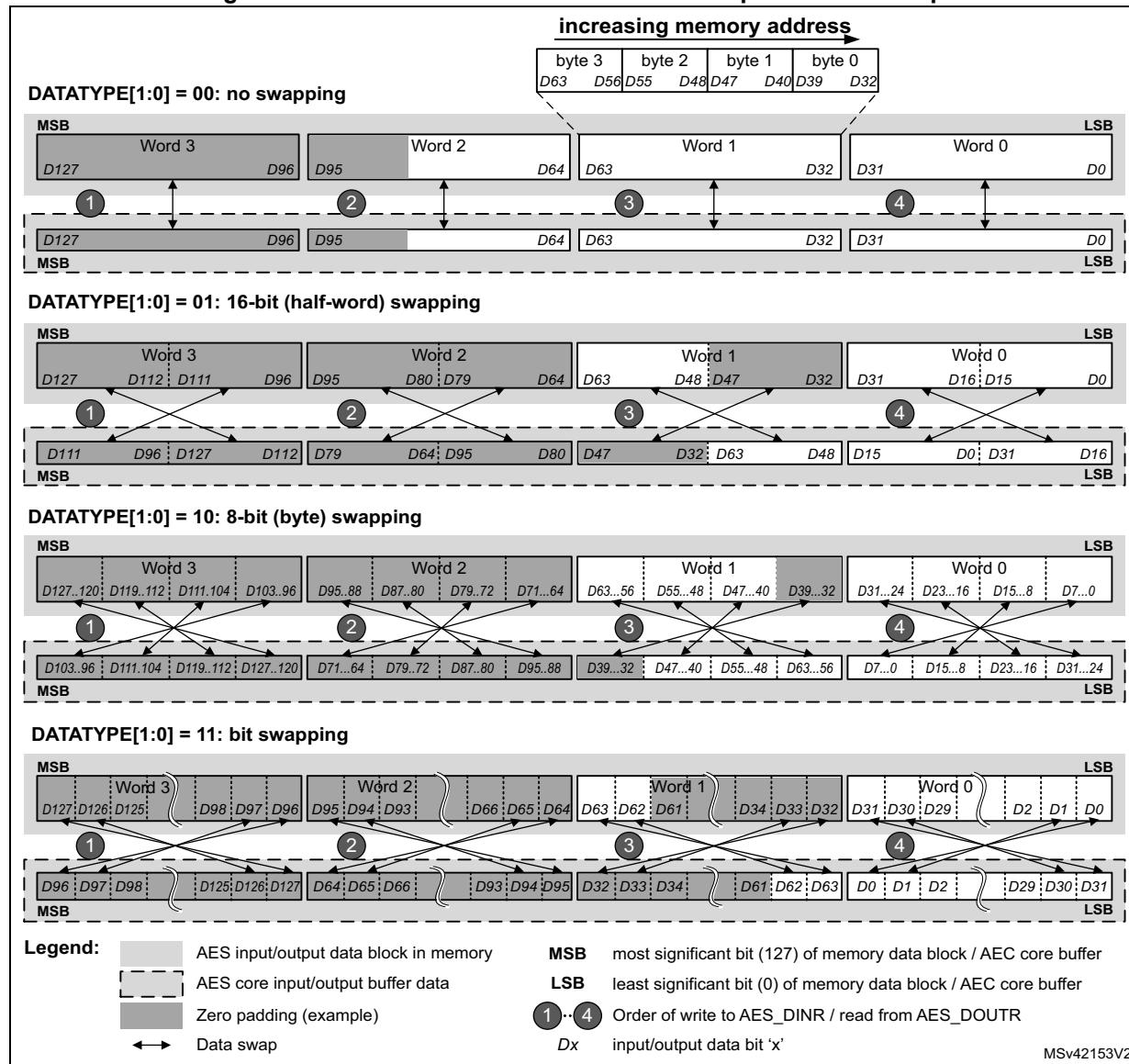
Data swapping

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 118* 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 118. 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 118](#) 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

22.4.14 AES key registers

The AES_KEYRx write-only registers store the encryption or decryption key bitfield KEY[127:0] or KEY[255:0]. The data to write to each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address (reads are not allowed for security reason).

The key is spread over eight registers as shown in [Table 121](#).

Table 121. 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.

22.4.15 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.

22.4.16 AES DMA interface

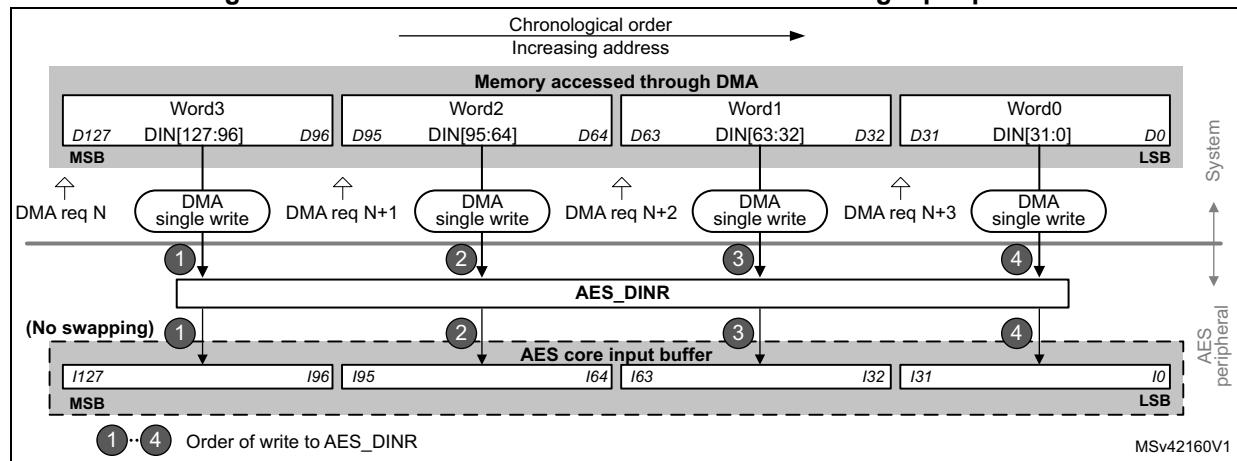
The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the AES_CR register.

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 119](#).

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 22.4.4: AES procedure to perform a cipher operation](#).

Figure 119. DMA transfer of a 128-bit data block during input phase

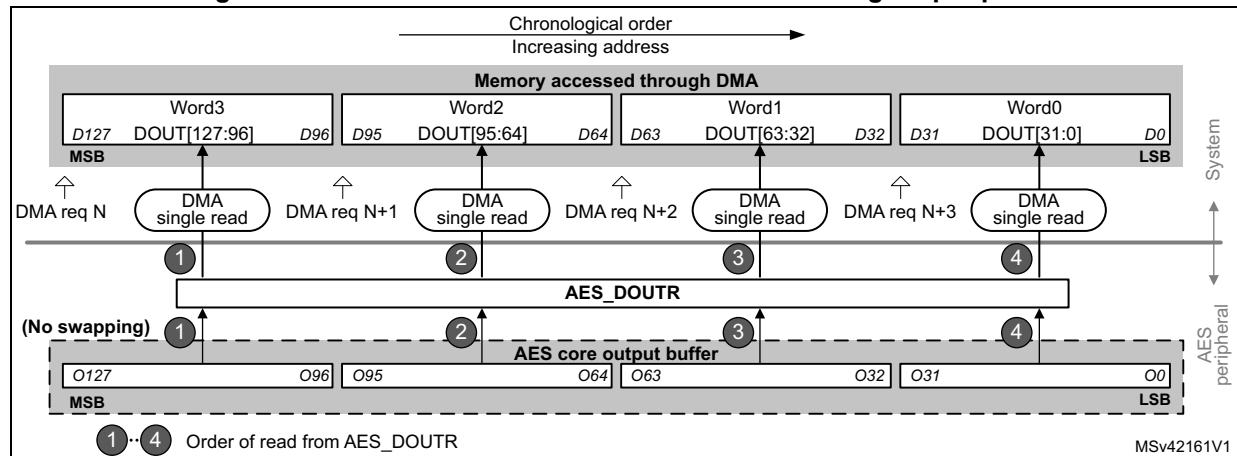


Data output using DMA

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 120](#).

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Figure 120. 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 22.4.4: AES procedure to perform a cipher operation](#), subsection [Data append](#), for details.

22.4.17 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_IER register is set. For more details, refer to [Section 22.5: AES interrupts](#).

The RDERR flag is cleared by setting the ERRIE bit of the AES_ICR 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_IER register is set. For more details, refer to [Section 22.5: AES interrupts](#).

The WRERR flag is cleared by setting the ERRC bit of the AES_ICR register.

22.5 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_IER register, and cleared by setting the corresponding bit of the AES_ICR register.

The status of each can be read from the AES_SR register.

[Table 122](#) gives a summary of the interrupt sources, their event flags and enable bits.

Table 122. AES interrupt requests

| Interrupt acronym | AES interrupt event | Event flag | Enable bit | Interrupt clear method |
|-------------------|----------------------------|------------|------------|-------------------------|
| AES | computation completed flag | CCF | CCFIE | set CCF ⁽¹⁾ |
| | read error flag | RDERR | ERRIE | set ERRC ⁽¹⁾ |
| | write error flag | WRERR | | |

1. Bit of the AES_ICR register.

22.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

Table 123. 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 |
| 256-bit | Mode 1: Encryption | ECB, CBC, CTR | 75 |
| | Mode 2: Key derivation | - | 82 |
| | Mode 3: Decryption | ECB, CBC, CTR | 75 |

Table 124. 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).

22.7 AES registers

22.7.1 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 | CCF | 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).

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).

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 CCF: 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: 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 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 22.4.13: 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.

22.7.2 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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res | BUSY | WRERR | RDERR |
| | | | | | | | | | | | r | r | r | r | CCF |

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_ICR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES_ICR 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_ICR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES_ICR 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 CCF bit of the AES_ICR 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.

22.7.3 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): ciphertext

The data swap operation is described in [Section 22.4.13: AES data registers and data swapping on page 532](#).

22.7.4 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
- **Mode 3** (decryption): plaintext

The data swap operation is described in [Section 22.4.13: AES data registers and data swapping on page 532](#).

22.7.5 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] | | | | | | | | | | | | | | | |
| 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]**: Cryptographic key, bits [31:0]

This write-only 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): 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.

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 22.4.14: AES key registers on page 534](#) for more details.

22.7.6 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] | | | | | | | | | | | | | | | |
| 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[47:32] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

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.

22.7.7 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] | | | | | | | | | | | | | | | |
| 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[79:64] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

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.

22.7.8 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] | | | | | | | | | | | | | | | |
| 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[111:96] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

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.

22.7.9 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 22.4.15: AES initialization vector registers on page 534](#) 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

22.7.10 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.

22.7.11 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.

22.7.12 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.

22.7.13 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] | | | | | | | | | | | | | | | |
| 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[143:128] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

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.

22.7.14 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] | | | | | | | | | | | | | | | |
| 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[175:160] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

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.

22.7.15 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] | | | | | | | | | | | | | | | |
| 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[207:192] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

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.

22.7.16 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] | | | | | | | | | | | | | | | |
| 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[239:224] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

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).*

22.7.17 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.

22.7.18 AES register map

Table 125. AES register map and reset values

Table 125. AES register map and reset values (continued)

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|---|
| 0x044 | AES_SUSP1R | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x048 | AES_SUSP2R | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x04C | AES_SUSP3R | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x050 | AES_SUSP4R | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x054 | AES_SUSP5R | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x058 | AES_SUSP6R | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x05C | AES_SUSP7R | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | |
| 0x060-0x3FF | Reserved | Res. | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

23 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.

23.1 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.

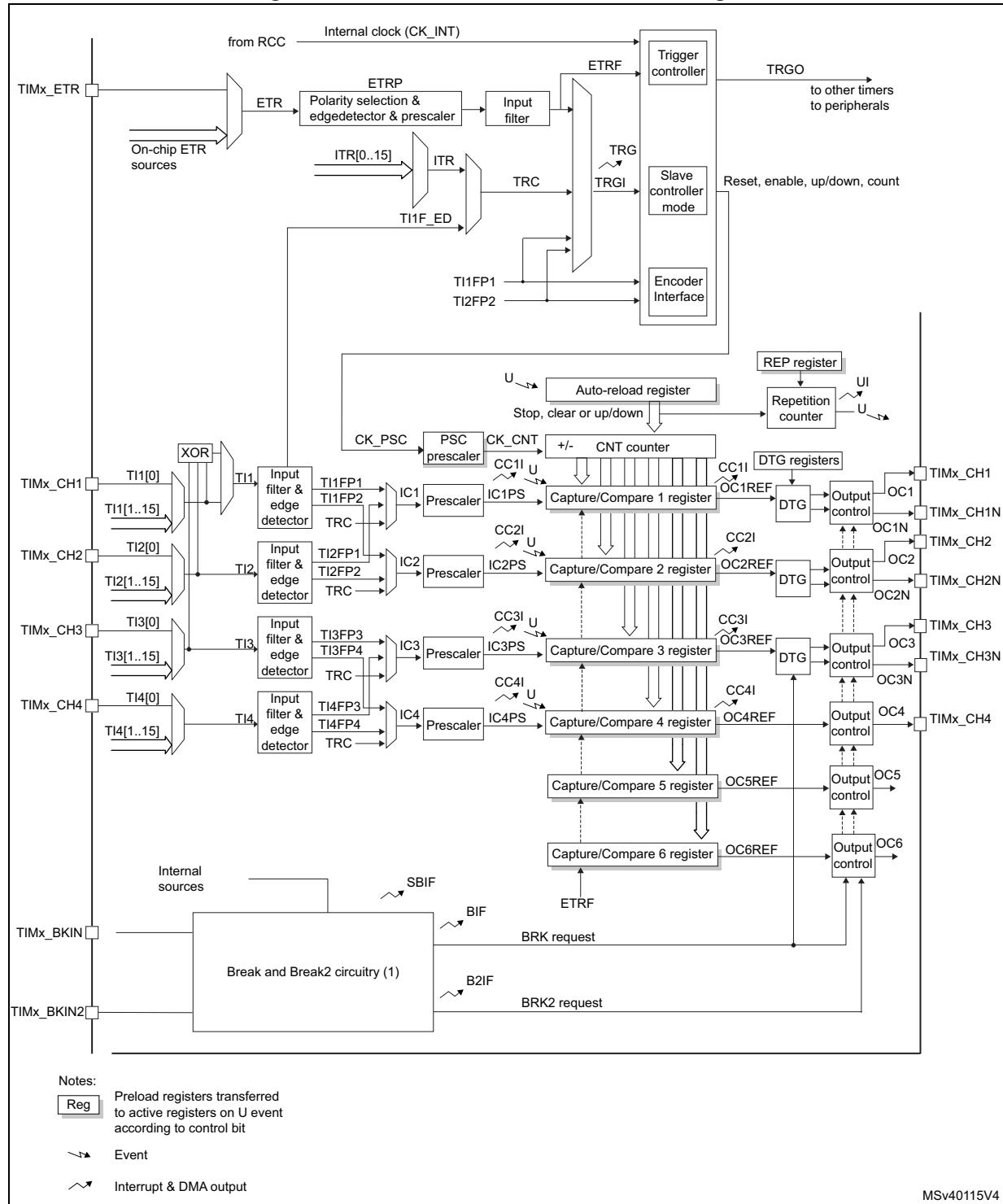
The advanced-control (TIM1) and general-purpose (TMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 23.3.26: Timer synchronization](#).

23.2 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 121. 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 5.2.10: Clock security system \(CSS\)](#)
 - A PVD output
 - SRAM parity error signal
 - Cortex®-M0+ LOCKUP (Hardfault) output
 - COMPx output, x = 1,2.

23.3 TIM1 functional description

23.3.1 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.

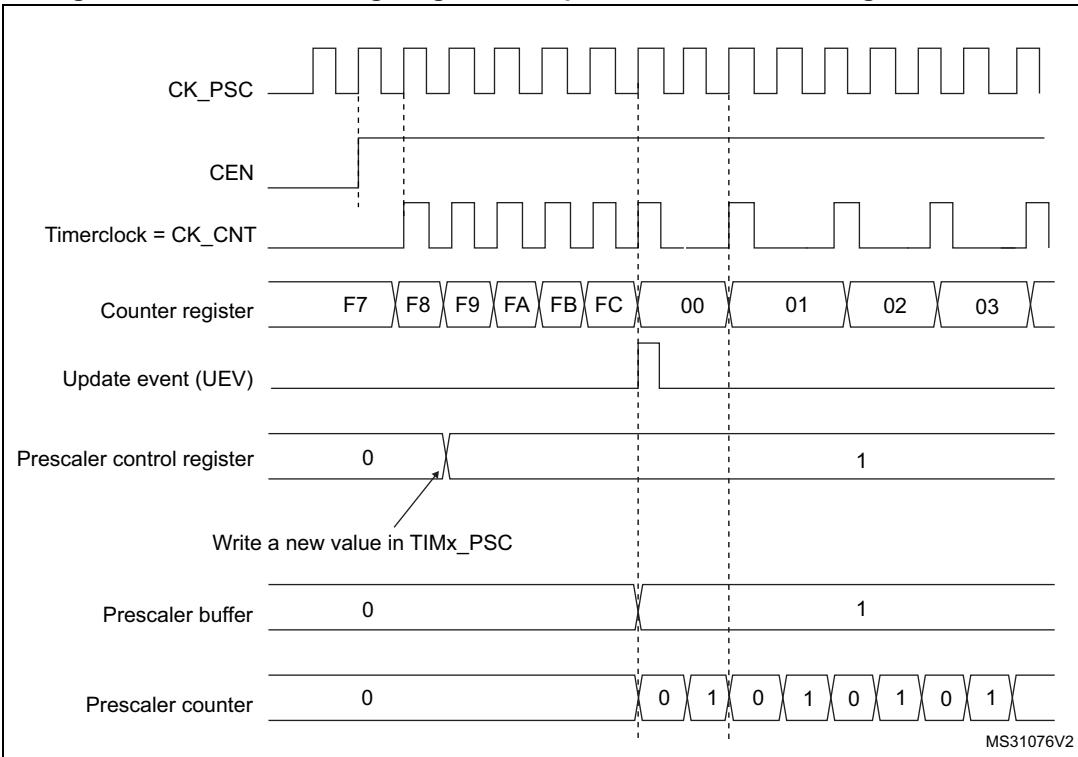
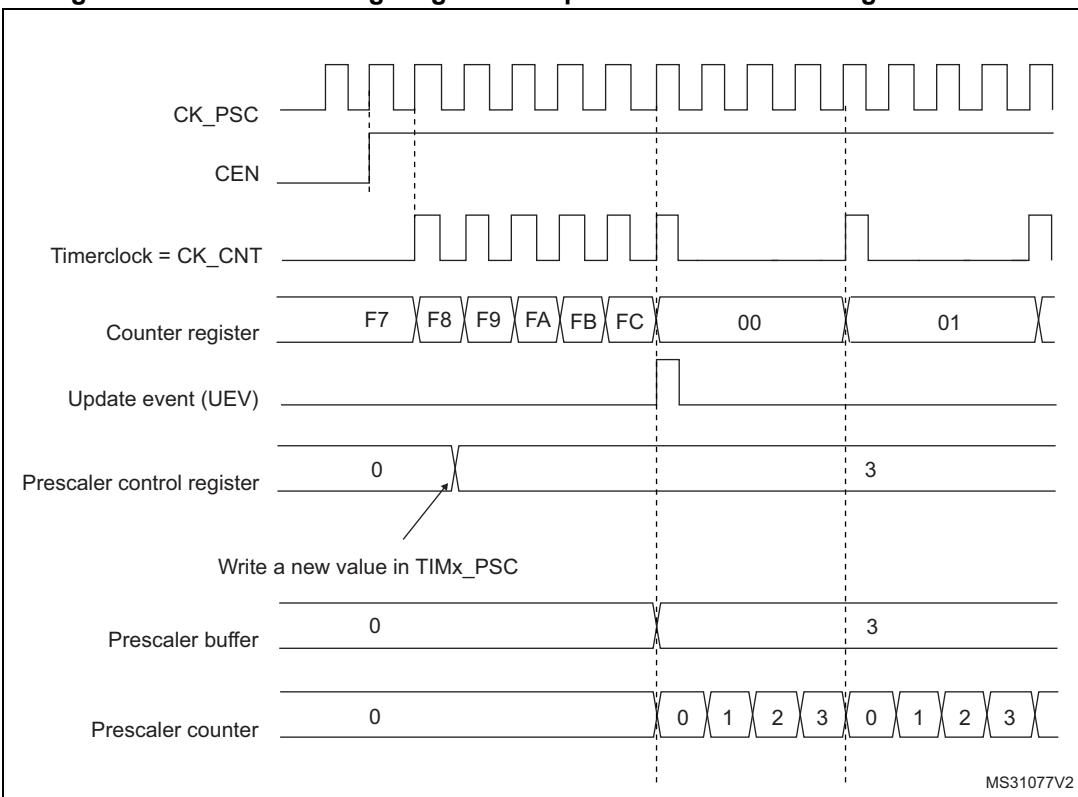
The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

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 122 and *Figure 123* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 122. Counter timing diagram with prescaler division change from 1 to 2**Figure 123. Counter timing diagram with prescaler division change from 1 to 4**

23.3.2 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.

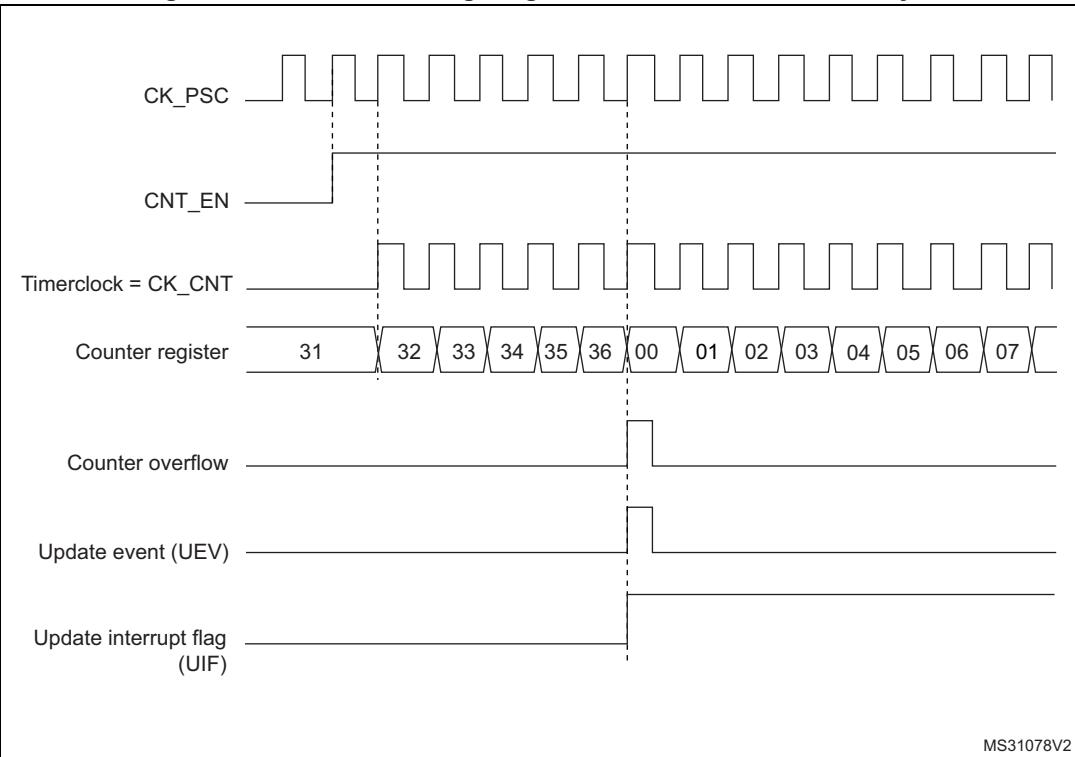
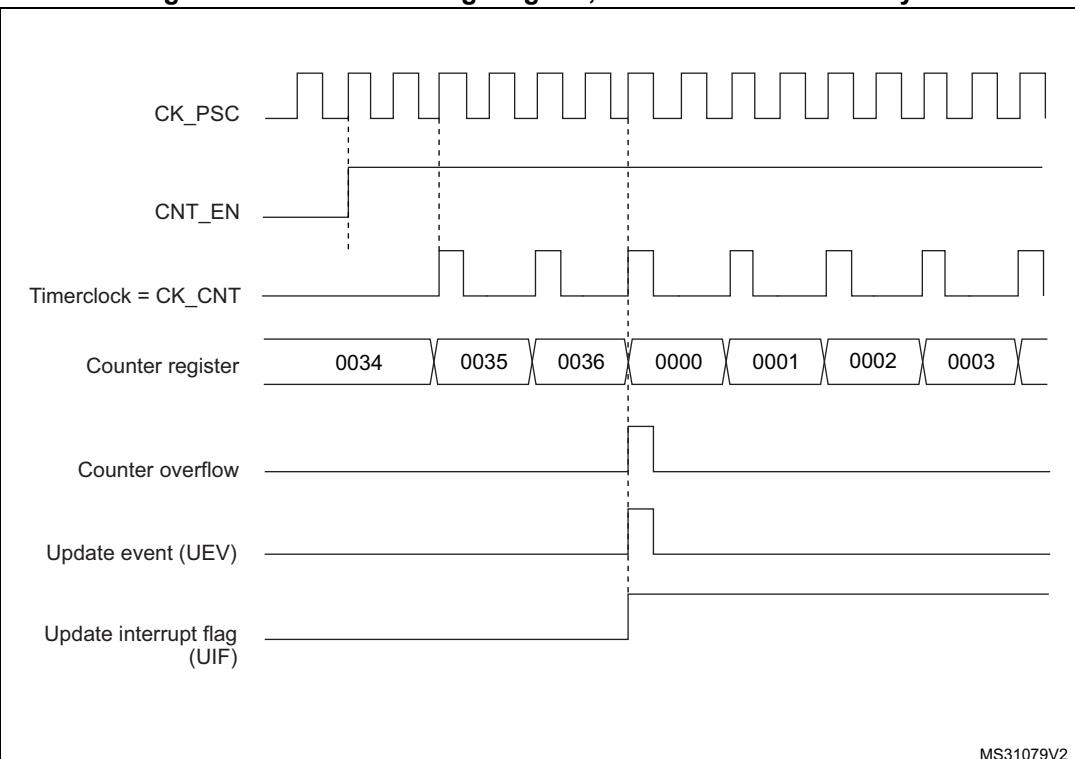
Figure 124. Counter timing diagram, internal clock divided by 1**Figure 125. Counter timing diagram, internal clock divided by 2**

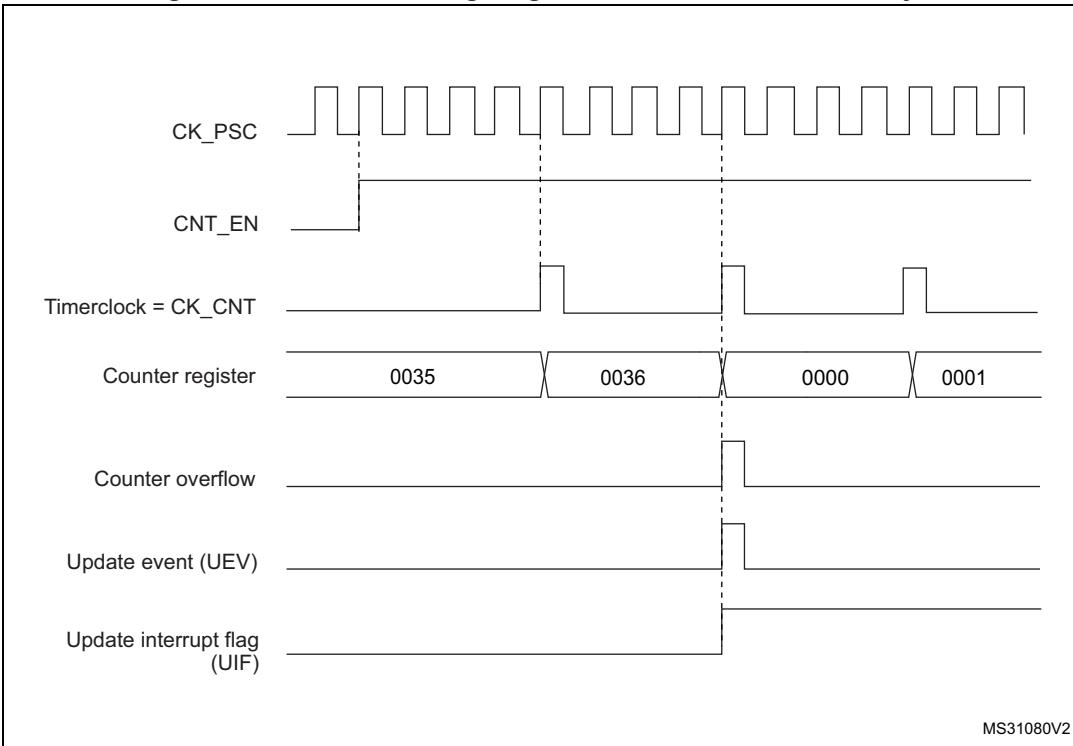
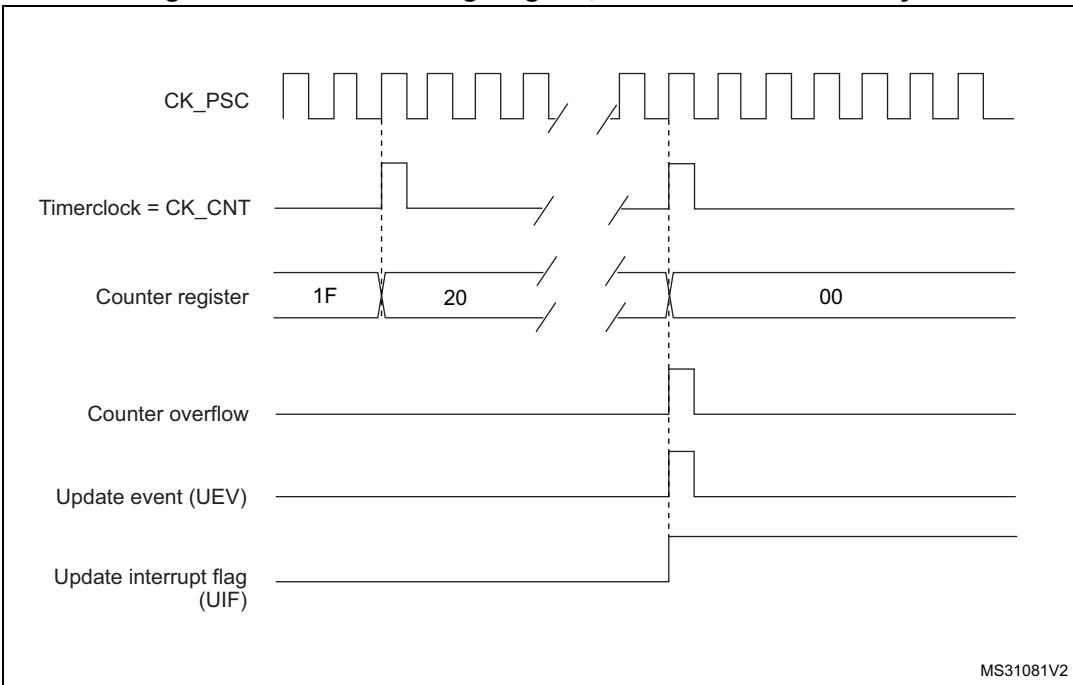
Figure 126. Counter timing diagram, internal clock divided by 4**Figure 127. Counter timing diagram, internal clock divided by N**

Figure 128. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

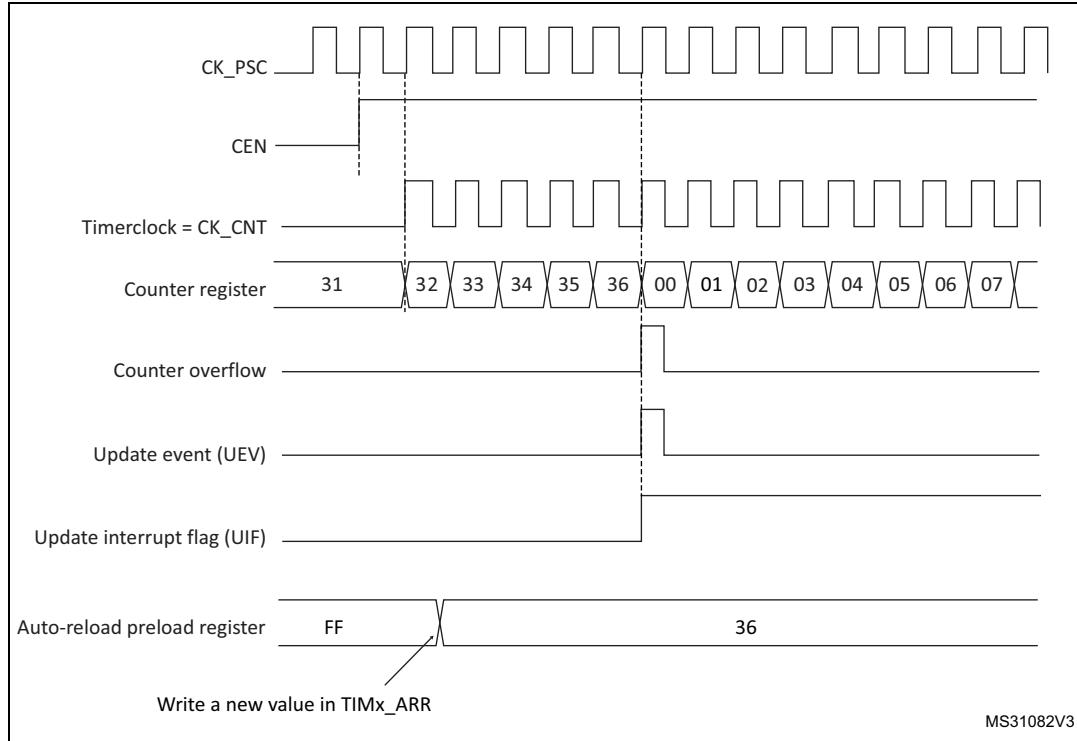
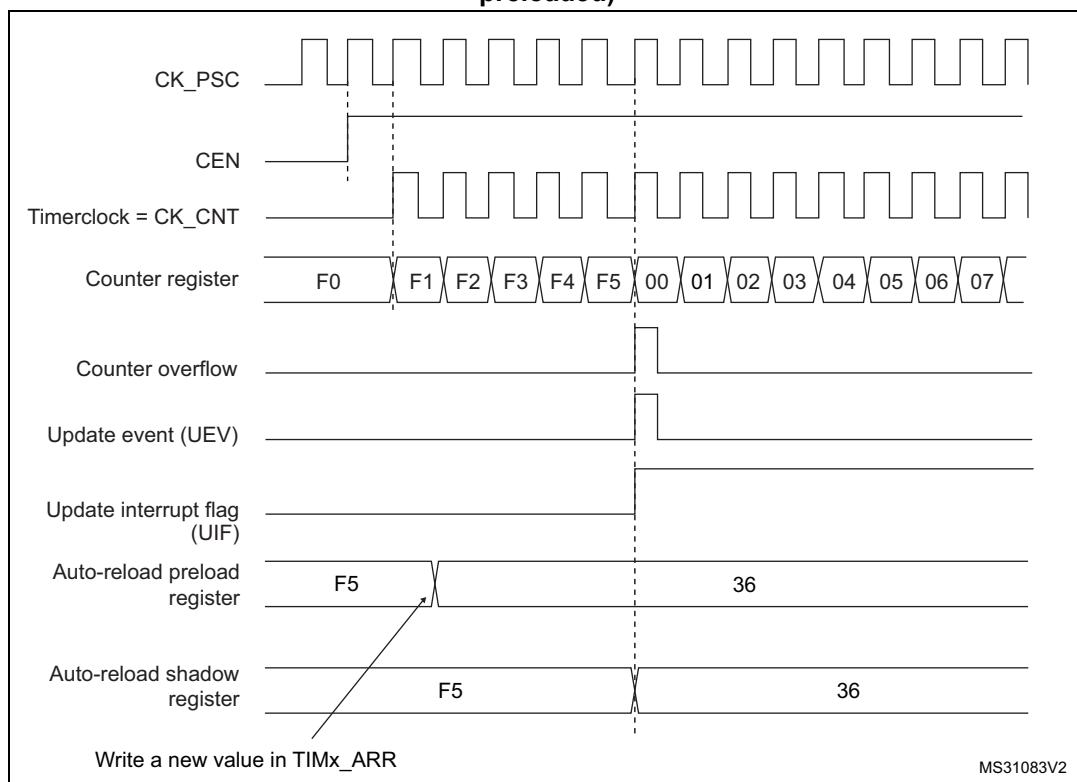


Figure 129. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



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.

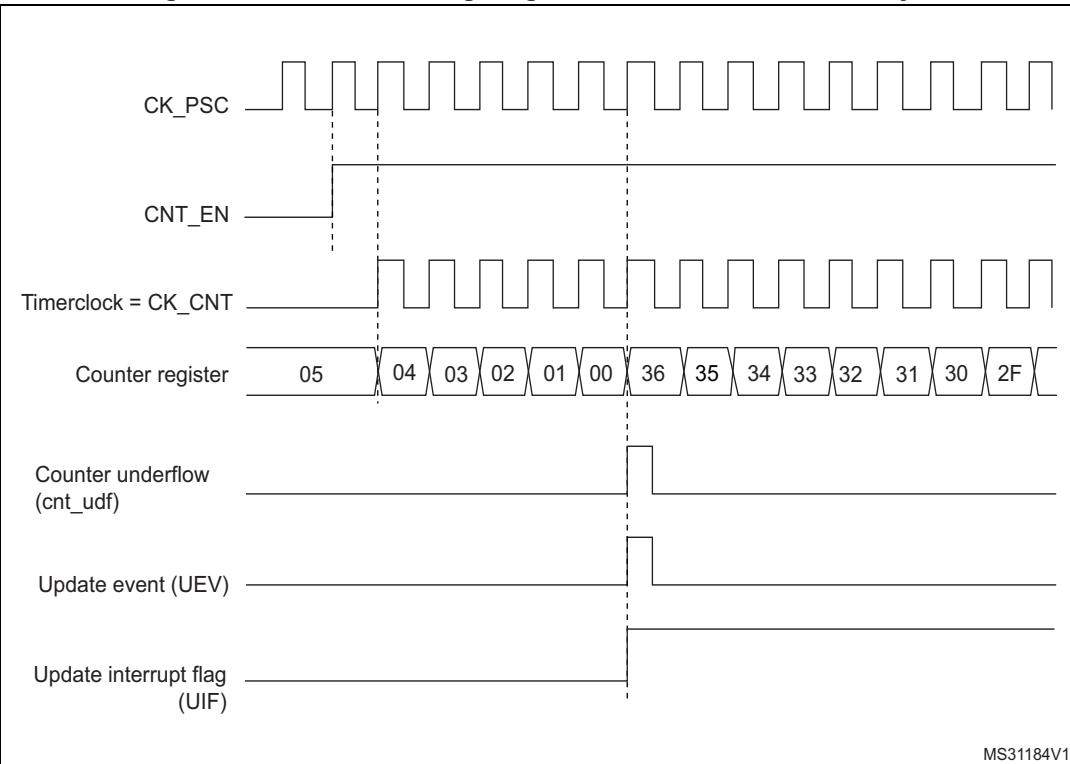
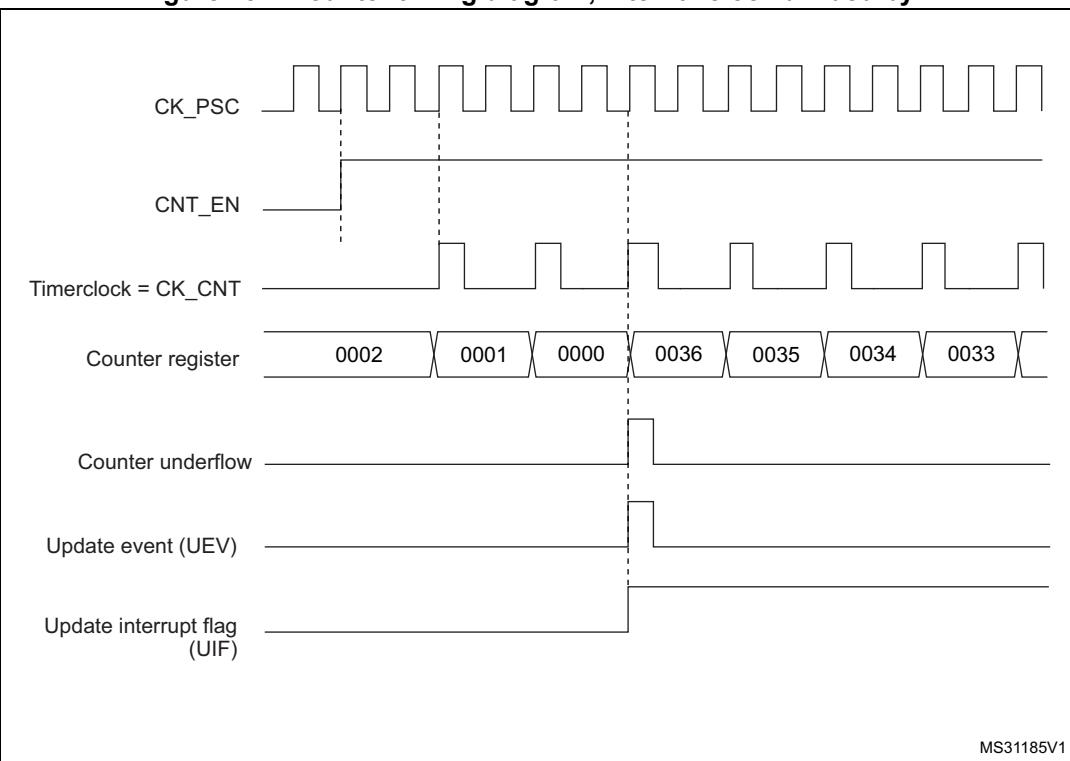
Figure 130. Counter timing diagram, internal clock divided by 1**Figure 131. Counter timing diagram, internal clock divided by 2**

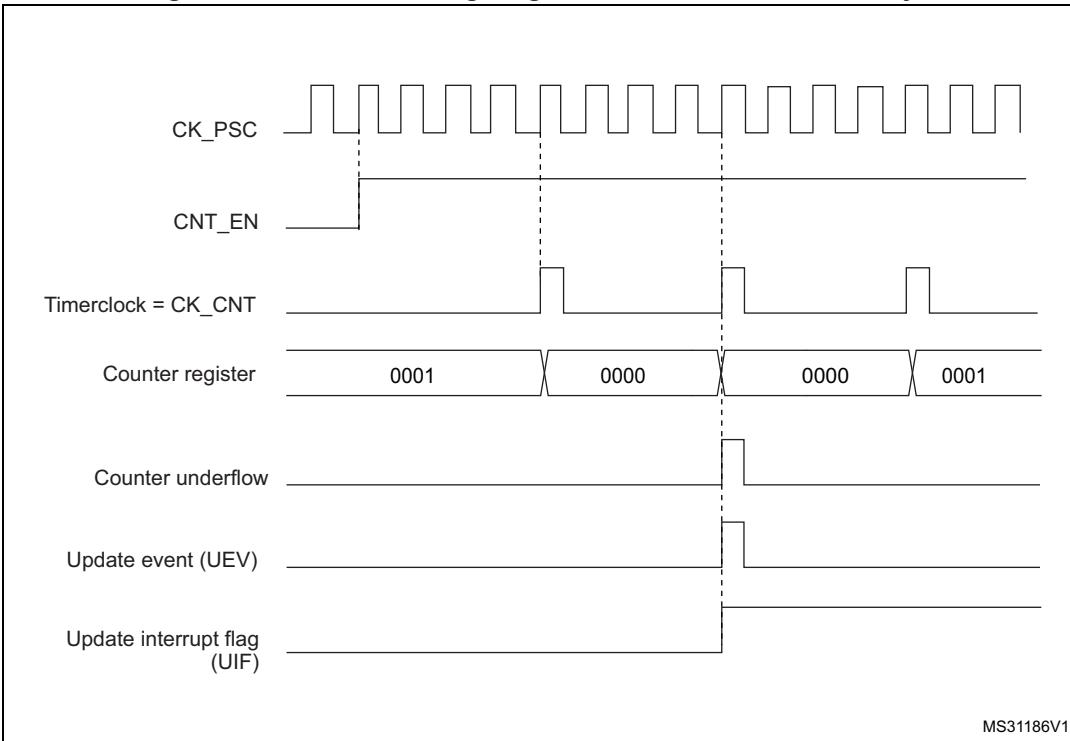
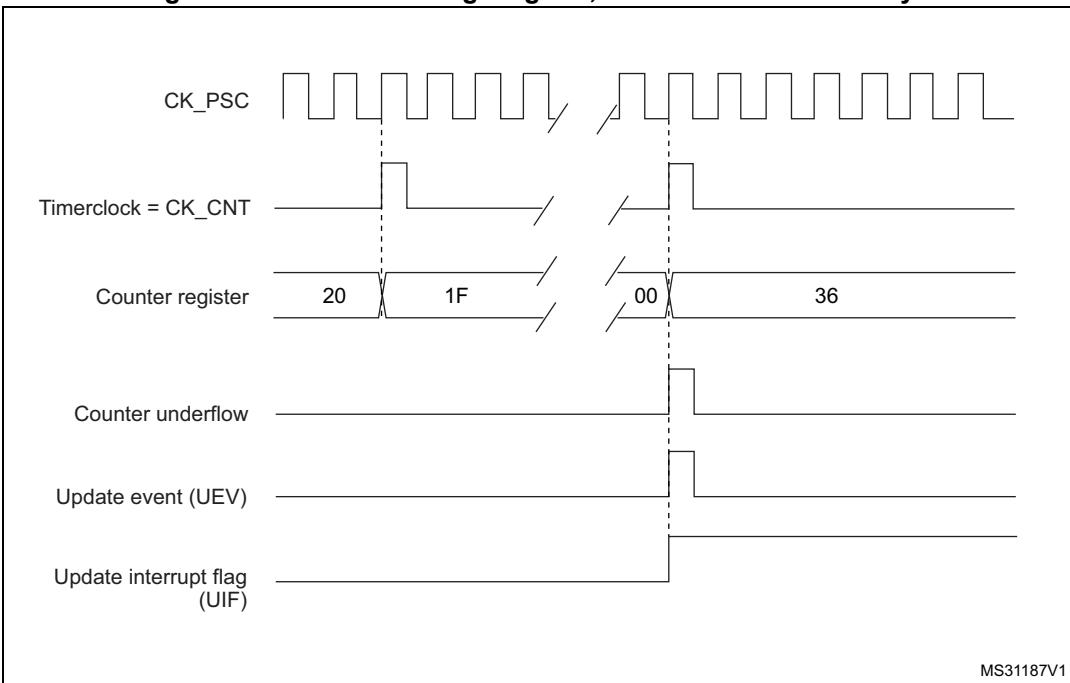
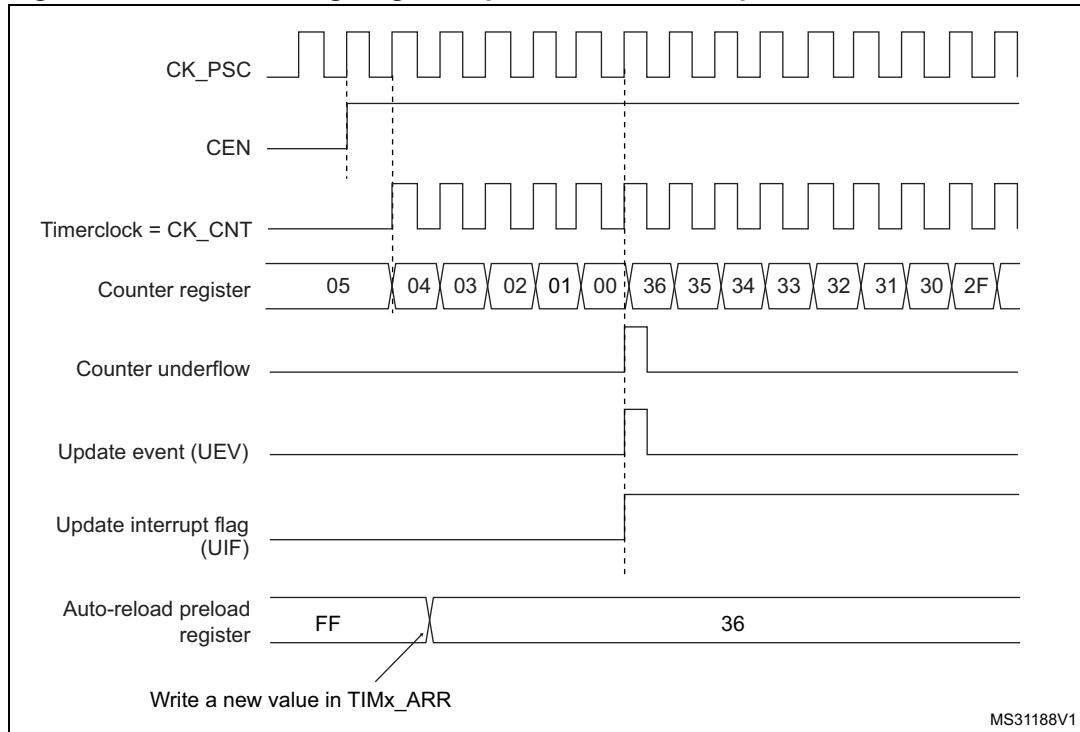
Figure 132. Counter timing diagram, internal clock divided by 4**Figure 133. Counter timing diagram, internal clock divided by N**

Figure 134. 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") or 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

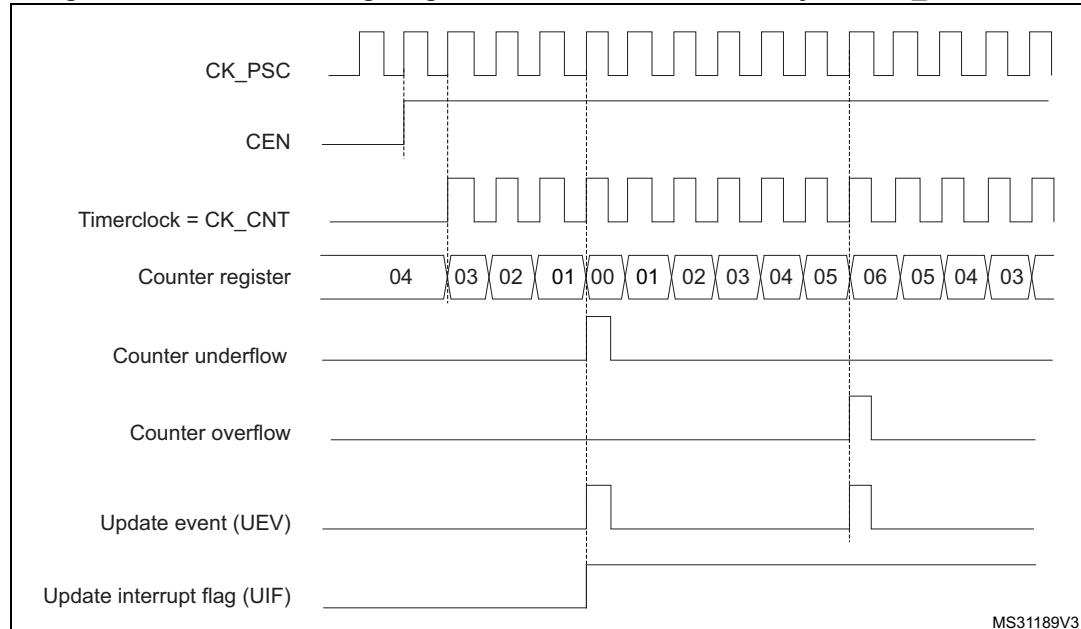
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

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 135. 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 23.4: TIM1 registers](#)).

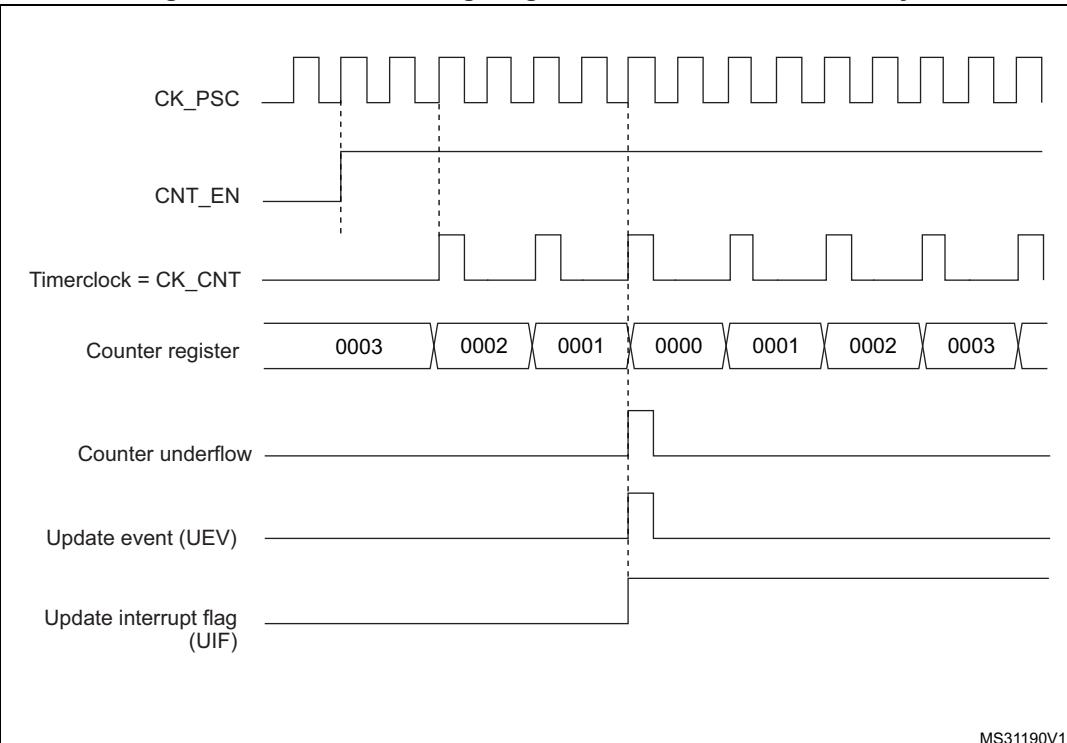
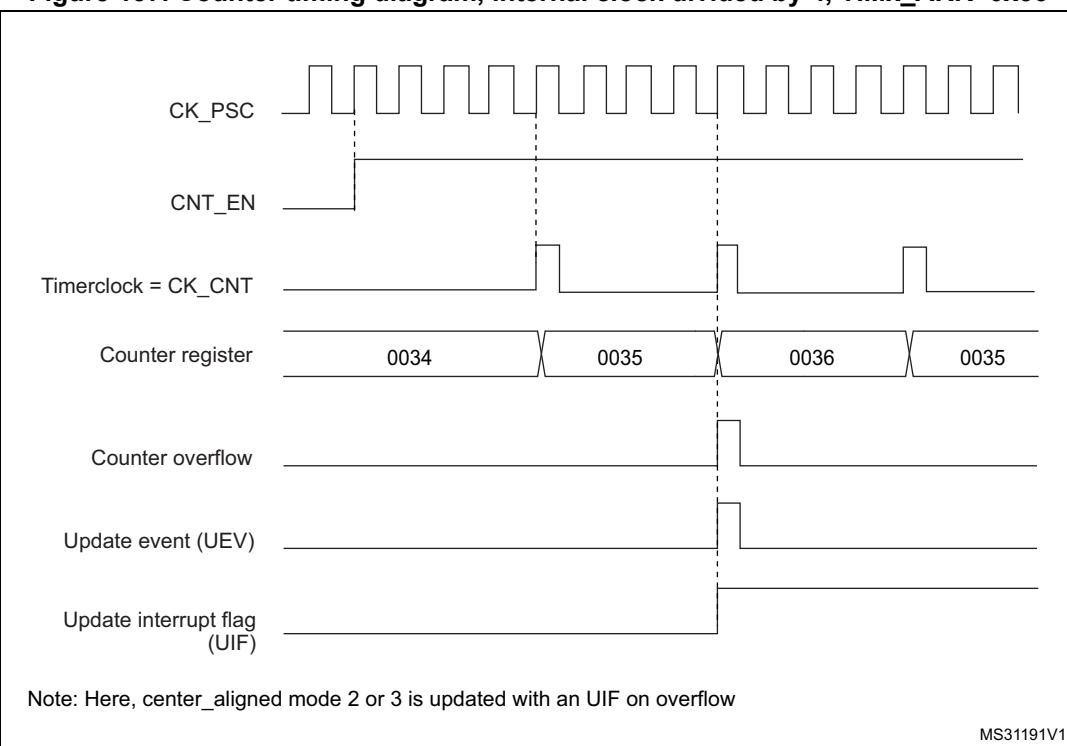
Figure 136. Counter timing diagram, internal clock divided by 2**Figure 137. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36**

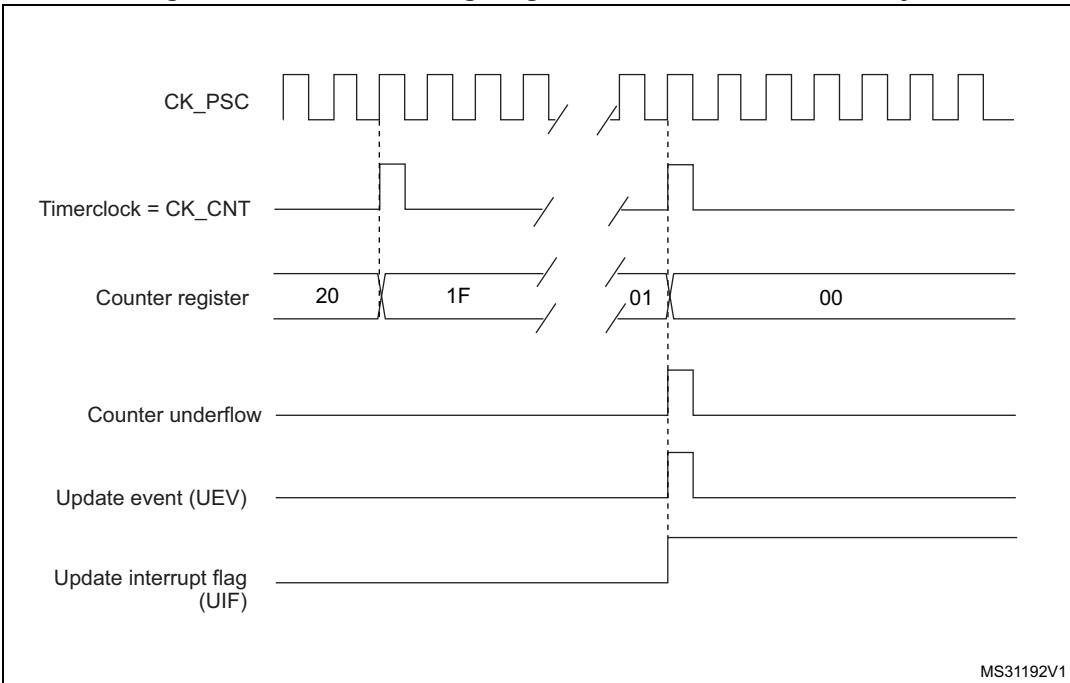
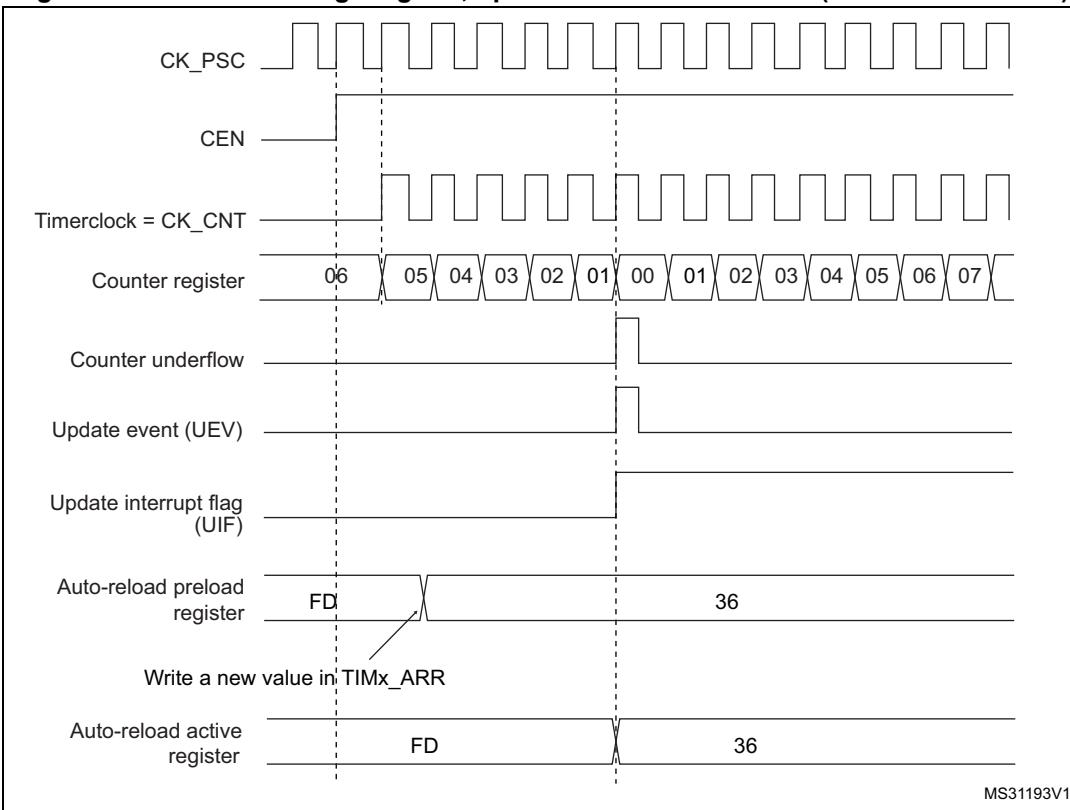
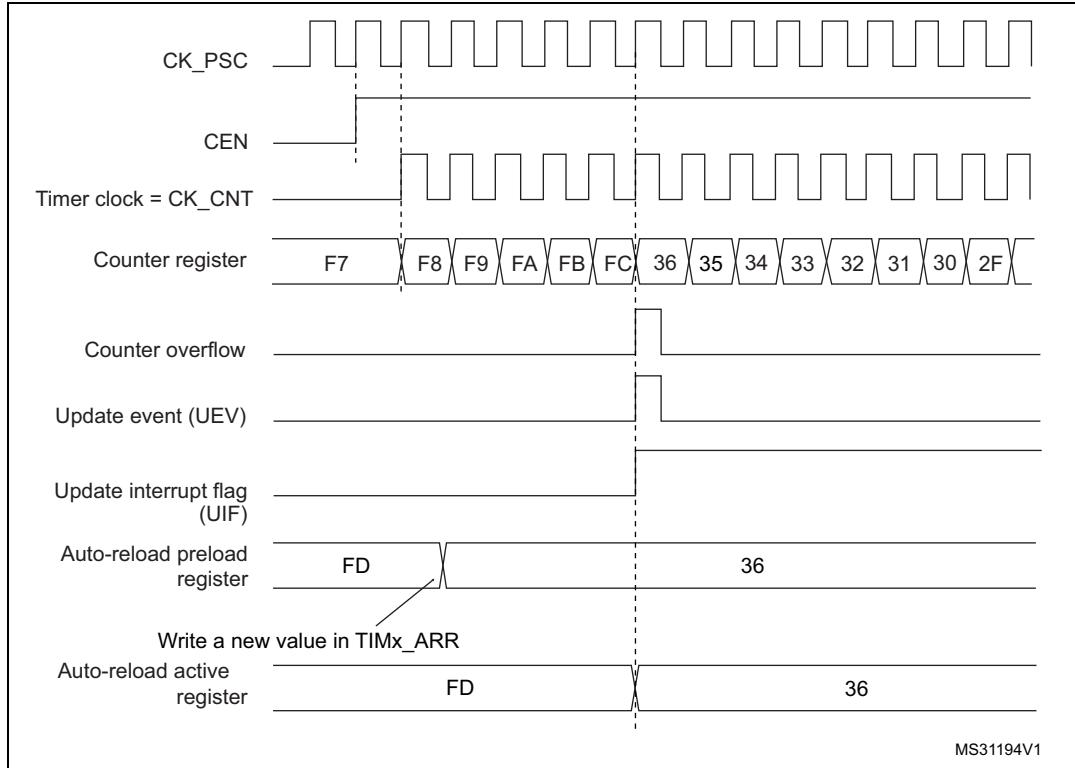
Figure 138. Counter timing diagram, internal clock divided by N**Figure 139. Counter timing diagram, update event with ARPE=1 (counter underflow)**

Figure 140. Counter timing diagram, Update event with ARPE=1 (counter overflow)

23.3.3 Repetition counter

[Section 23.3.1: 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:

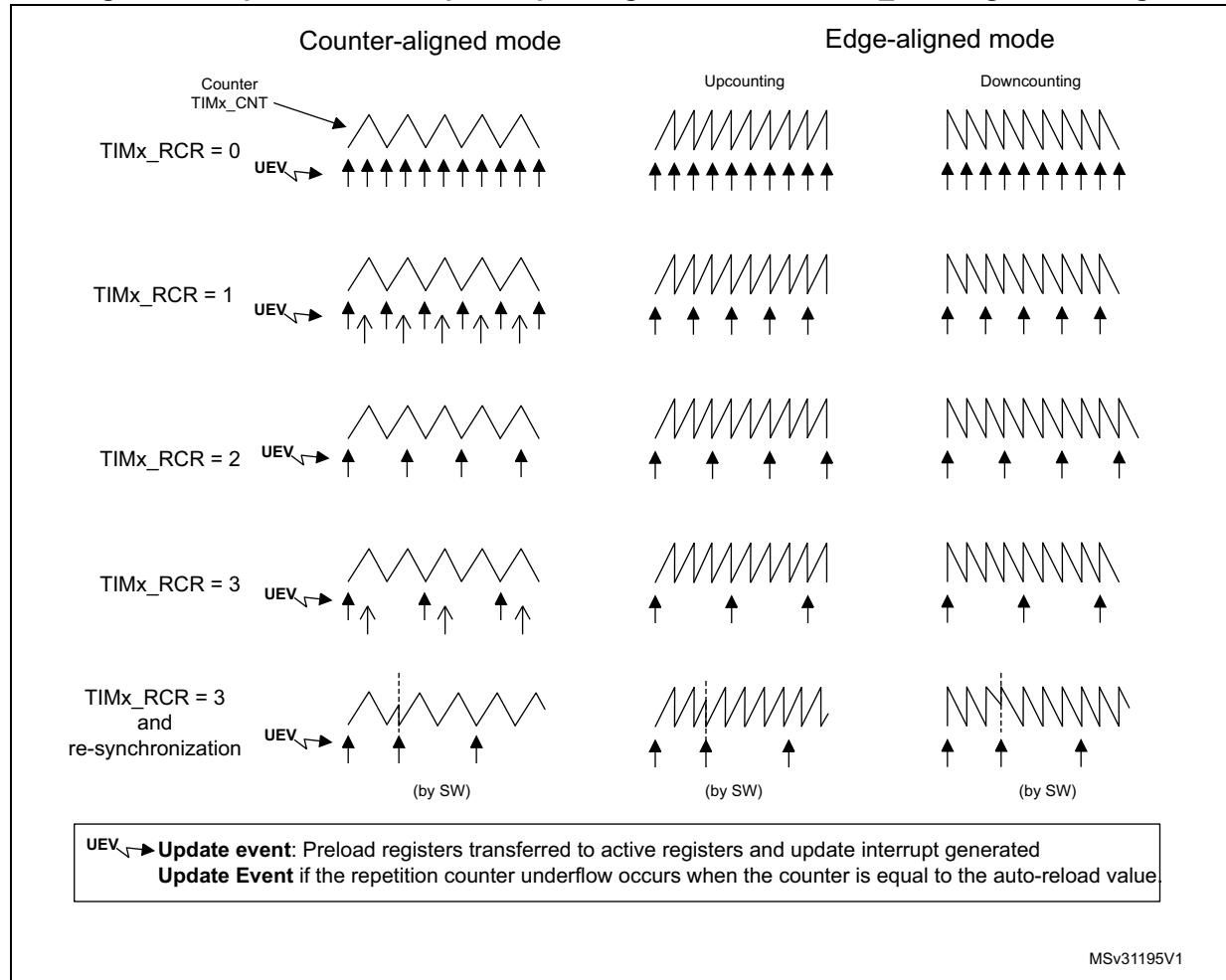
- At each counter overflow in upcounting mode,
 - At each counter underflow in downcounting mode,
 - At each counter overflow and at each counter underflow in center-aligned mode.
- Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2 \times T_{ck}$, 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 141](#)). 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 141. Update rate examples depending on mode and TIMx_RCR register settings



MSv31195V1

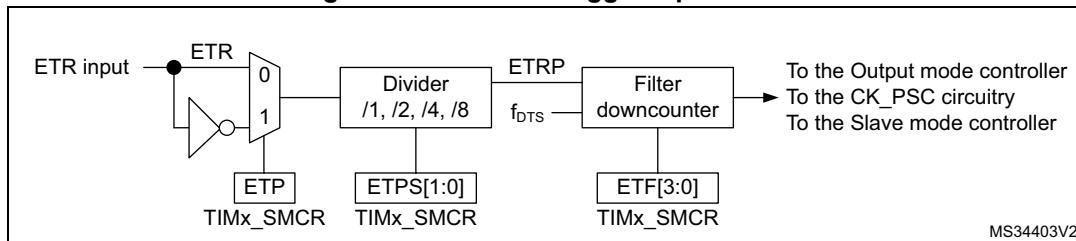
23.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see [Section 23.3.5](#))
- trigger for the slave mode (see [Section 23.3.26](#))
- PWM reset input for cycle-by-cycle current regulation (see [Section 23.3.7](#))

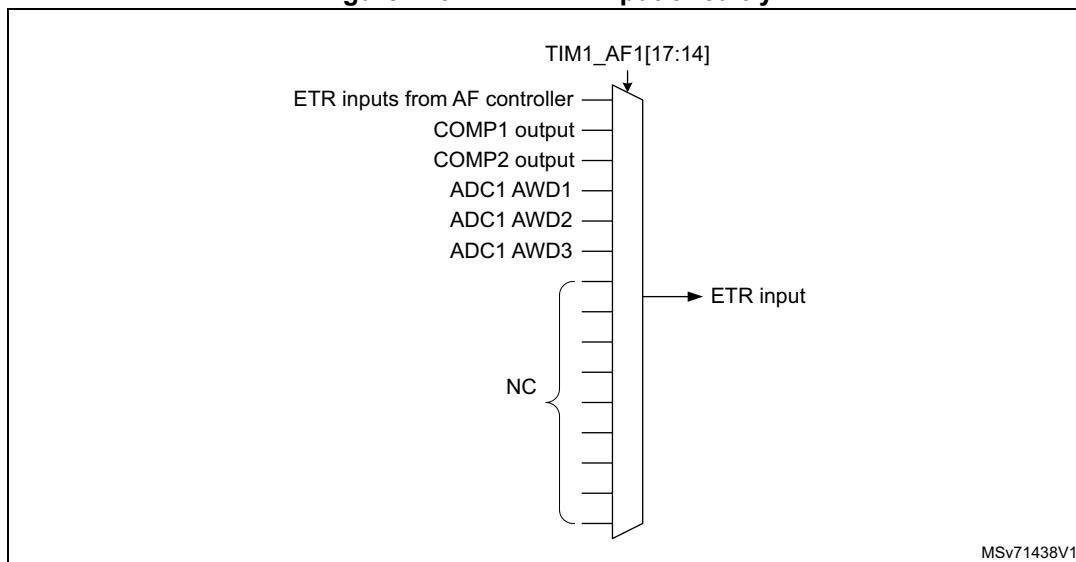
[Figure 142](#) below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

Figure 142. External trigger input block



The ETR input comes from multiple sources: input pins (default configuration), comparator outputs and analog watchdogs. The selection is done with the ETRSEL[3:0] bitfield.

Figure 143. TIM1 ETR input circuitry



23.3.5 Clock selection

The counter clock can be provided by the following clock sources:

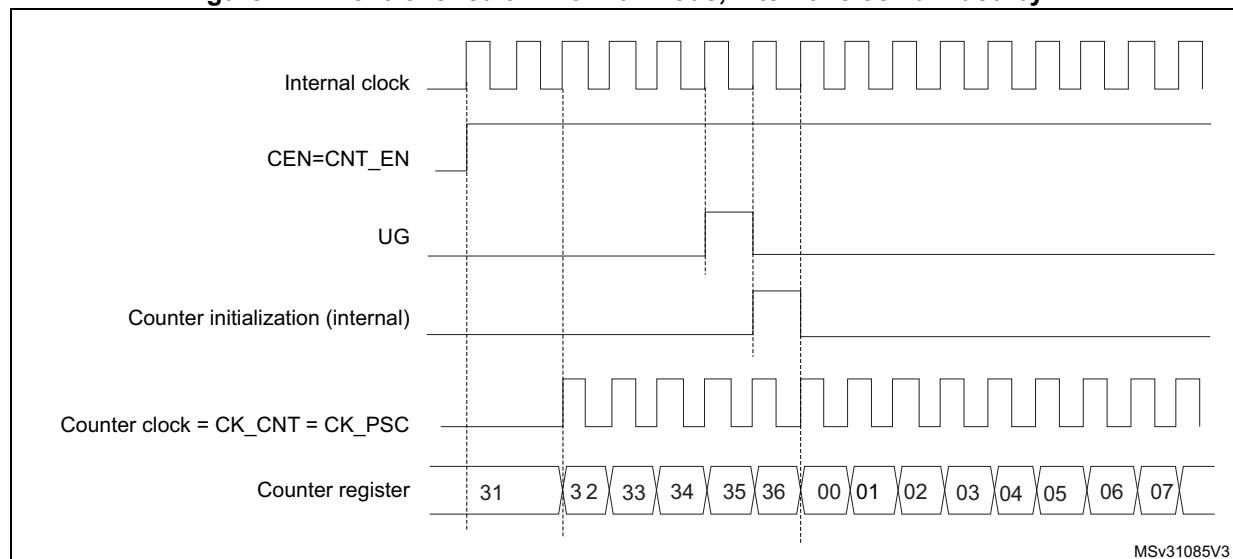
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

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 144 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

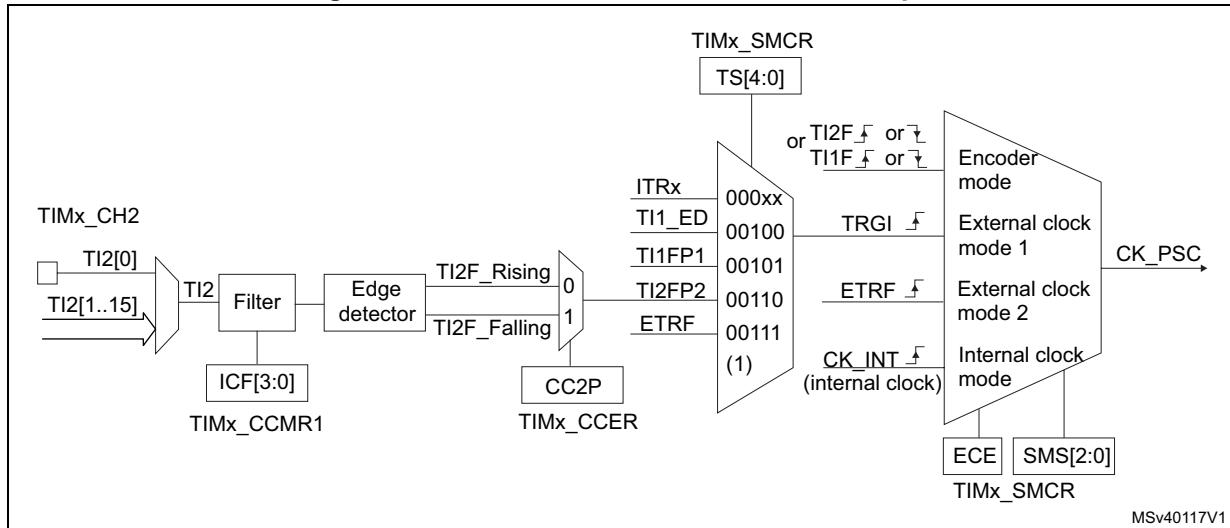
Figure 144. Control circuit in normal mode, internal clock divided by 1



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 145. 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:

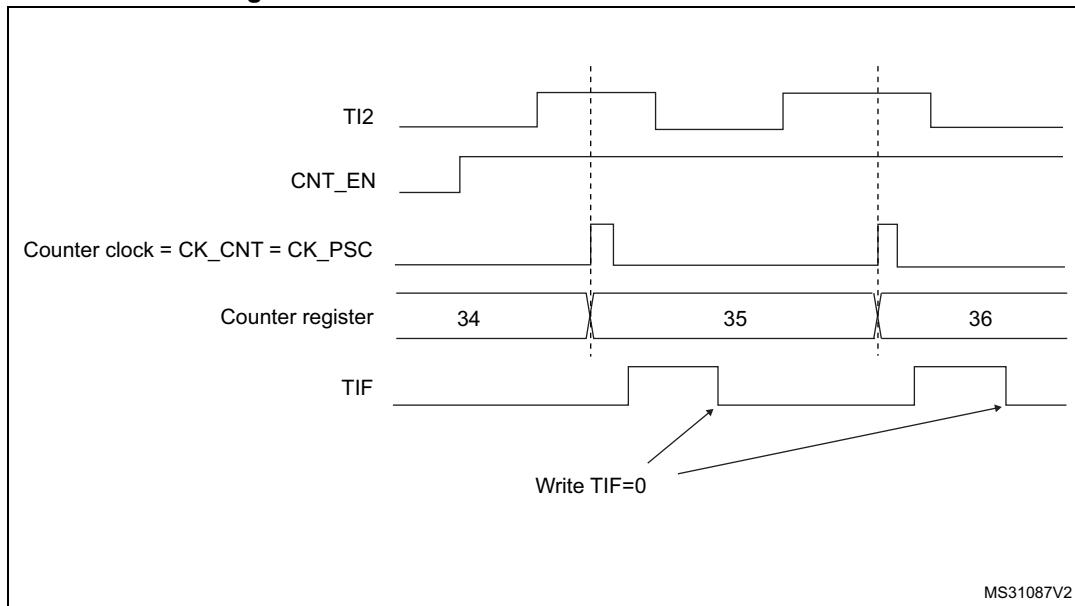
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 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 146. Control circuit in external clock mode 1



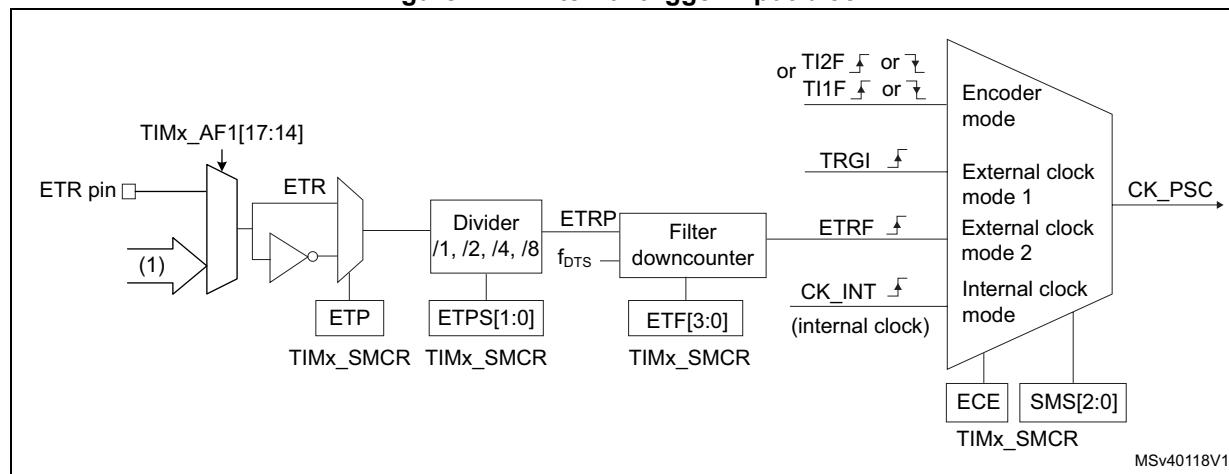
External clock source mode 2

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 147](#) gives an overview of the external trigger input block.

Figure 147. External trigger input block



1. Refer to *Figure 143: TIM1 ETR input circuitry*.

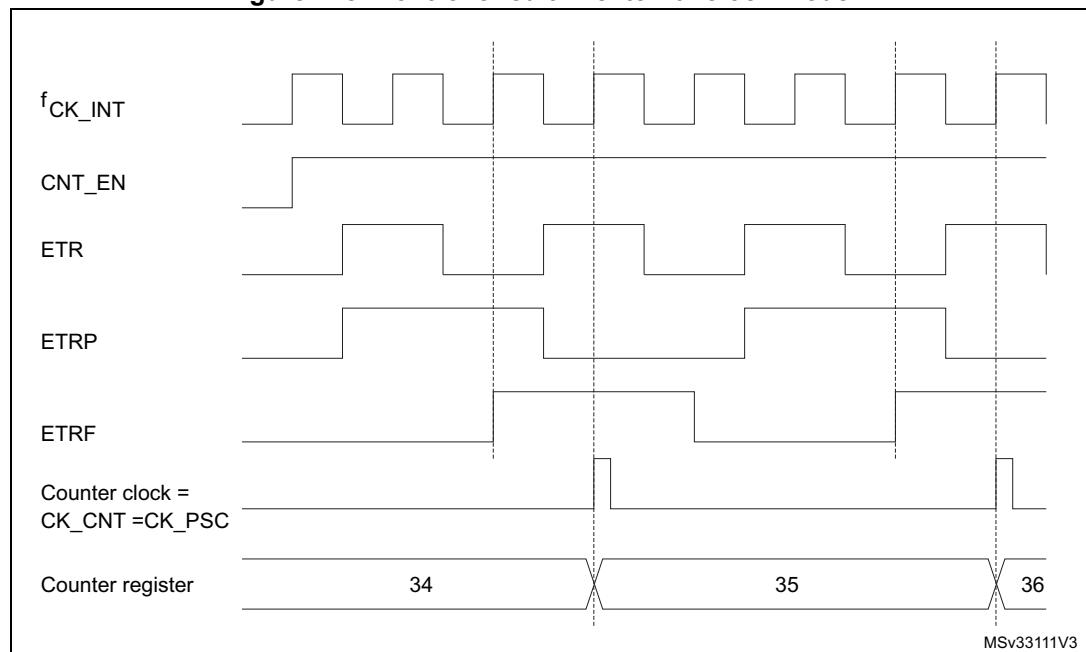
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

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 $\frac{1}{4}$ 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 148. Control circuit in external clock mode 2



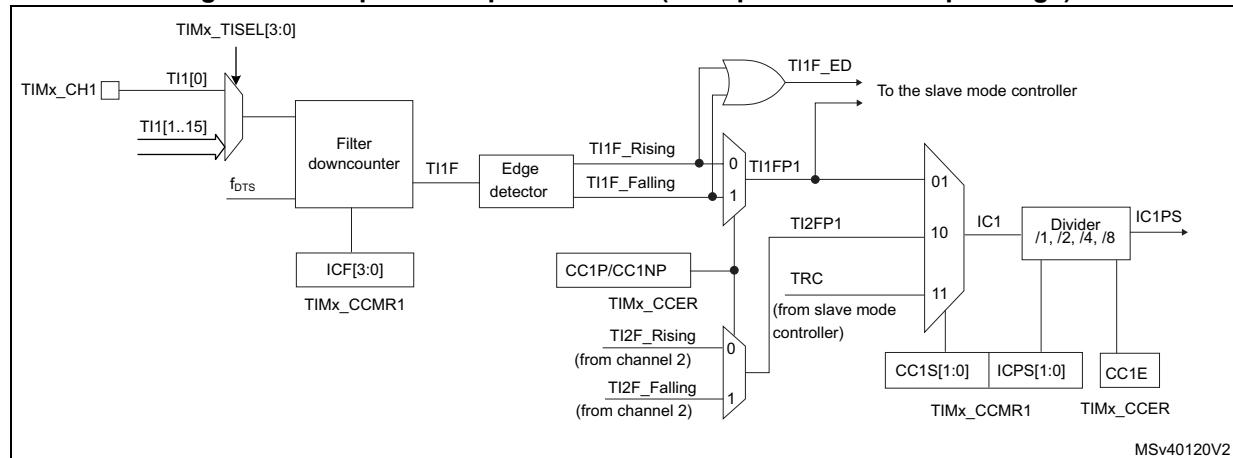
23.3.6 Capture/compare channels

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 149](#) to [Figure 152](#) 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 149. 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 150. Capture/compare channel 1 main circuit

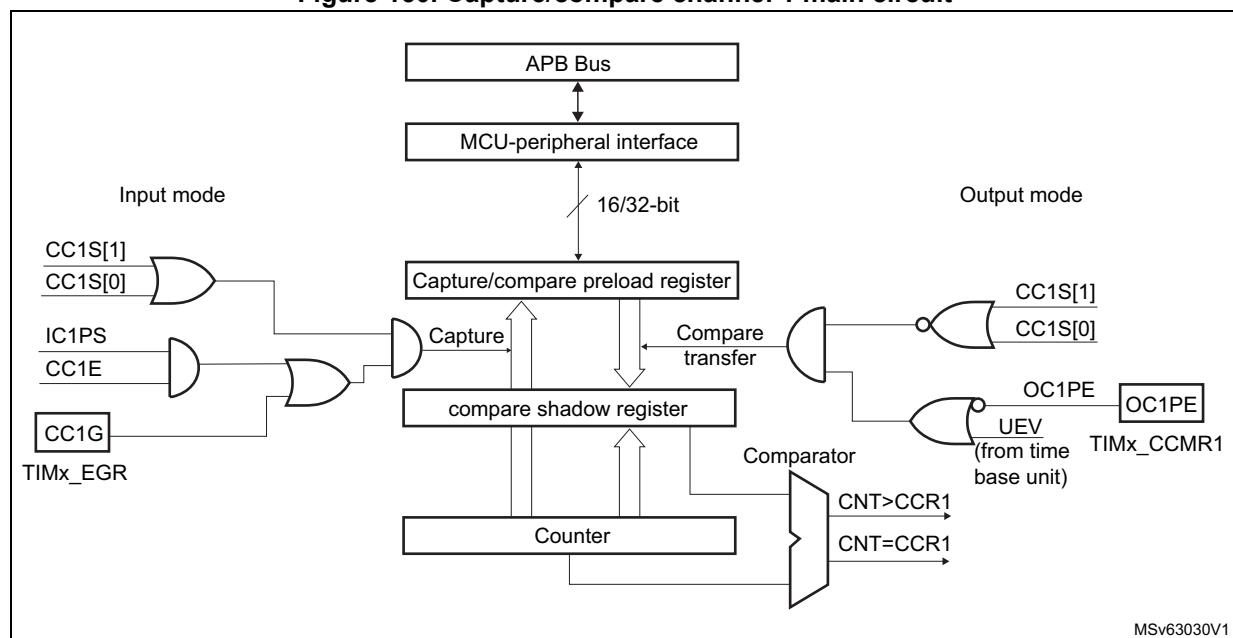


Figure 151. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)

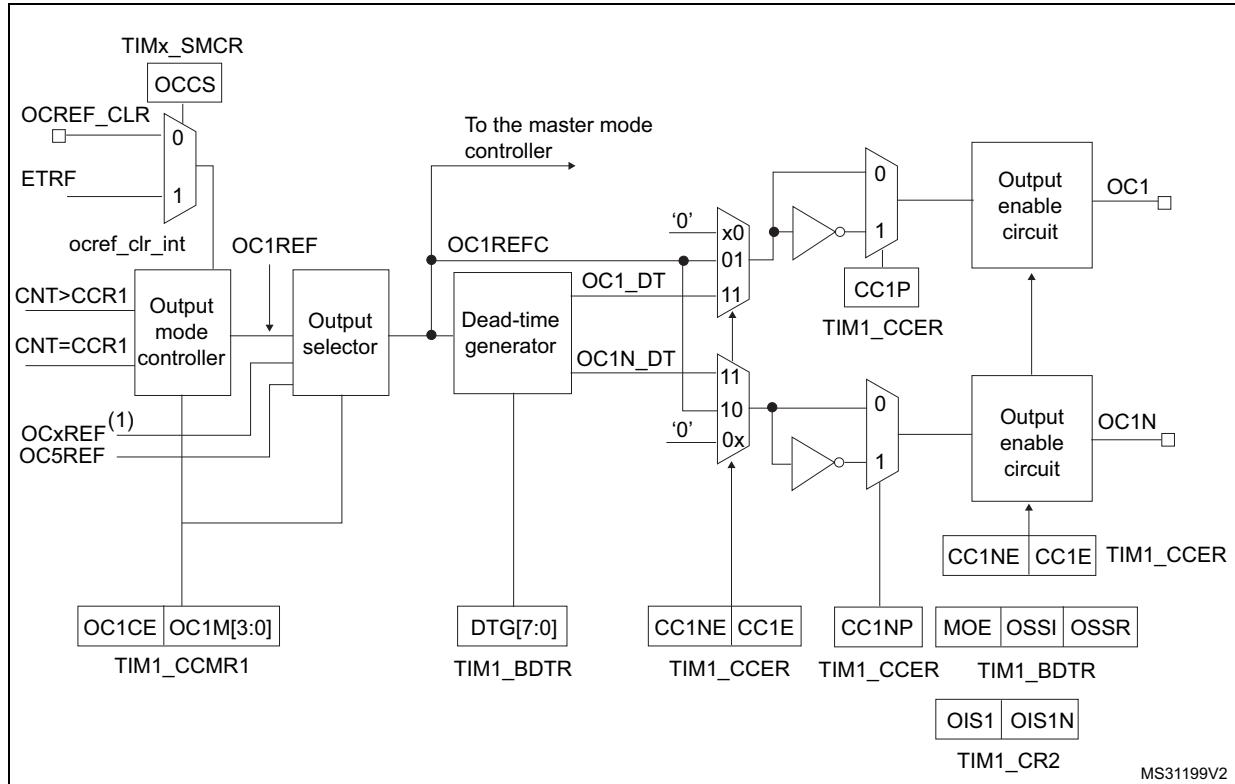


Figure 152. Output stage of capture/compare channel (channel 4)

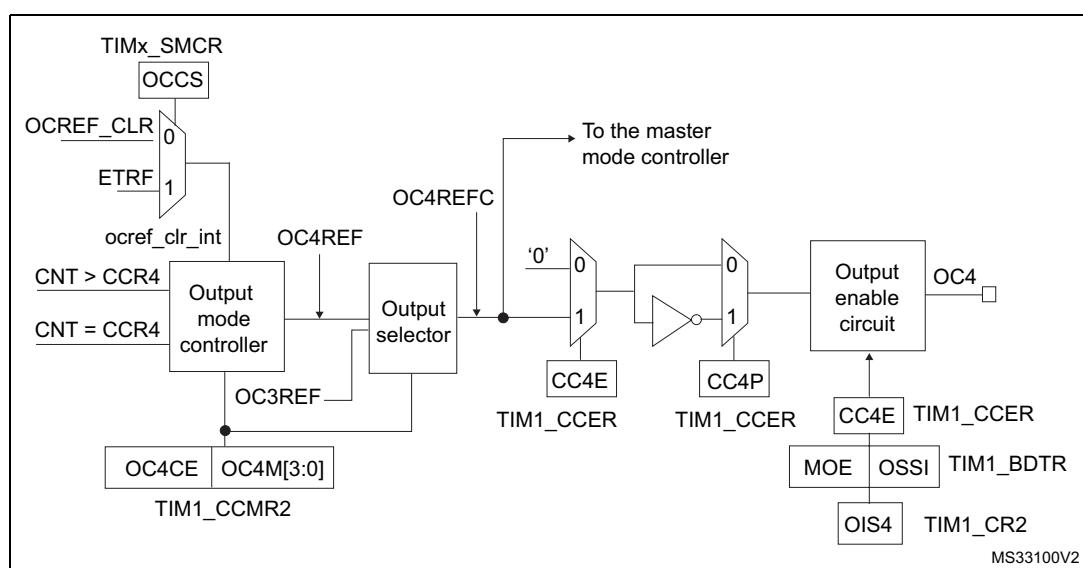
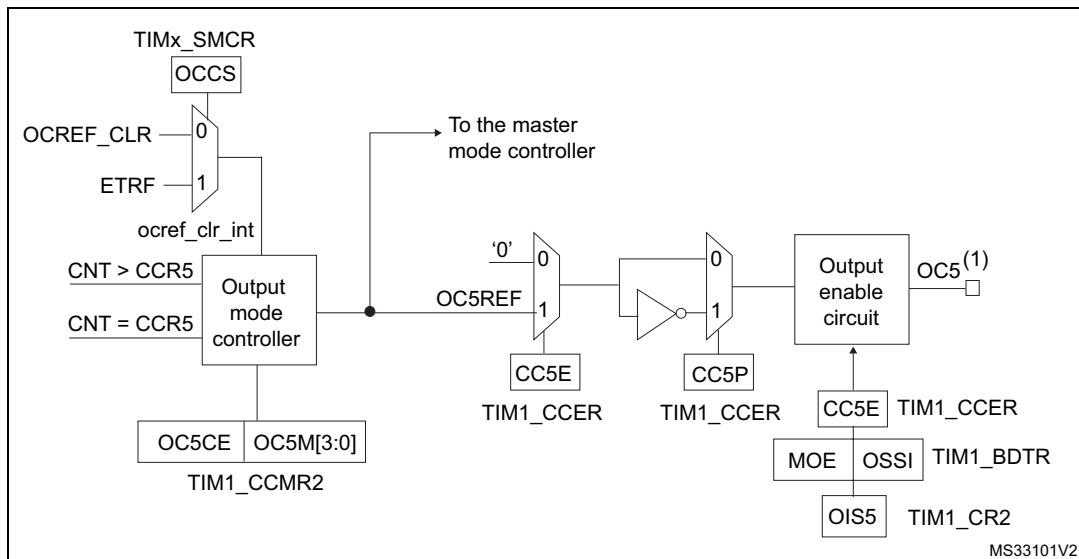


Figure 153. Output stage of capture/compare channel (channel 5, idem ch. 6)



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.

23.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCR_x) are used to latch the value of the counter after a transition detected by the corresponding IC_x 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 CCR_x 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 most 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.

23.3.8 PWM input mode

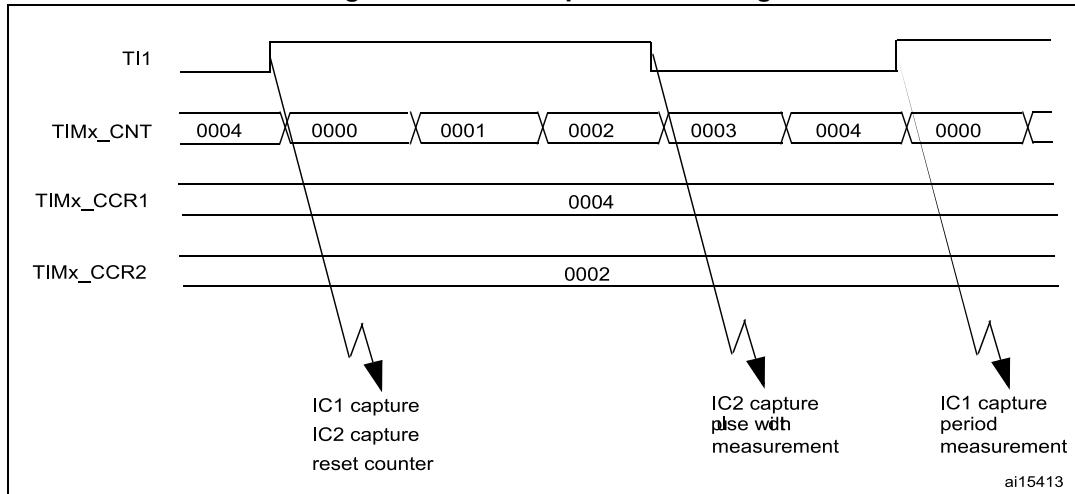
This mode is a particular case of input capture mode. The procedure is the same except:

- 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 154. PWM input mode timing



23.3.9 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.

23.3.10 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).

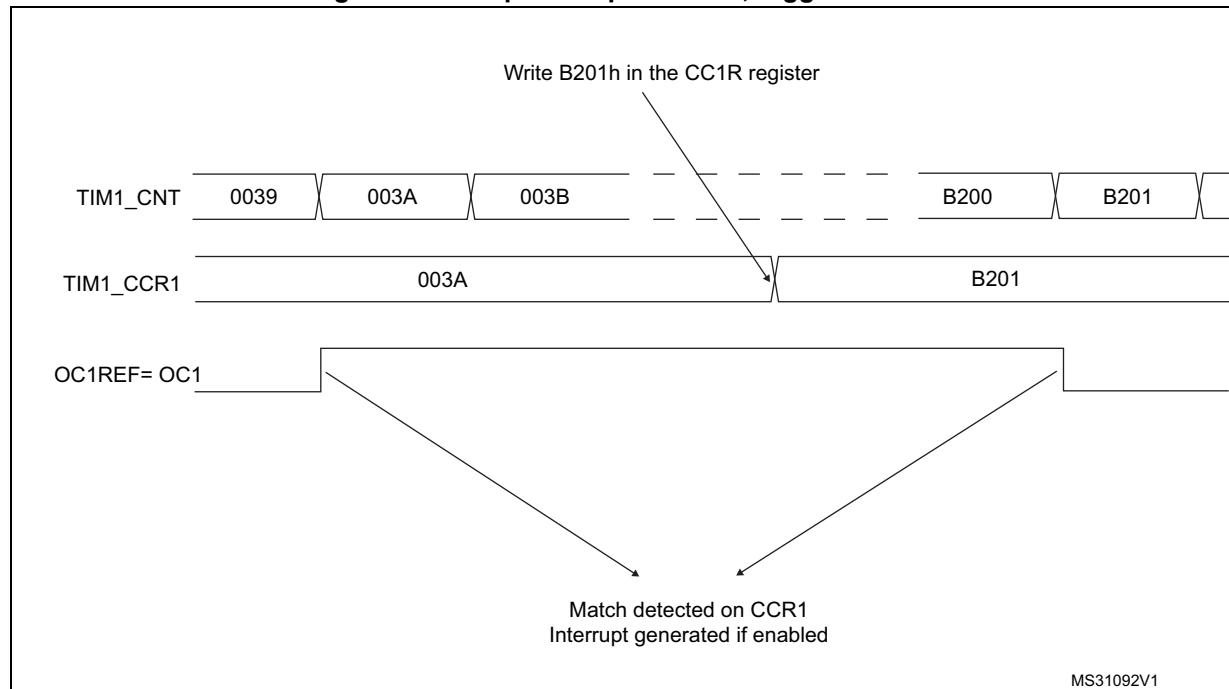
The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

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 155](#).

Figure 155. Output compare mode, toggle on OC1

23.3.11 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_CC Rx 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, OSS1 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_CC Rx are always compared to determine whether TIMx_CC Rx \leq TIMx_CNT or TIMx_CNT \leq TIMx_CC Rx (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

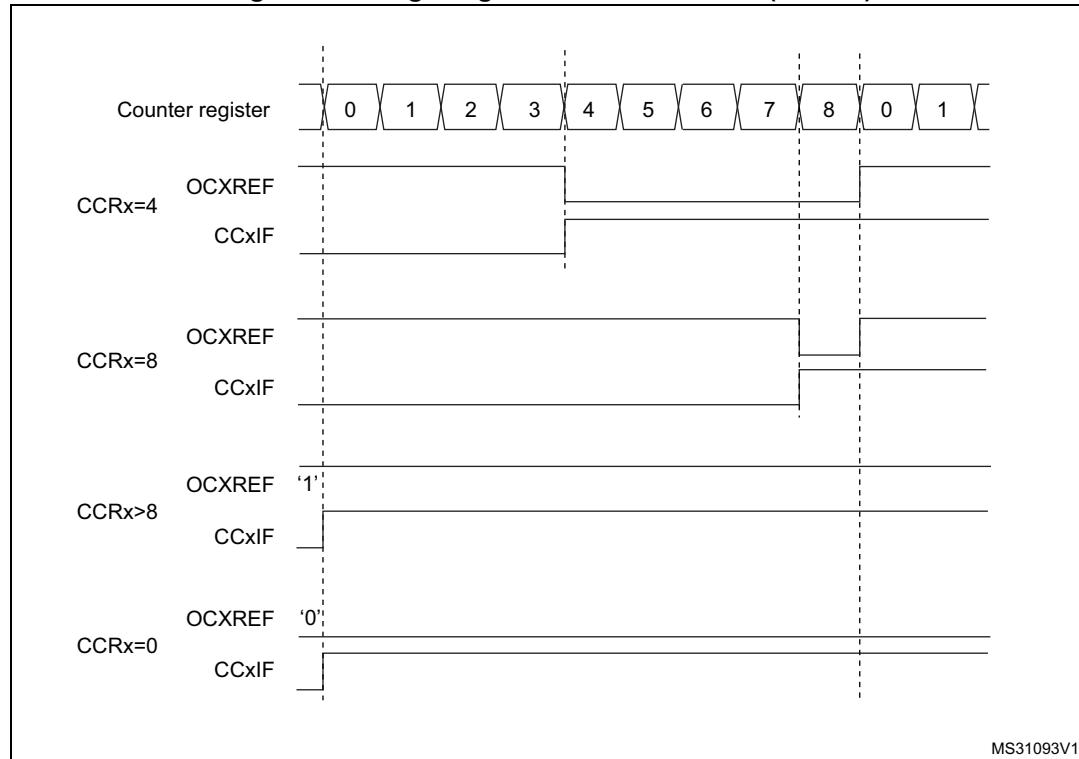
- Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the [Upcounting mode on page 555](#).

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as $\text{TIMx_CNT} < \text{TIMx_CCR}_x$ else it becomes low. If the compare value in TIMx_CCR_x 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 156](#) shows some edge-aligned PWM waveforms in an example where $\text{TIMx_ARR}=8$.

Figure 156. 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 559](#)

In PWM mode 1, the reference signal OCxRef is low as long as $\text{TIMx_CNT} > \text{TIMx_CCR}_x$ else it becomes high. If the compare value in TIMx_CCR_x 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

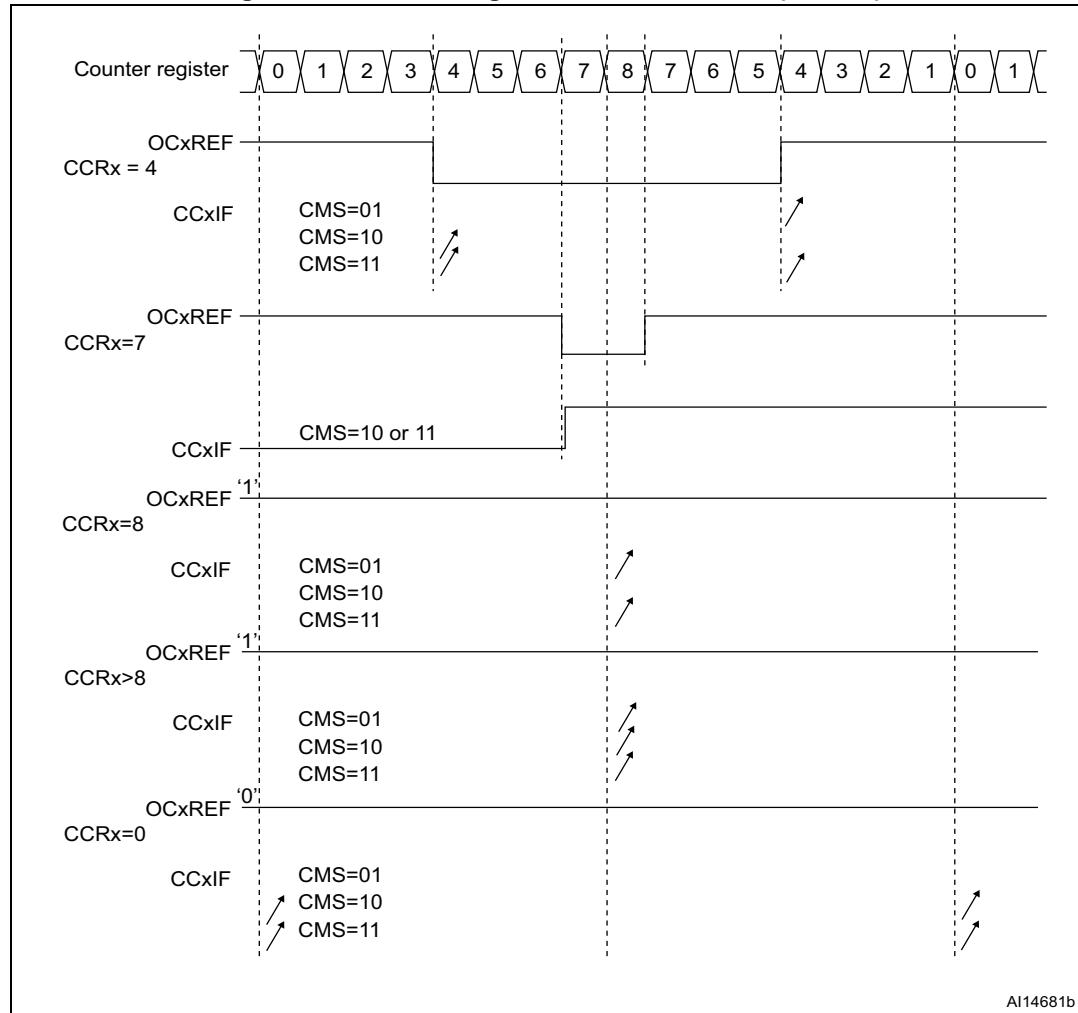
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the 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 562](#).

[Figure 157](#) 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 157. Center-aligned PWM waveforms (ARR=8)



AI14681b

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.

23.3.12 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:

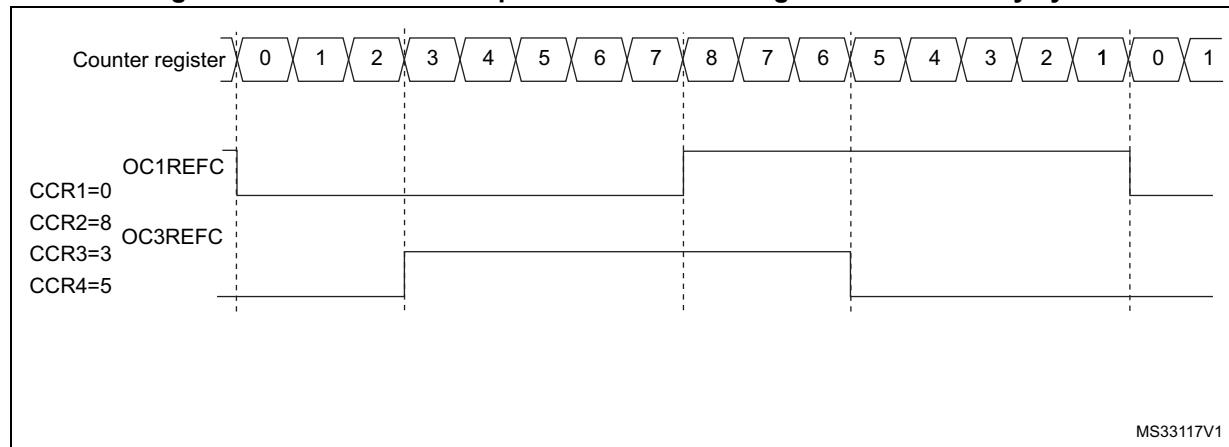
- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

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 158 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 158. Generation of 2 phase-shifted PWM signals with 50% duty cycle

23.3.13 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.

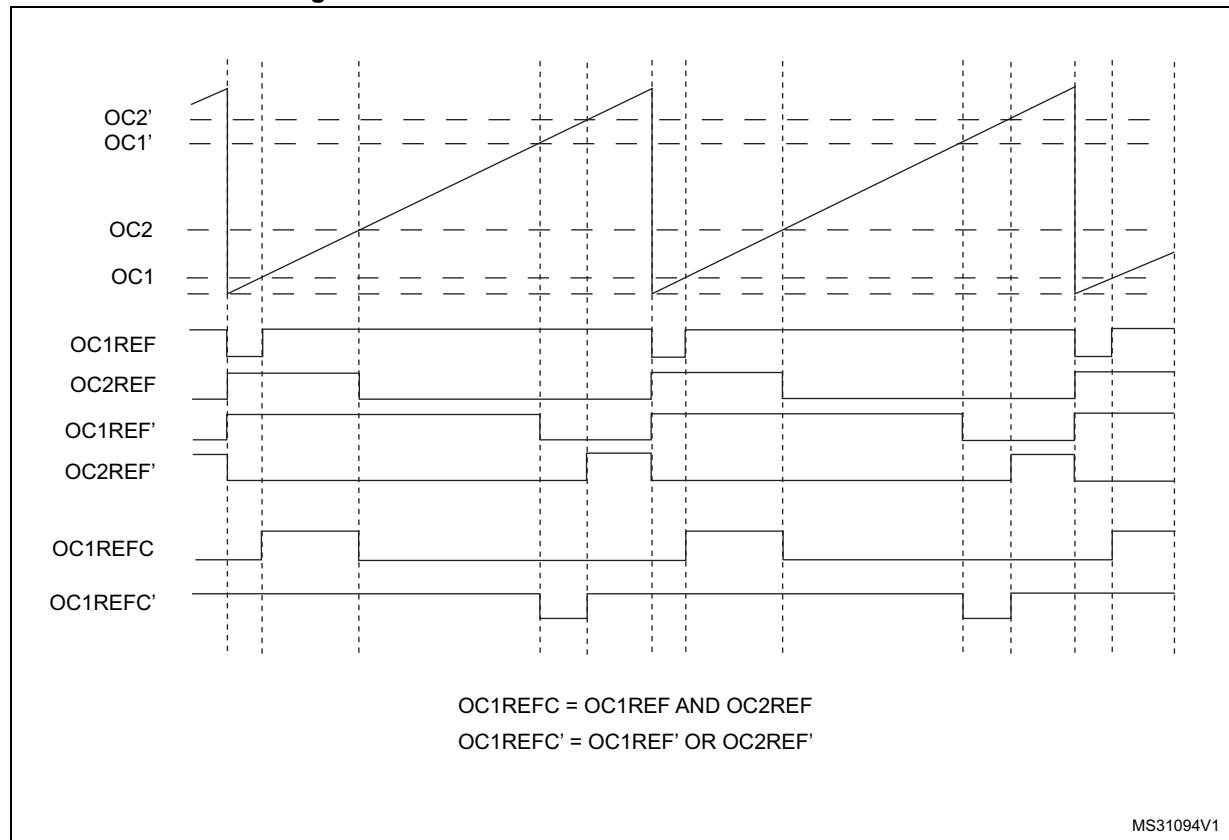
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 159 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.

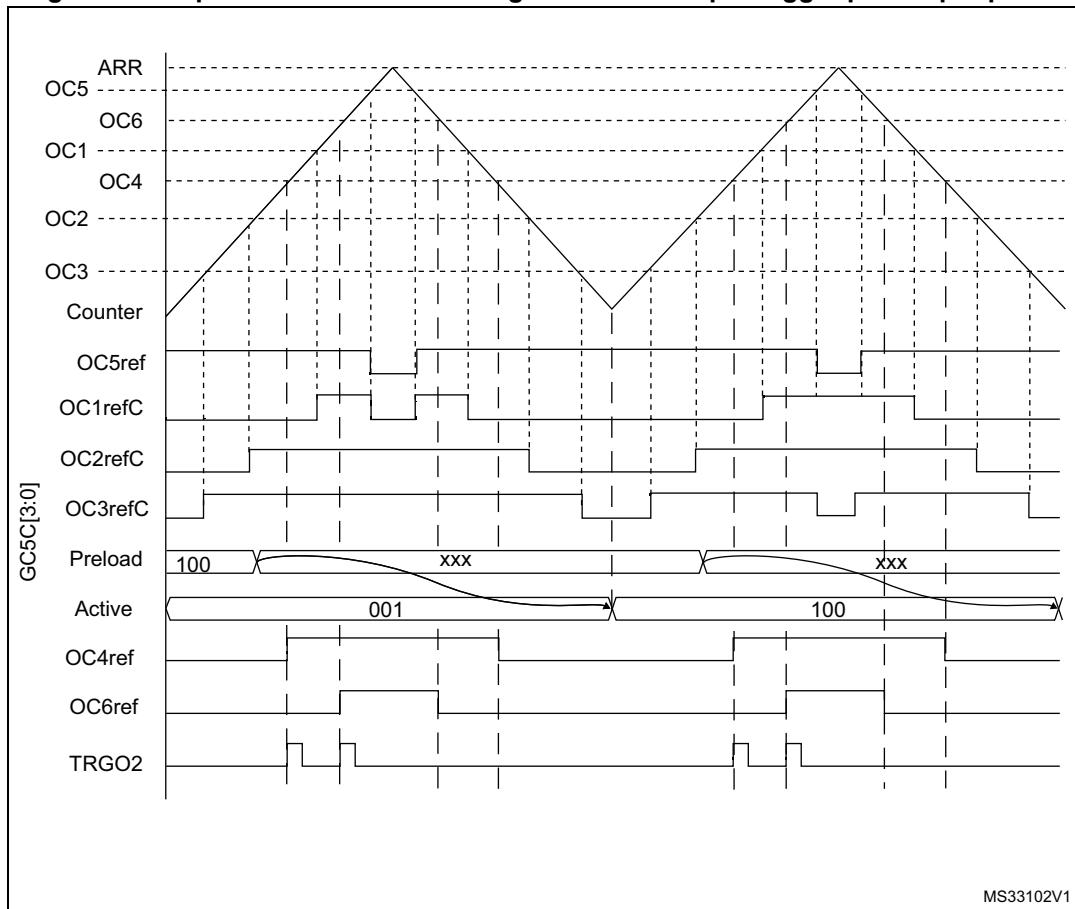
Figure 159. Combined PWM mode on channel 1 and 3

23.3.14 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 160. 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 23.3.27: ADC synchronization](#) for more details.

23.3.15 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 OC_x or complementary OC_{xN}) can be selected independently for each output. This is done by writing to the CC_{xP} and CC_{xNP} bits in the TIM_x_CCER register.

The complementary signals OC_x and OC_{xN} are activated by a combination of several control bits: the CC_{xE} and CC_{xNE} bits in the TIM_x_CCER register and the MOE, OIS_x, OIS_{xN}, OSS_I and OSS_R bits in the TIM_x_BDTR and TIM_x_CR2 registers. Refer to

[Table 130: Output control bits for complementary OC_x and OC_{xN} channels with break feature on page 631](#) for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

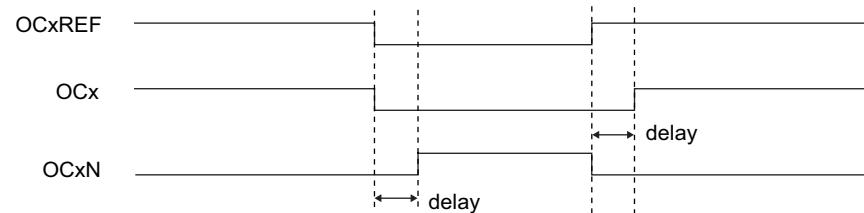
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- 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.

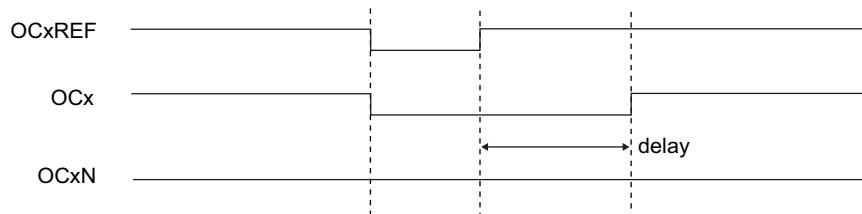
The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 161. Complementary output with dead-time insertion

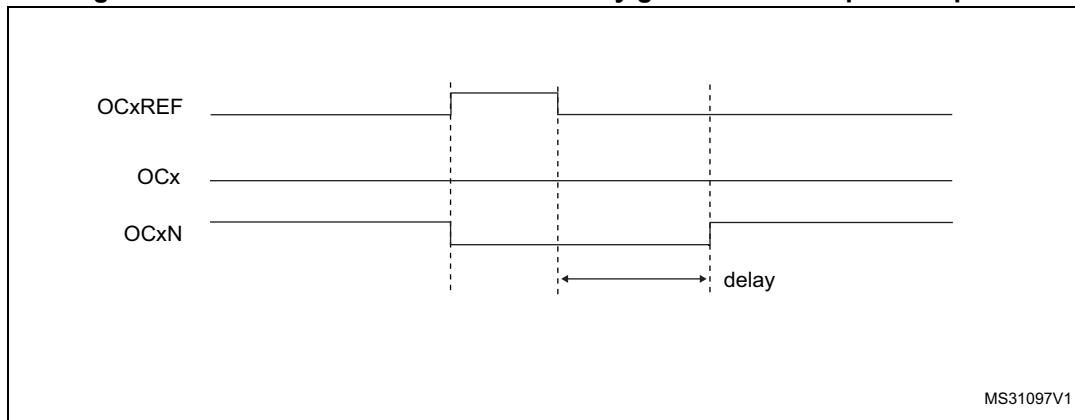


MS31095V1

Figure 162. Dead-time waveforms with delay greater than the negative pulse



MS31096V1

Figure 163. 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 23.4.20: 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.

23.3.16 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 built-in comparator), 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 OSS1 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 130: Output control bits for complementary OCx and OCxN channels with break feature on page 631](#) 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_OR2 and TIMx_OR3 registers.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering
- An internal source:
 - the Cortex®-M0+ LOCKUP output
 - the PVD output
 - the SRAM parity error signal
 - a flash memory ECC dual error detection
 - a clock failure event generated by the CSS detector
 - the output from a comparator, with polarity selection and optional digital filtering

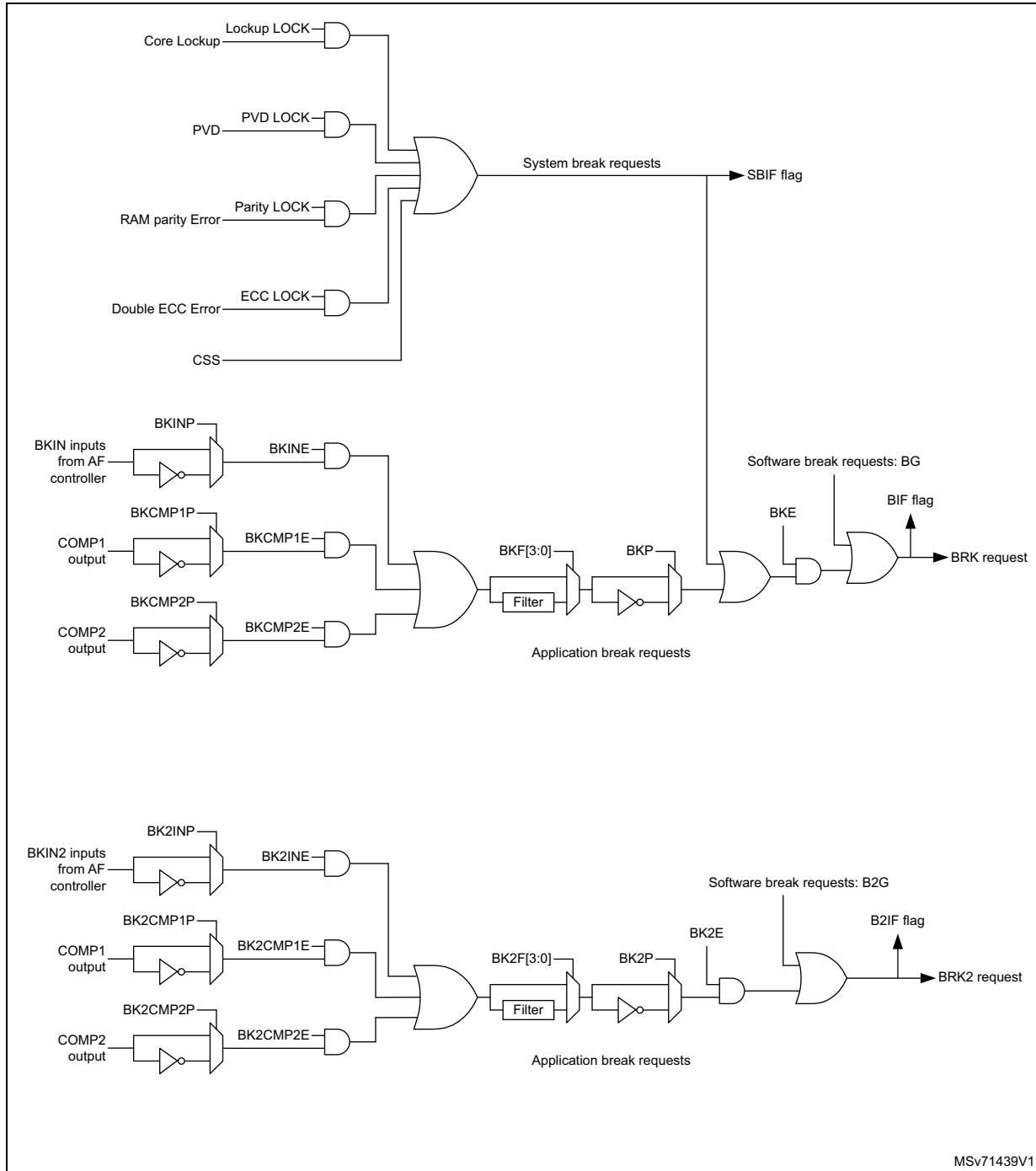
The sources for break2 (BRK2) are:

- An external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering
- An internal source coming from a comparator output.

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 164](#) below.

Figure 164. Break and Break2 circuitry overview



MSv71439V1

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 OSS1 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 OSS1=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 OSS1=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 OSS1 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.

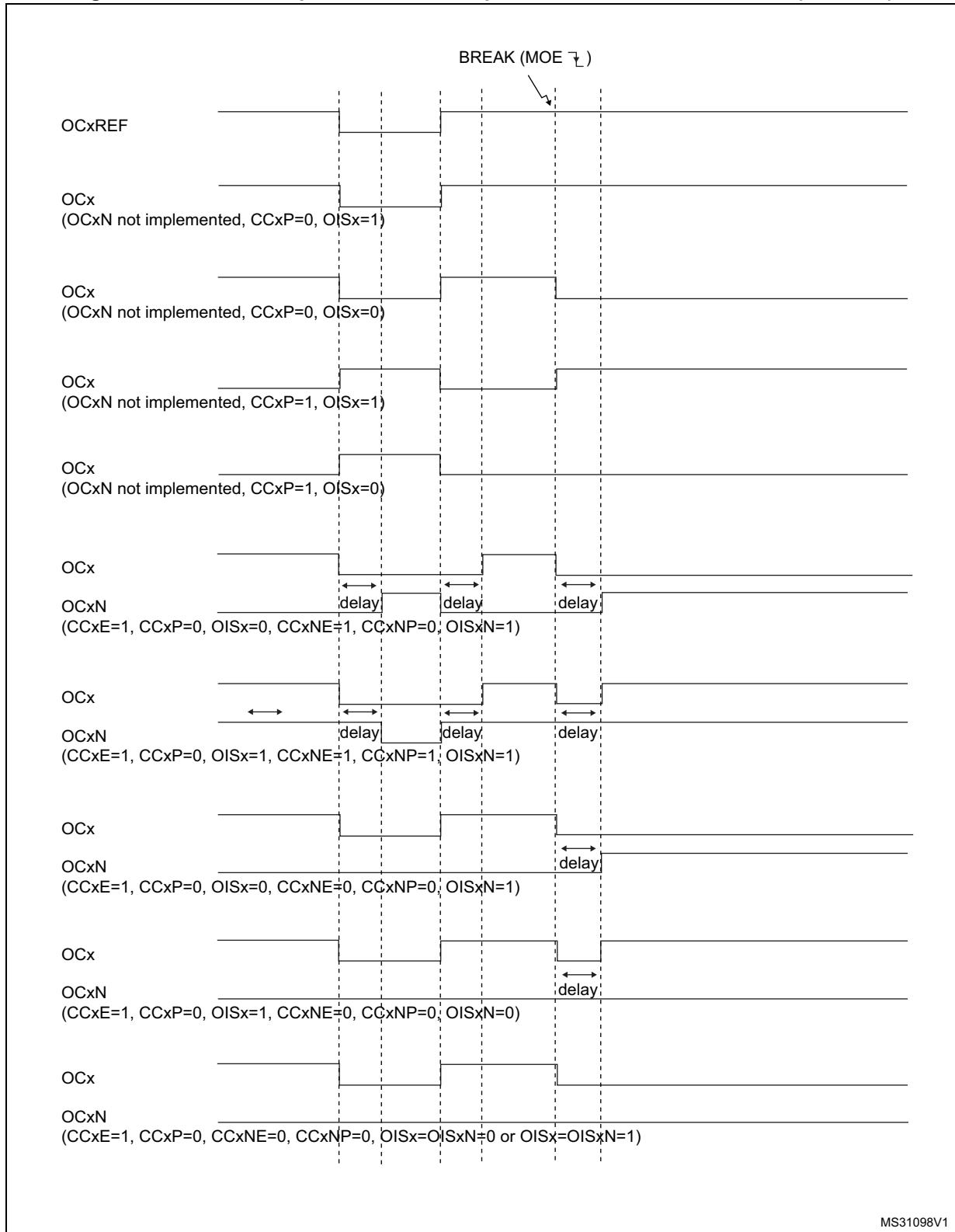
Note:

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 23.4.20: TIM1 break and dead-time register \(TIM1_BDTR\)](#). The LOCK bits can be written only once after an MCU reset.

Figure 165 shows an example of behavior of the outputs in response to a break.

Figure 165. Various output behavior in response to a break event on BRK (OSSI = 1)



MS31098V1

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 126](#).

Note: *BRK2 must only be used with OSSR = OSSI = 1.*

Table 126. 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 | <ul style="list-style-type: none"> – Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic) | ON after deadtime insertion | OFF |
| Inactive | Active | Inactive | OFF | OFF |

[Figure 166](#) 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 166. PWM output state following BRK and BRK2 pins assertion (OSSI=1)

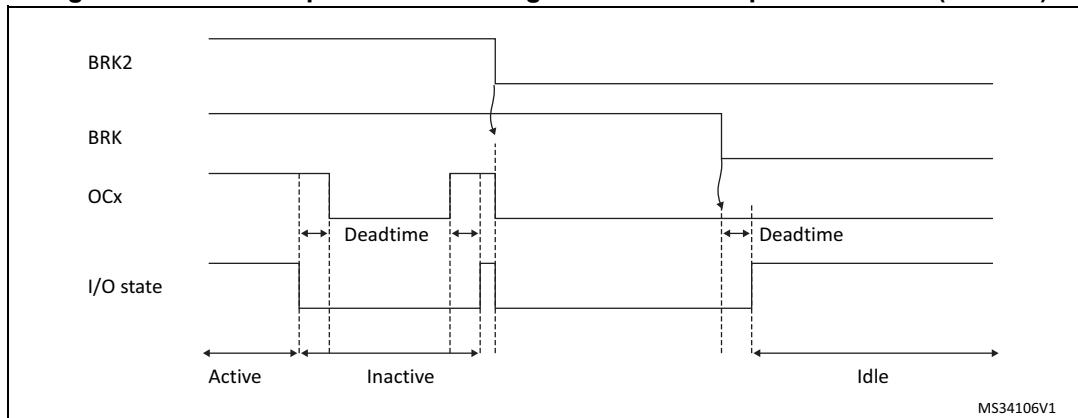
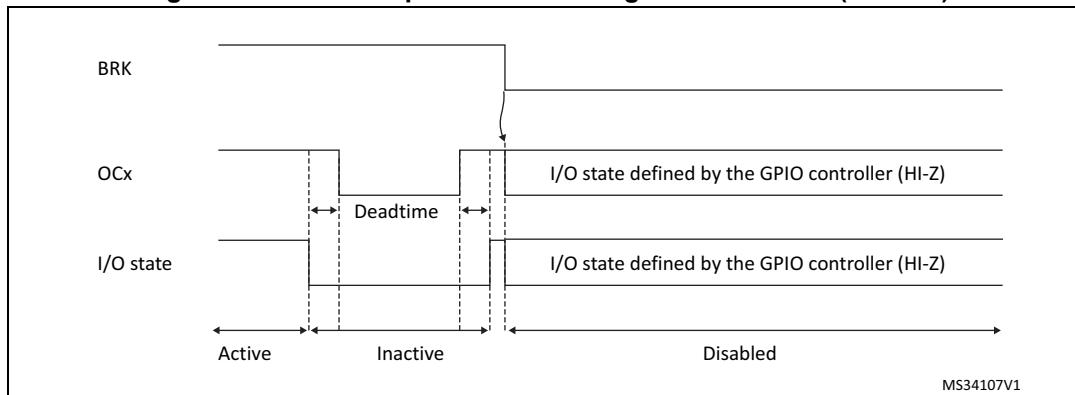


Figure 167. PWM output state following BRK assertion (OSSI=0)

23.3.17 Bidirectional break inputs

The TIM1 are featuring bidirectional break I/Os, as represented on [Figure 168](#).

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 BKBDID and BK2BDID bits in the TIMxBDTR register. The BKBDID 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 127](#))

Table 127. 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 |

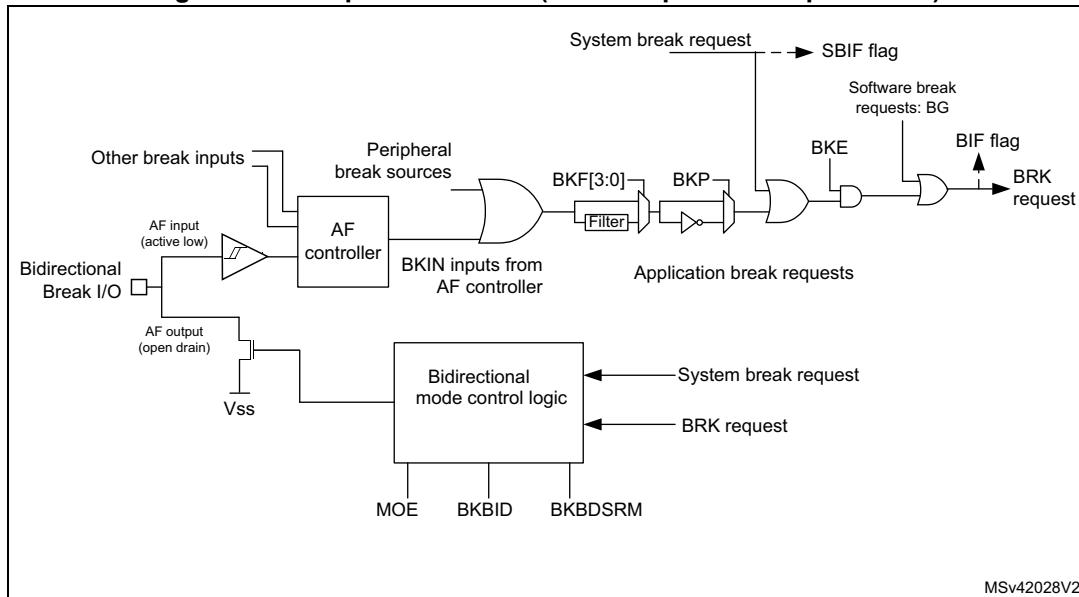
Arming and re-arming break circuitry

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 168. Output redirection (BRK2 request not represented)

MSv42028V2

23.3.18 Clearing the OCxREF signal on an external event

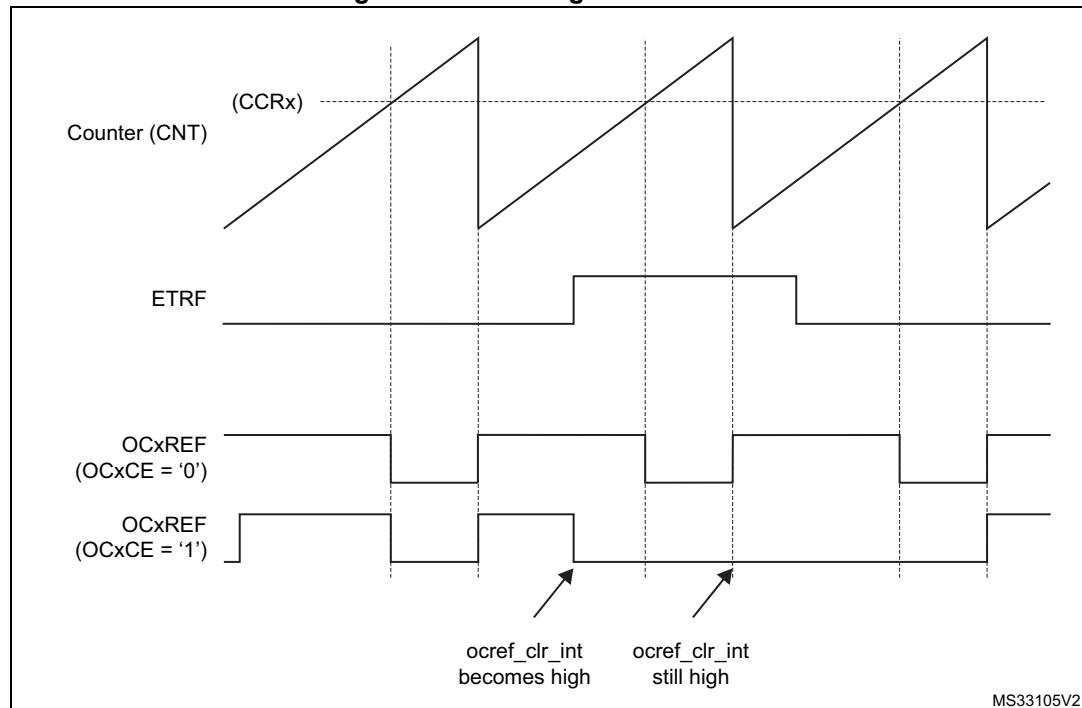
The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next transition to the active state, on the following PWM cycle. This function can only be used 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 169 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 169. 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.

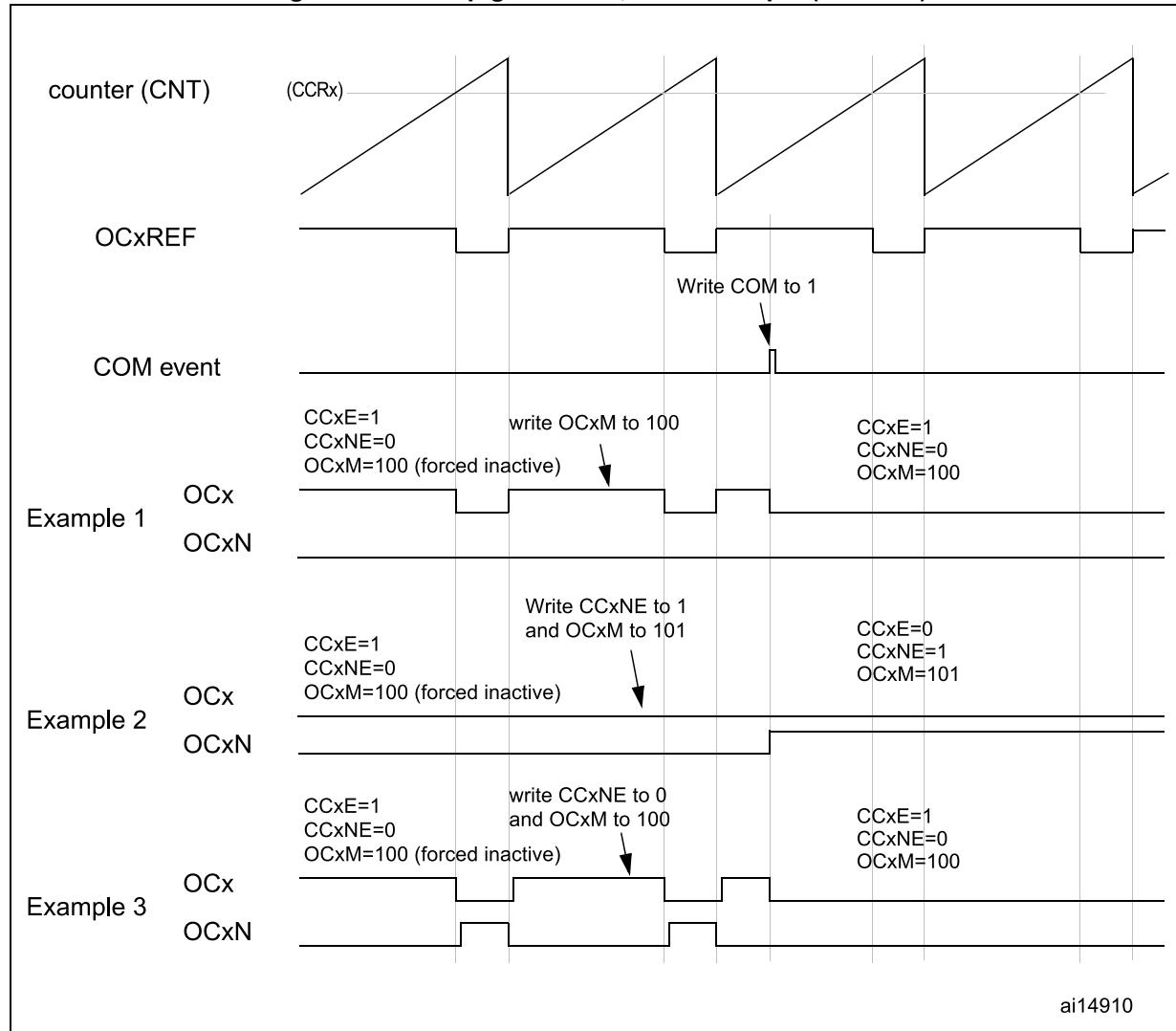
23.3.19 6-step PWM generation

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 170](#) describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 170. 6-step generation, COM example (OSSR=1)



23.3.20 One-pulse mode

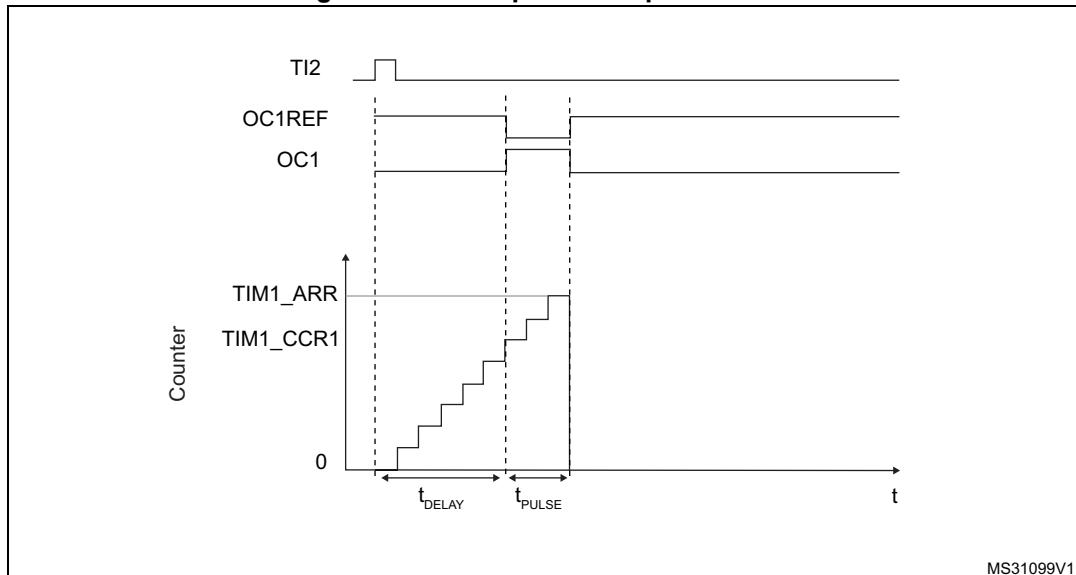
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 \leq ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 171. Example of one pulse mode.



MS31099V1

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.

23.3.21 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 23.3.20](#):

- 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

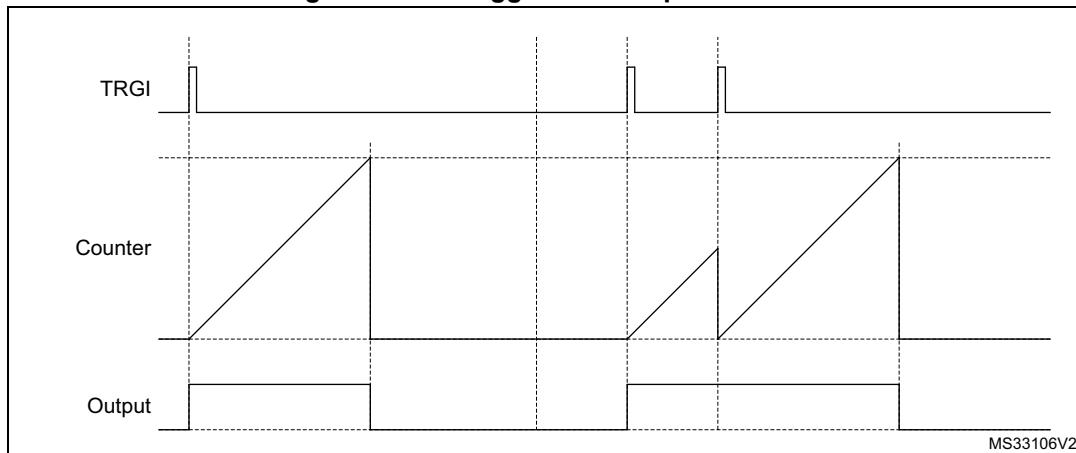
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for retriggerable OPM mode 1 or 2.

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 172. Retriggerable one pulse mode



23.3.22 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 128](#). 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 128. 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 173](#) 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 173. Example of counter operation in encoder interface mode.

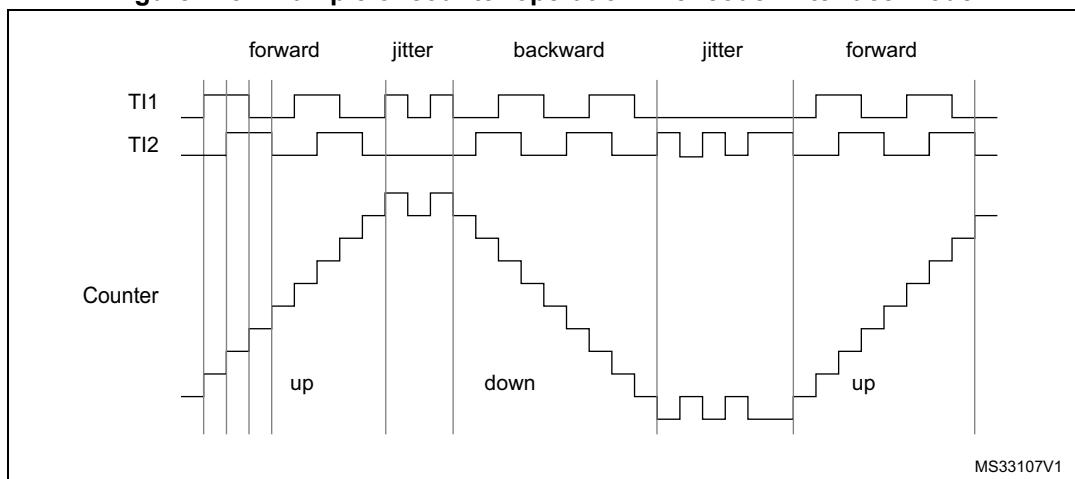
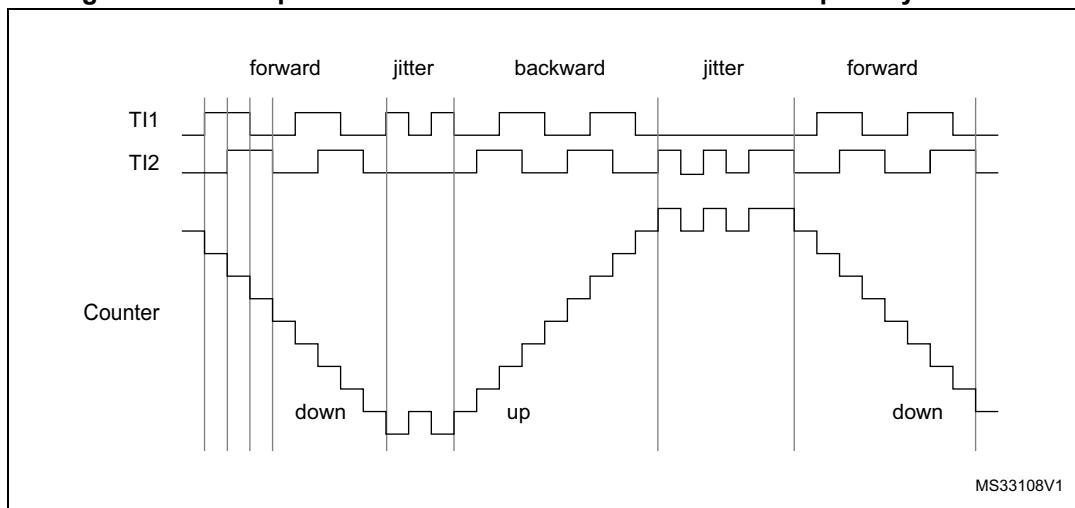


Figure 174 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 174. 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).

23.3.23 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).

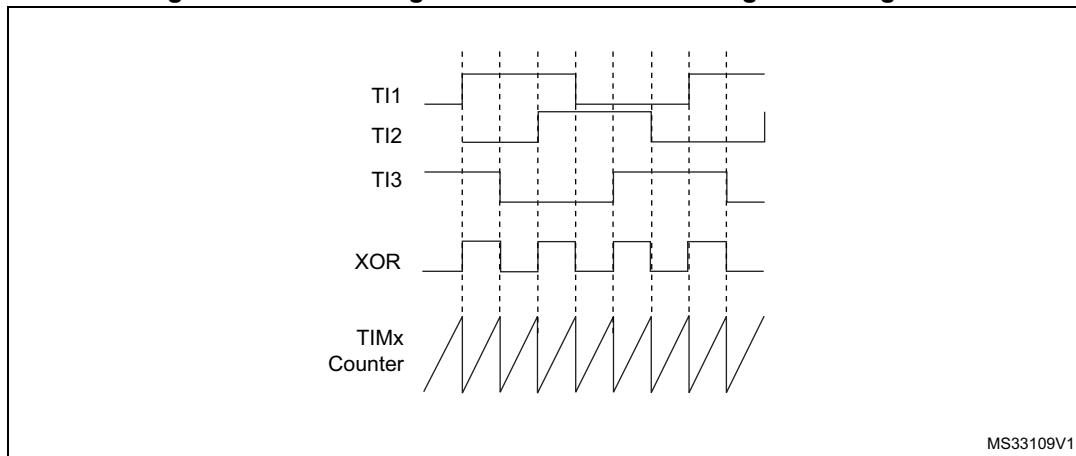
There is no latency between the UIF and UIFCPY flags assertion.

23.3.24 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 175](#) below.

Figure 175. Measuring time interval between edges on 3 signals



23.3.25 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, TIM3) referred to as “interfacing timer” in [Figure 176](#). 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 149: Capture/compare channel \(example: channel 1 input stage\) on page 573](#)). 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.

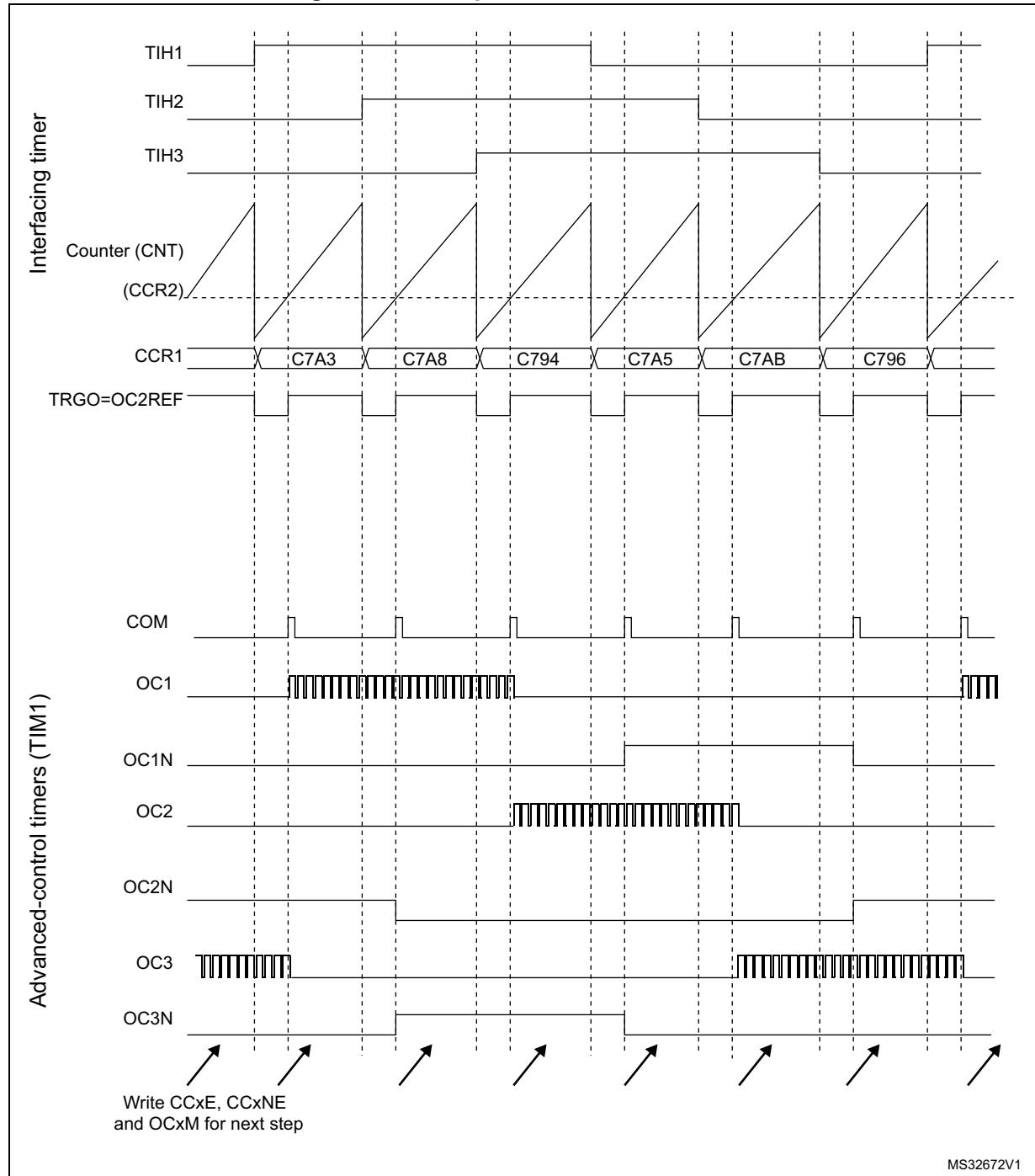
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- 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 176](#) describes this example.

Figure 176. Example of Hall sensor interface



23.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to [Section 24.3.19: 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.

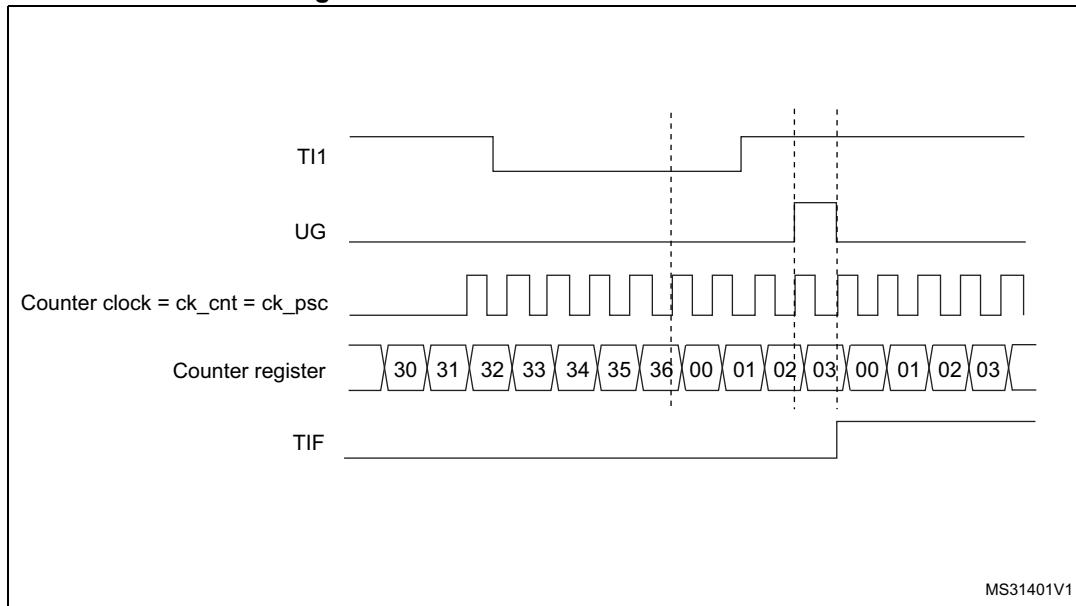
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- 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 177. Control circuit in reset mode



MS31401V1

Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

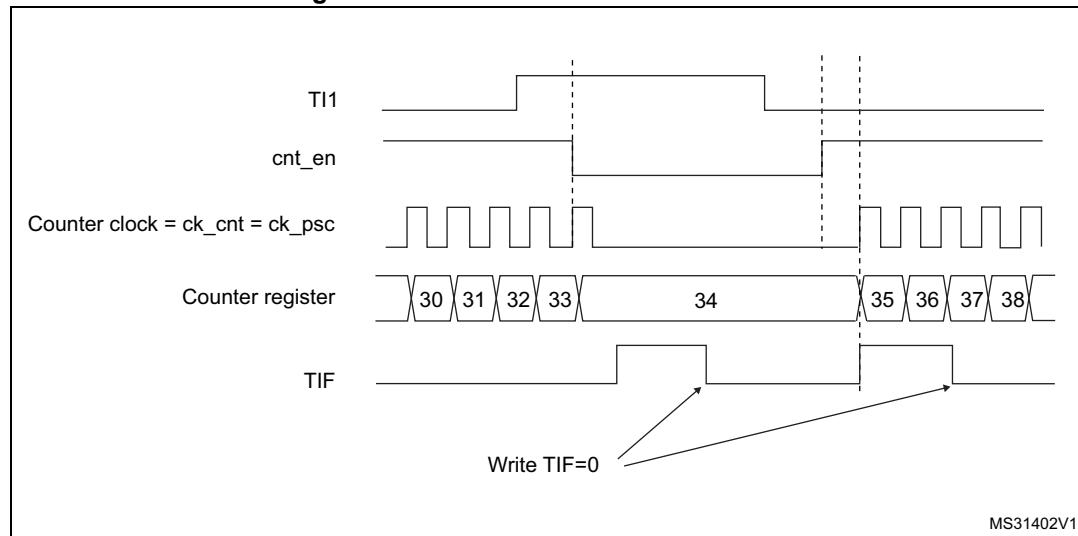
In the following example, the upcounter counts only when TI1 input is low:

- 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 178. 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

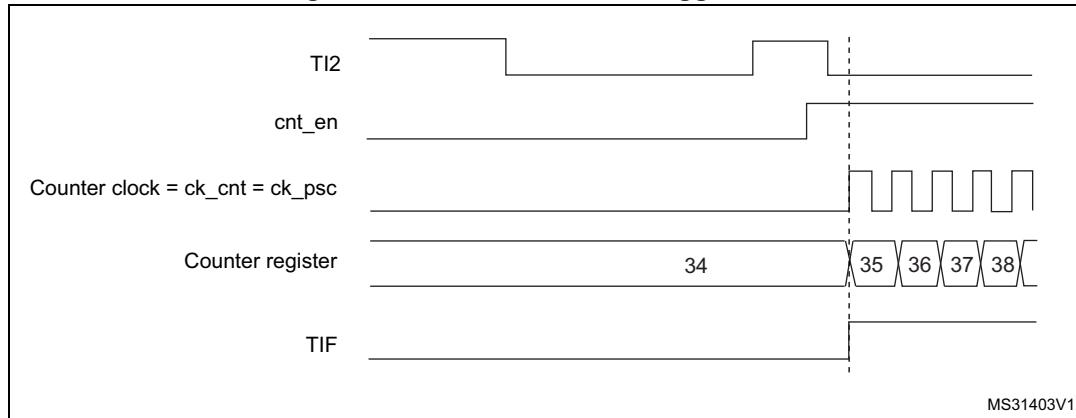
register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- 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 179. 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.

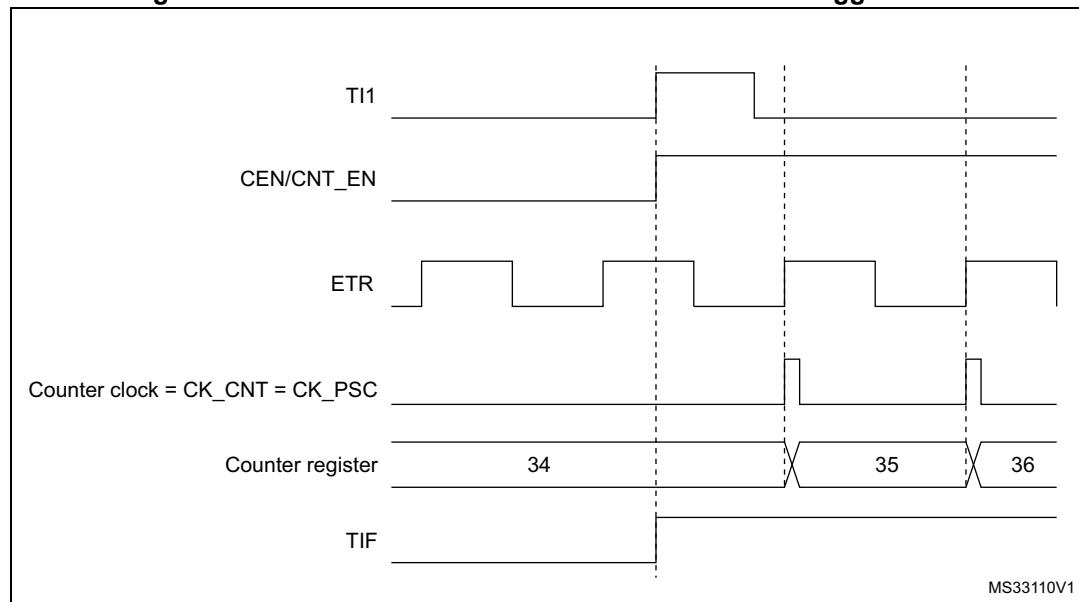
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

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 = 01 in 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 180. 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.

23.3.27 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 160 on page 585](#).

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.*

23.3.28 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.

23.3.29 Debug mode

When the microcontroller enters debug mode (Cortex[®]-M0+ core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG 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).

23.4 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).

23.4.1 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. | UIFRE MAP | Res. | CKD[1:0] | | ARPE | CMS[1:0] | | DIR | OPM | URS | UDIS | CEN |
| | | | | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:12 Reserved, 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.

23.4.2 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 | 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.

23.4.3 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 129: TIM1 internal trigger connection on page 617](#) 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 COMP1 or COMP2 output depending on TIM1_OR1.OCREF_CLR
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 (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

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 129. TIM1 internal trigger connection

| Slave TIM | ITR0 (TS = 00000) | ITR1 (TS = 00001) | ITR2 (TS = 00010) | ITR3 (TS = 00011) |
|-----------|-------------------|-------------------|-------------------|-------------------|
| TIM1 | TIM15 | TIM2 | TIM3 | Reserved |

23.4.4 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 |

- 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

23.4.5 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 | SBIFF | CC4OF | CC3OF | CC2OF | CC1OF | B2IF | BIF | TIF | COMIF | CC4IF | CC3IF | CC2IF | CC1IF | UIF |
| | | 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 **SBIFF**: 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 23.4.3: TIM1 slave mode control register \(TIM1_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx_CR1 register.

23.4.6 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. | B2G | BG | TG | COMG | CC4G | CC3G | CC2G | CC1G | UG |

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).

23.4.7 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).

23.4.8 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. OCCE 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).

23.4.9 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).

23.4.10 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).

23.4.11 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, OSS1, OSSR, OIS1, OIS1N and CC1E bits.
- 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, 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), TIxFP1 is 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, OSS1, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 130](#) 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 130. 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) OCx=0, OCxN=0 | | | |
| | | 0 | 0 | 1 | Output disabled (not driven by the timer: Hi-Z) OCx=0 OCxREF + Polarity OCxN = OCxREF xor CCxNP | | | |
| | | 0 | 1 | 0 | OCxREF + Polarity OCx=OCxREF xor CCxP Output Disabled (not driven by the timer: Hi-Z) 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) OCx=CCxP OCxREF + Polarity OCxN = OCxREF x or CCxNP | | | |
| | | 1 | 1 | 0 | OCxREF + Polarity OCx=OCxREF xor CCxP Off-State (output enabled with inactive state) OCxN=CCxNP | | | |
| 0 | 0 | X | X | Output disabled (not driven by the timer: Hi-Z). | | | | |
| | | 0 | 0 | | | | | |
| | | 0 | 1 | Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered). | | | | |
| | | 1 | 0 | | | | | |
| | | 1 | 1 | 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). Note: BRK2 can only be used if OSSI = OSSR = 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.

23.4.12 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. |
| r | | | | | | | | | | | | | | | |
| 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

23.4.13 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”).

23.4.14 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 23.3.1: Time-base unit on page 553](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

23.4.15 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.

23.4.16 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.

23.4.17 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.

23.4.18 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.

23.4.19 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.

23.4.20 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 | BK2DSRM | BK DSRM | BK2P | BK2E | BK2F[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, OSSR, 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 BK_{BID}: Break Bidirectional

- 0: Break input BRK in input mode
- 1: Break input BRK in bidirectional mode

In the bidirectional mode (BK_{BID} 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 BK_{2DSRM}: Break2 Disarm

Refer to BK_{DSRM} description

Bit 26 BK_{DSRM}: 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 BK_{DSRM} 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 BK_{2P}: 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 BK_{2E}: Break 2 enable

- 0: Break input BRK2 disabled
- 1: Break input BRK2 enabled

Note: The BRK2 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: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=2
- 0010: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=4
- 0011: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=8
- 0100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N=6
- 0101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N=8
- 0110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N=6
- 0111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N=8
- 1000: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N=6
- 1001: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N=8
- 1010: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=5
- 1011: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=6
- 1100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=8
- 1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N=5
- 1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N=6
- 1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, 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: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=2
- 0010: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=4
- 0011: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=8
- 0100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N=6
- 0101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N=8
- 0110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N=6
- 0111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N=8
- 1000: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N=6
- 1001: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N=8
- 1010: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=5
- 1011: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=6
- 1100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=8
- 1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N=5
- 1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N=6
- 1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, 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 OSS1 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 23.4.11: 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 164: 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 23.4.11: 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 23.4.11: 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 BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSS1, 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 OSS1 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 x t_{DTS}.

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t_{DTG} with t_{DTG} = 8 x t_{DTS}.

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t_{DTG} with t_{DTG} = 16 x 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).

23.4.21 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 | 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:
 $(\text{TIMx_CR1 address}) + \text{DBA} + (\text{DMA index})$, where DMA index = DBL

In this example, 7 bytes are added to $(\text{TIMx_CR1 address}) + \text{DBA}$, 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: $(\text{TIMx_CR1 address}) + \text{DBA}$

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,

...

23.4.22 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).

23.4.23 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | OCREF_CLR [1:0] | |
| | | | | | | | | | | | | | | rw | rw |

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

Bits 1:0 **OCREF_CLR[1:0]**: Ocref_clr source selection

This bit selects the ocref_clr input source.

00: COMP1 output is connected to the OCREF_CLR input

01: COMP2 output is connected to the OCREF_CLR input

Others: Reserved

23.4.24 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.

23.4.25 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. |
| rw | rw | rw | | | | | | | | | | | | | |
| 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.

23.4.26 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.

23.4.27 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. | BK CMP2P | BK CMP1P | BKINP | Res. | Res. | Res. | Res. | Res. | Res. | BK CMP2E | BK CMP1E | BKINE |
| rw | rw | | | rw | rw | 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

0001: COMP1 output

0010: COMP2 output

0011: ADC1 AWD1

0100: ADC1 AWD2

0101: ADC1 AWD3

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 13:12 Reserved, must be kept at reset value.

Bit 11 **BKCMPP2P: BRK COMP2 input polarity**

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: COMP2 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).

Bit 10 **BKCMPP1P: BRK COMP1 input polarity**

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: COMP1 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).

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 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCM2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCM1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN 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 143: TIM1 ETR input circuitry](#) and to [Figure 164: Break and Break2 circuitry overview](#).

23.4.28 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. | BK2 CMP2 P | BK2 CMP1 P | BK2 INP | Res. | Res. | Res. | Res. | Res. | Res. | BK2 CMP2E | BK2 CMP1E | BK2INE |
| | | | | rw | rw | rw | | | | | | | rw | rw | rw |

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BK2P polarity bit.

- 0: COMP2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)
- 1: COMP2 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).

Bit 10 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BK2P polarity bit.

- 0: COMP1 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)
- 1: COMP1 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).

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 8:3 Reserved, must be kept at reset value.

Bit 2 **BK2CMP2E**: BRK2 COMP2 enable

This bit enables the COMP2 for the timer's BRK2 input. COMP2 output is 'ORed' with the other BRK2 sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: BRK2 COMP1 enable

This bit enables the COMP1 for the timer's BRK2 input. COMP1 output is 'ORed' with the other BRK2 sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **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 164: Break and Break2 circuitry overview](#).

23.4.29 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

0001: COMP2 output

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

0001: COMP1 output

Others: Reserved

23.4.30 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 131. TIM1 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 | TIM1_CR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 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 | TIM1_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. | 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 | 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. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 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 | TIM1_DIER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TIM1_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. | 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 | 0 | 0 | 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. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x18 | TIM1_CCMR1 Output Compare 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. | 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 | 0 | 0 | 0 | 0 | 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. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TIM1_CCMR2 Output Compare 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. | 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 | 0 | 0 | 0 | 0 | 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. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TIM1_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. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 131. TIM1 register map and reset values (continued)

| Offset | Register name | Reset value | UIFCP | 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 | 0 | Res. | CNT[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | CNT[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x28 | TIM1_PSC | 0 | Res. | PSC[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | PSC[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2C | TIM1_ARR | 0 | Res. | ARR[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | ARR[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x30 | TIM1_RCR | 0 | Res. | REP[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | REP[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x34 | TIM1_CCR1 | 0 | Res. | CCR1[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | CCR1[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x38 | TIM1_CCR2 | 0 | Res. | CCR2[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | CCR2[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3C | TIM1_CCR3 | 0 | Res. | CCR3[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | CCR3[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x40 | TIM1_CCR4 | 0 | Res. | CCR4[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | CCR4[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x44 | TIM1_BDTR | 0 | Res. | BK2F[3:0] | | | | | BKF[3:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | BK2F[3:0] | | | | | BKF[3:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x48 | TIM1_DCR | 0 | Res. | DBL[4:0] | | | | | DBL[4:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | DBL[4:0] | | | | | DBL[4:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4C | TIM1_DMAR | 0 | DMAB[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | DMAB[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x54 | TIM1_CCMR3 Output Compare mode | 0 | Res. | OC6M[3] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | OC6M[3] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x58 | TIM1_CCR5 | 0 | Res. | CCR5[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | CCR5[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 131. 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 | | | |
|--------|---------------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|
| 0x5C | TIM1_CCR6 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 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. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x68 | TIM1_TISEL | TI4SEL[3:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

24 General-purpose timers (TIM2/TIM3)

24.1 TIM2/TIM3 introduction

The general-purpose timers consist of a 16-bit/32-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 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.

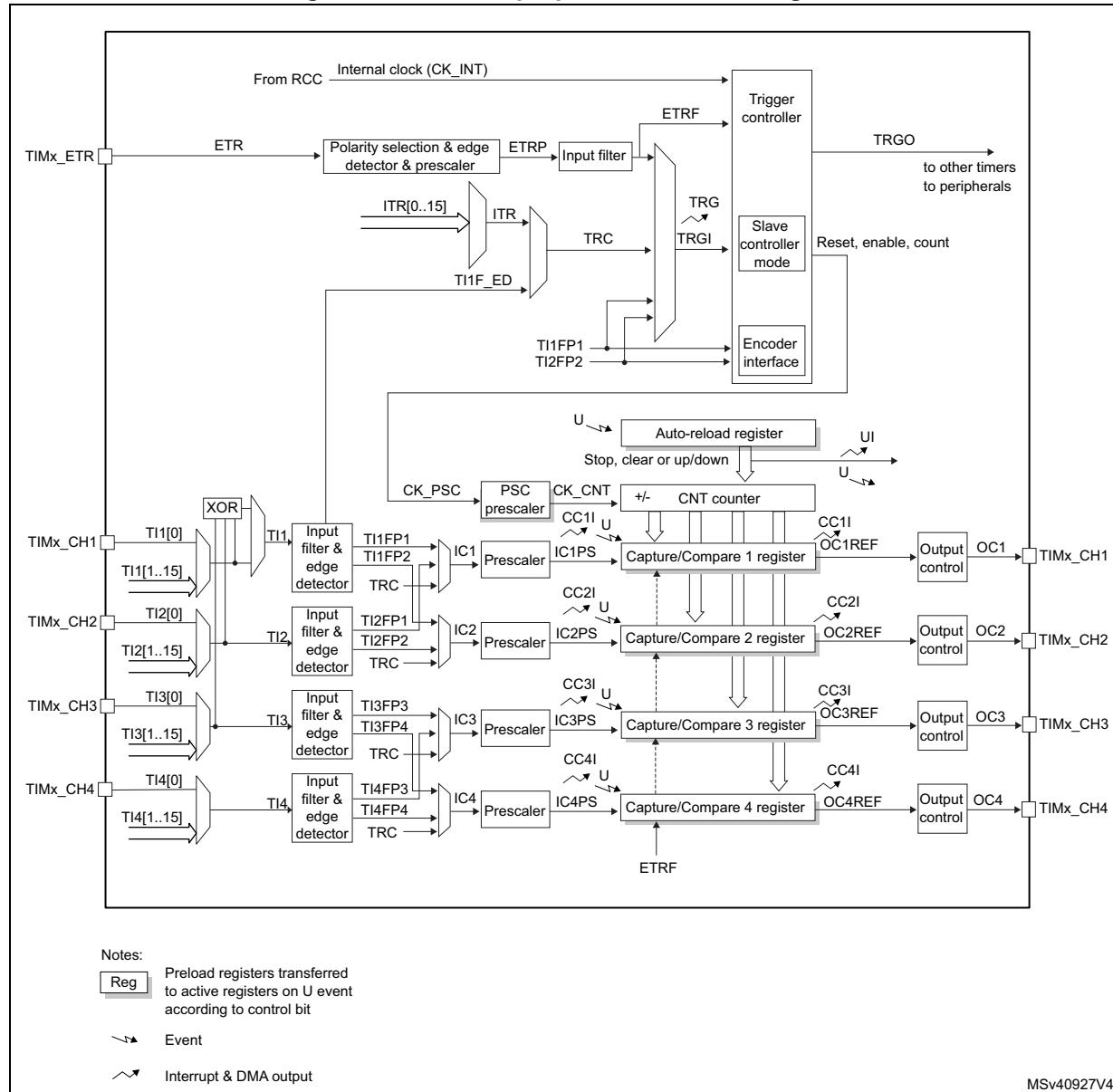
The timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 24.3.19: Timer synchronization](#).

24.2 TIM2/TIM3 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3) or 32-bit (TIM2) 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 181. General-purpose timer block diagram



24.3 TIM2/TIM3 functional description

24.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-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)

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.

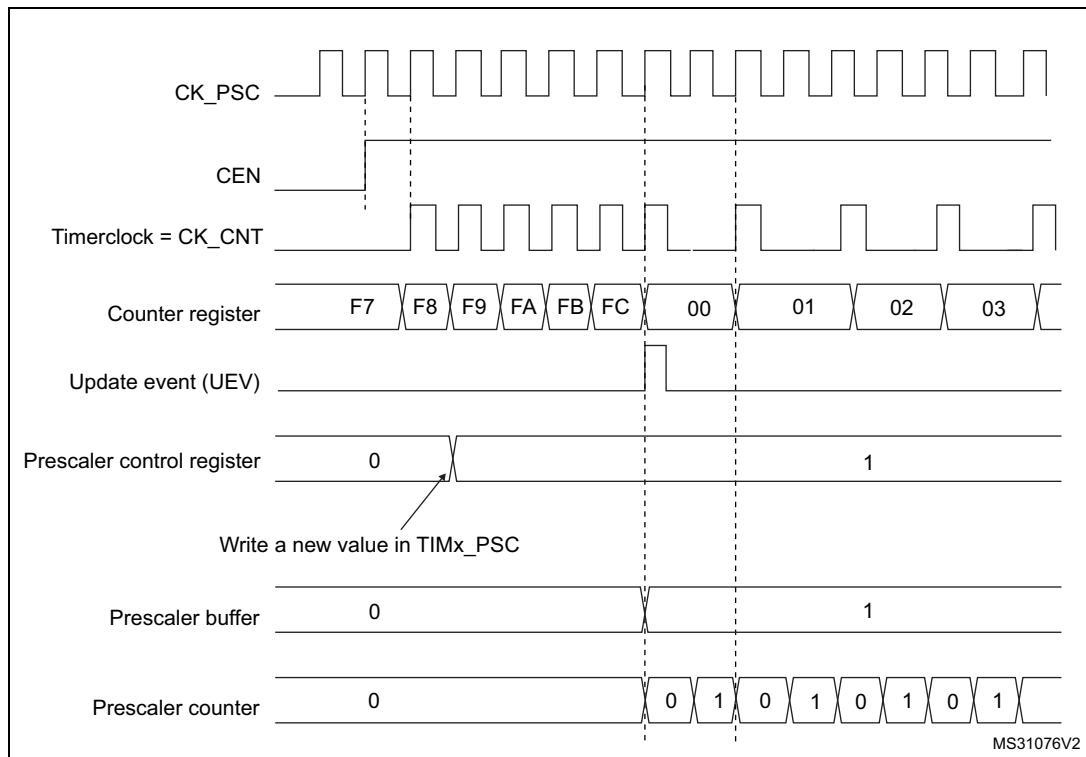
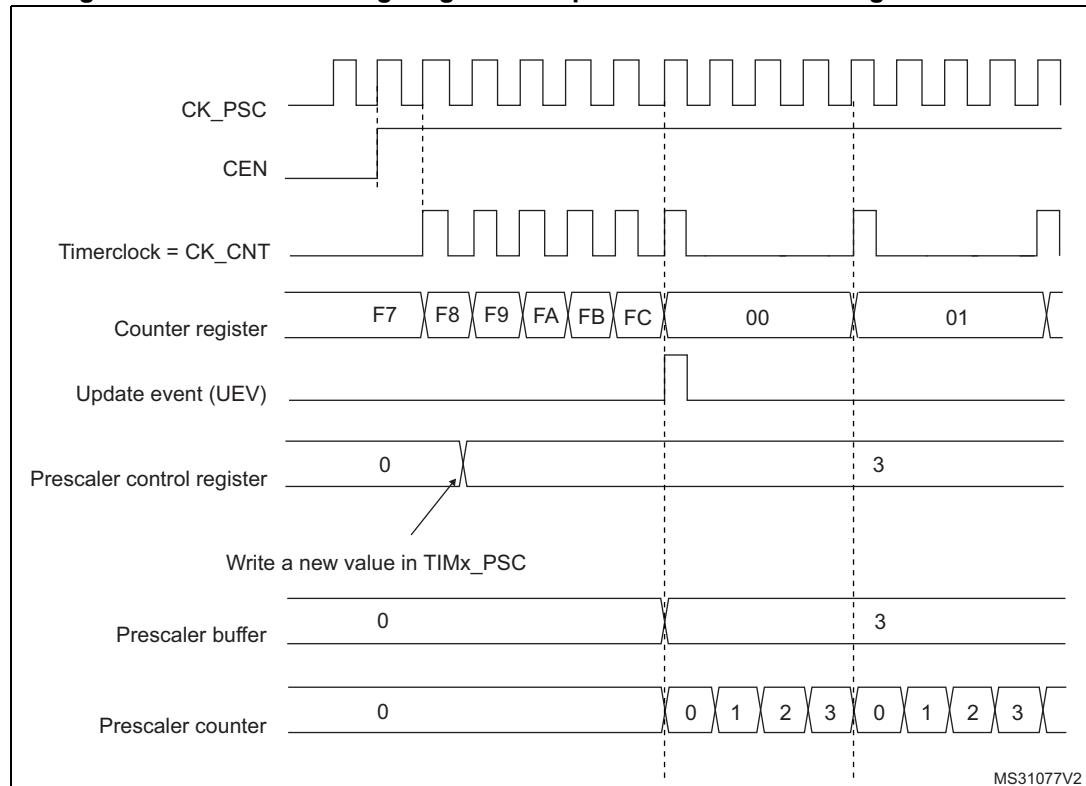
The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

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 182 and *Figure 183* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 182. Counter timing diagram with prescaler division change from 1 to 2**Figure 183. Counter timing diagram with prescaler division change from 1 to 4**

24.3.2 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 184. Counter timing diagram, internal clock divided by 1

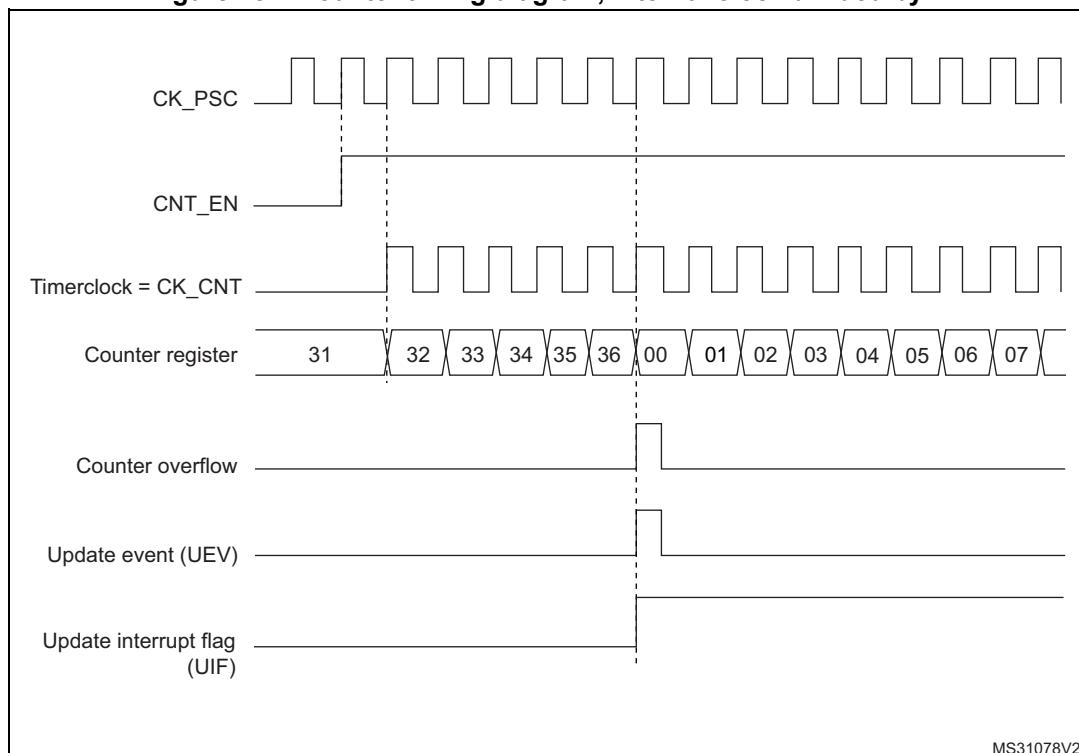


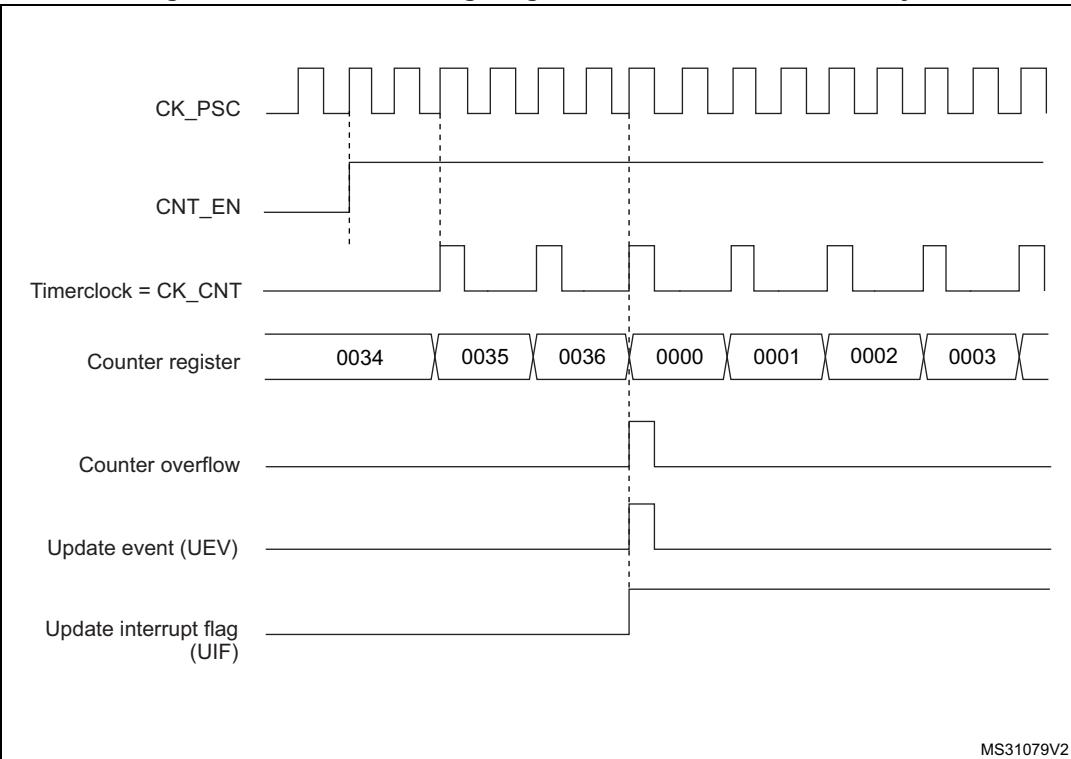
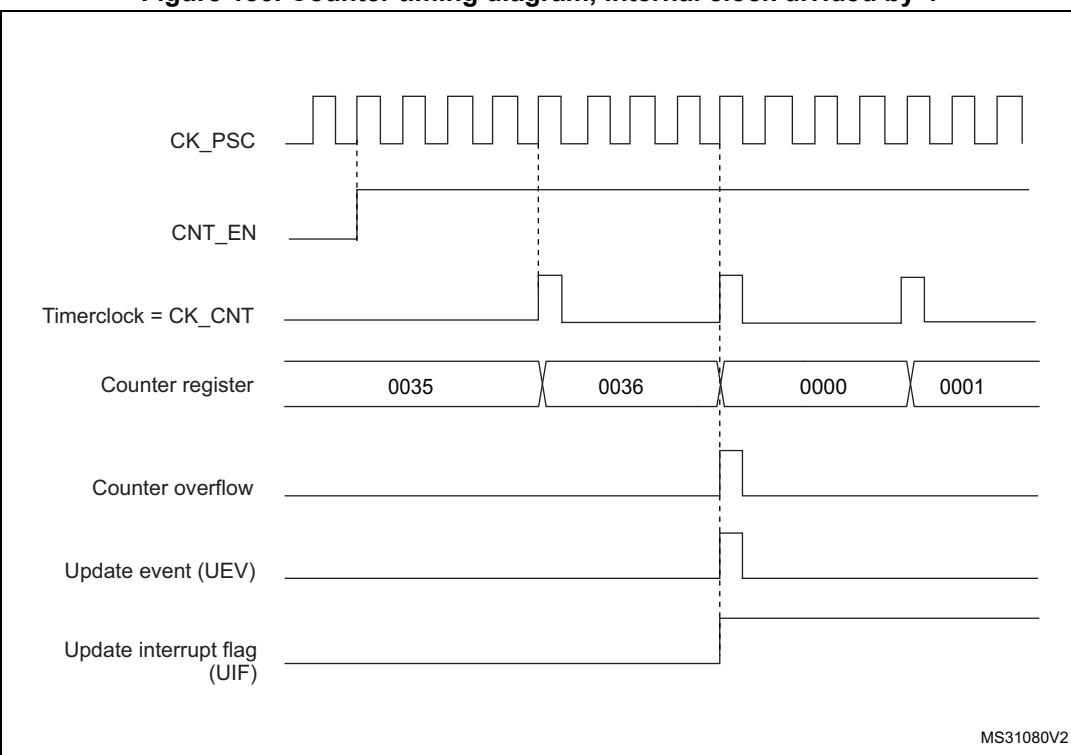
Figure 185. Counter timing diagram, internal clock divided by 2**Figure 186. Counter timing diagram, internal clock divided by 4**

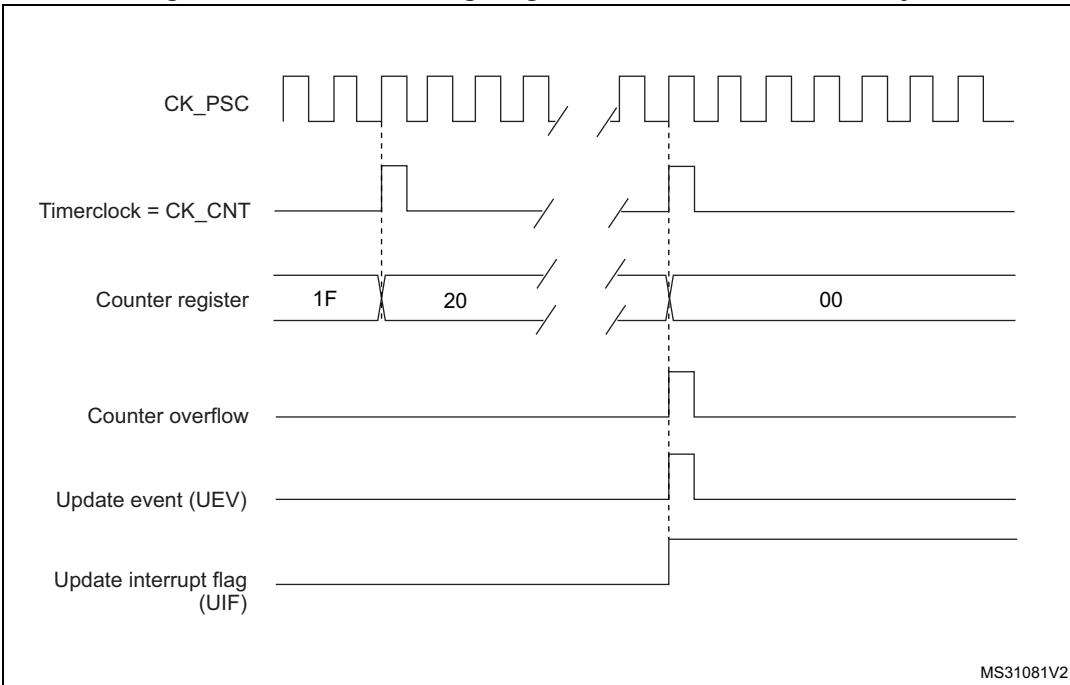
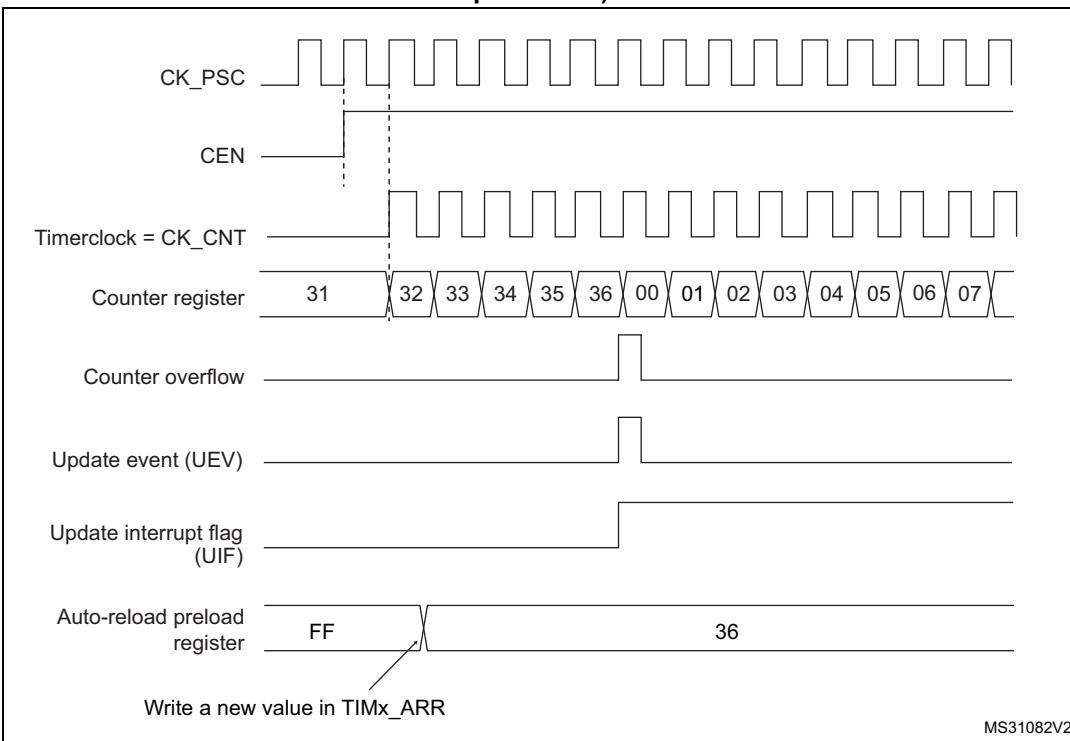
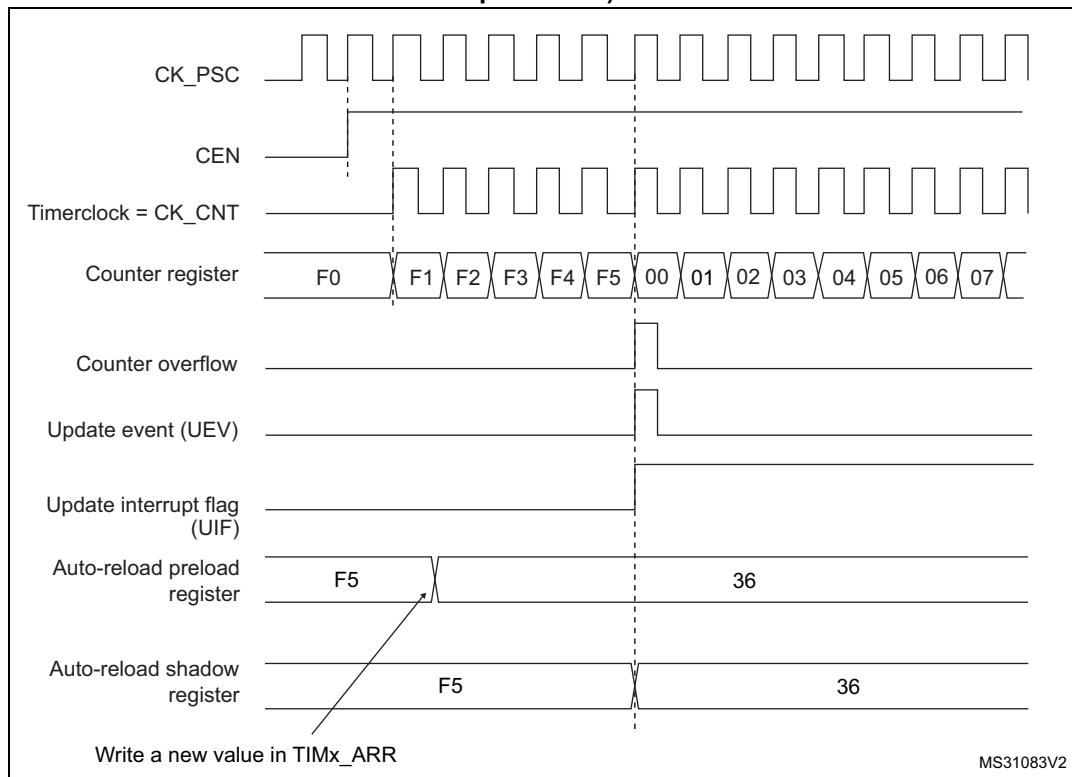
Figure 187. Counter timing diagram, internal clock divided by N**Figure 188. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)**

Figure 189. 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 generated 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 190. Counter timing diagram, internal clock divided by 1

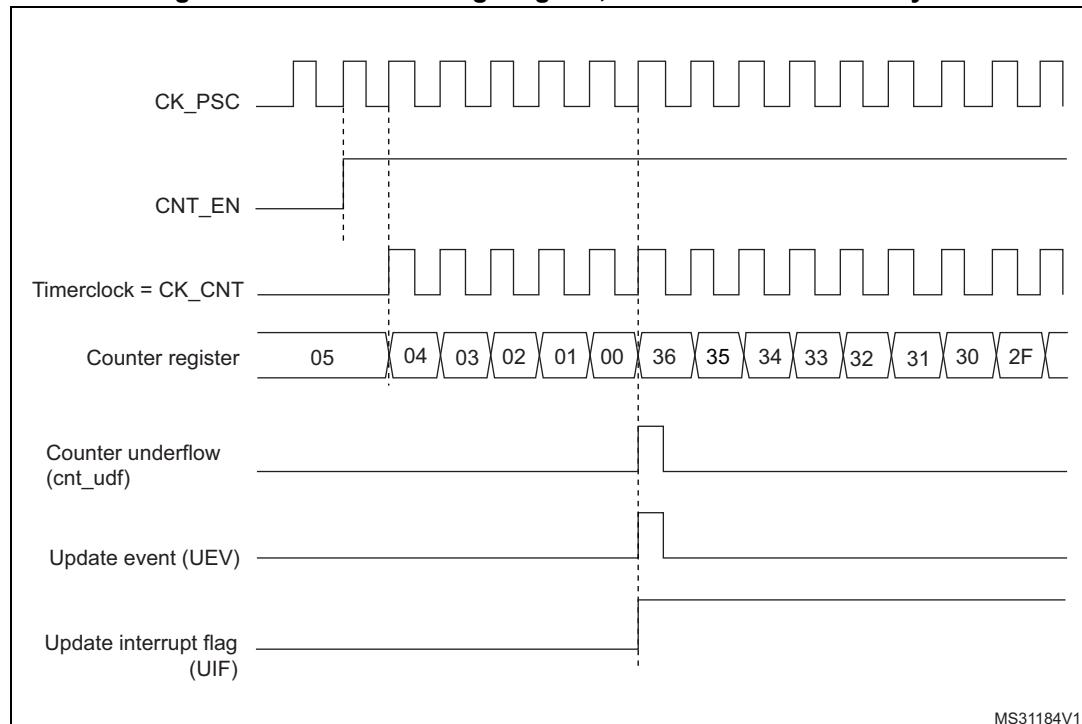


Figure 191. Counter timing diagram, internal clock divided by 2

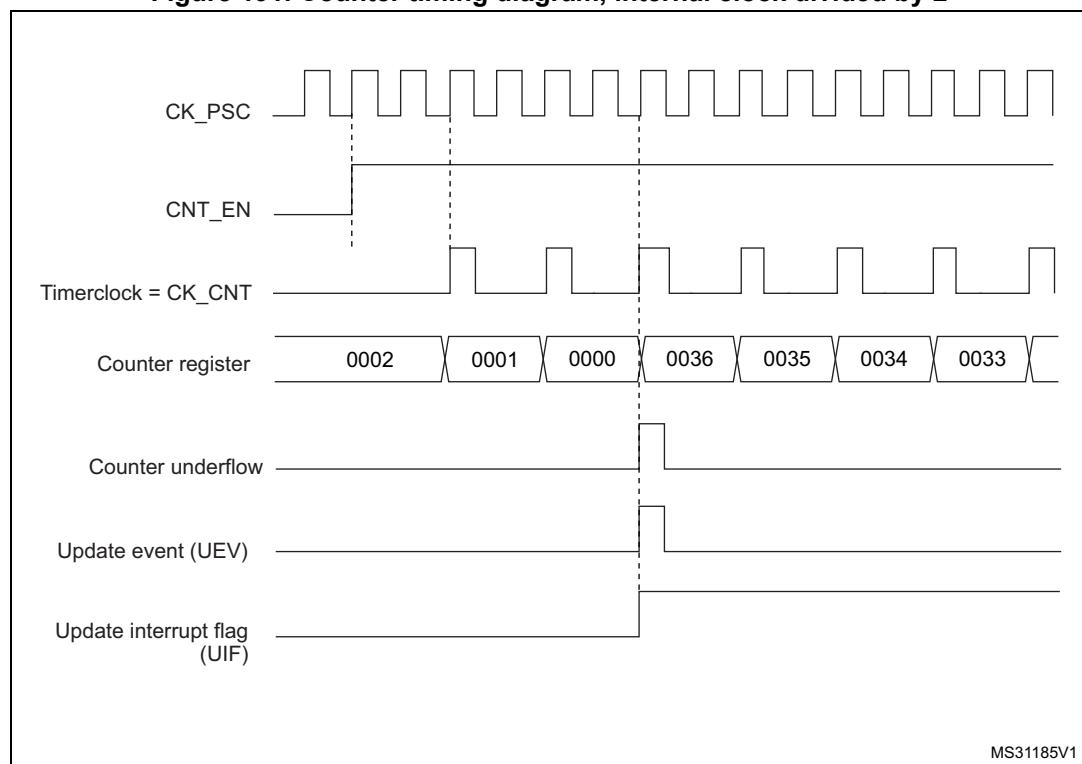


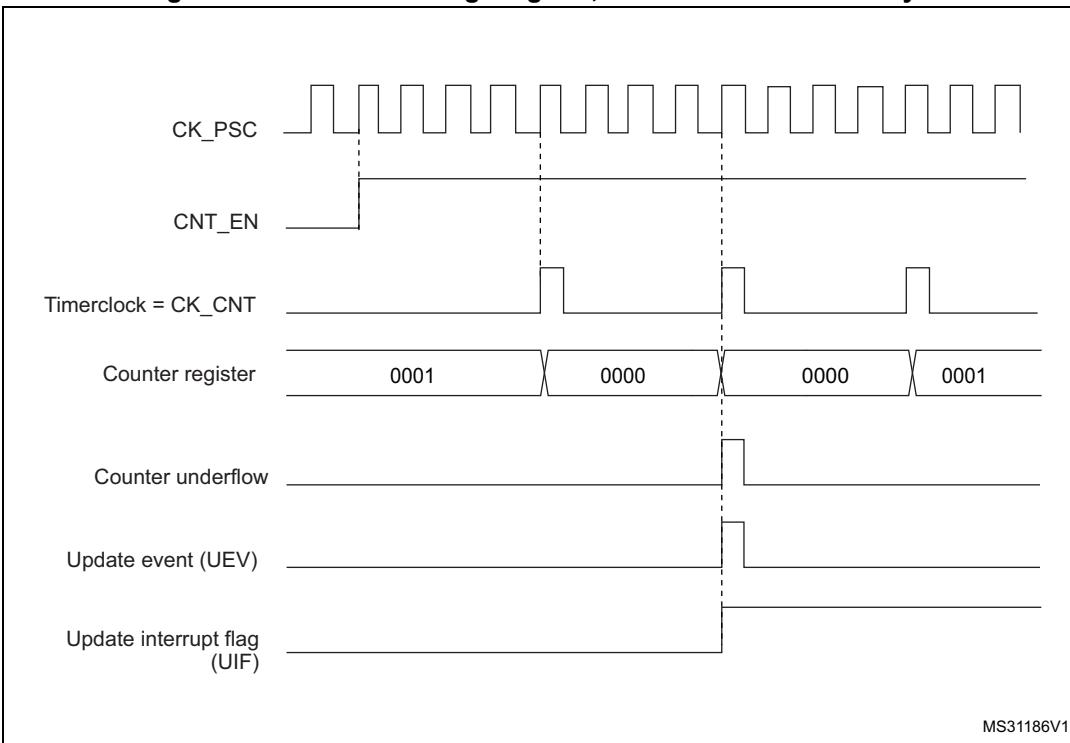
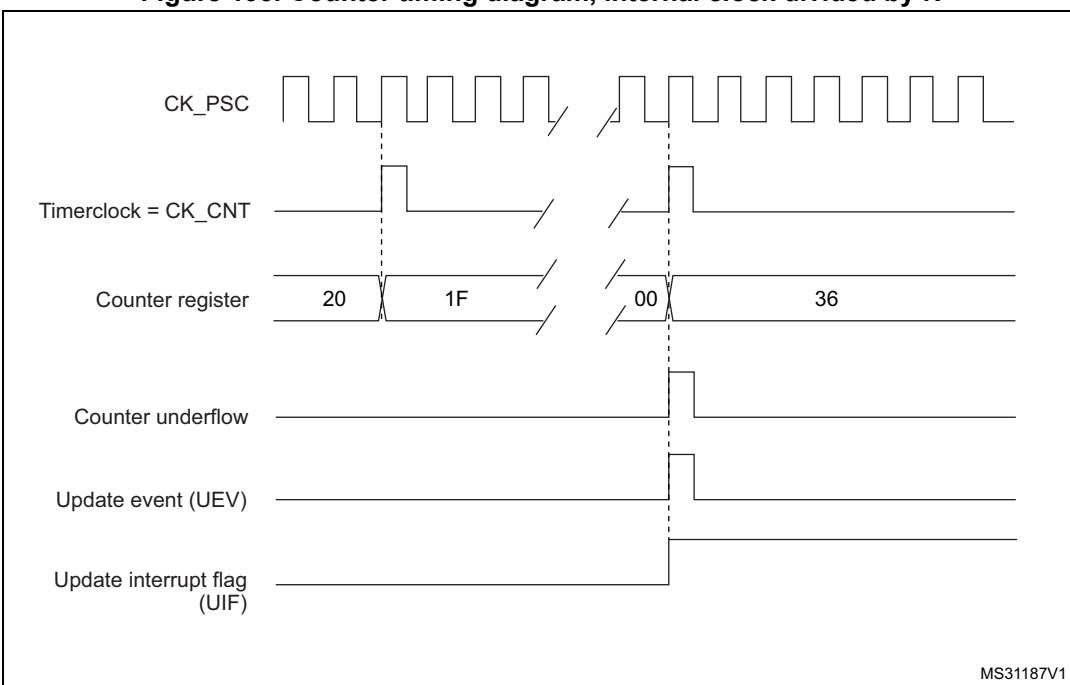
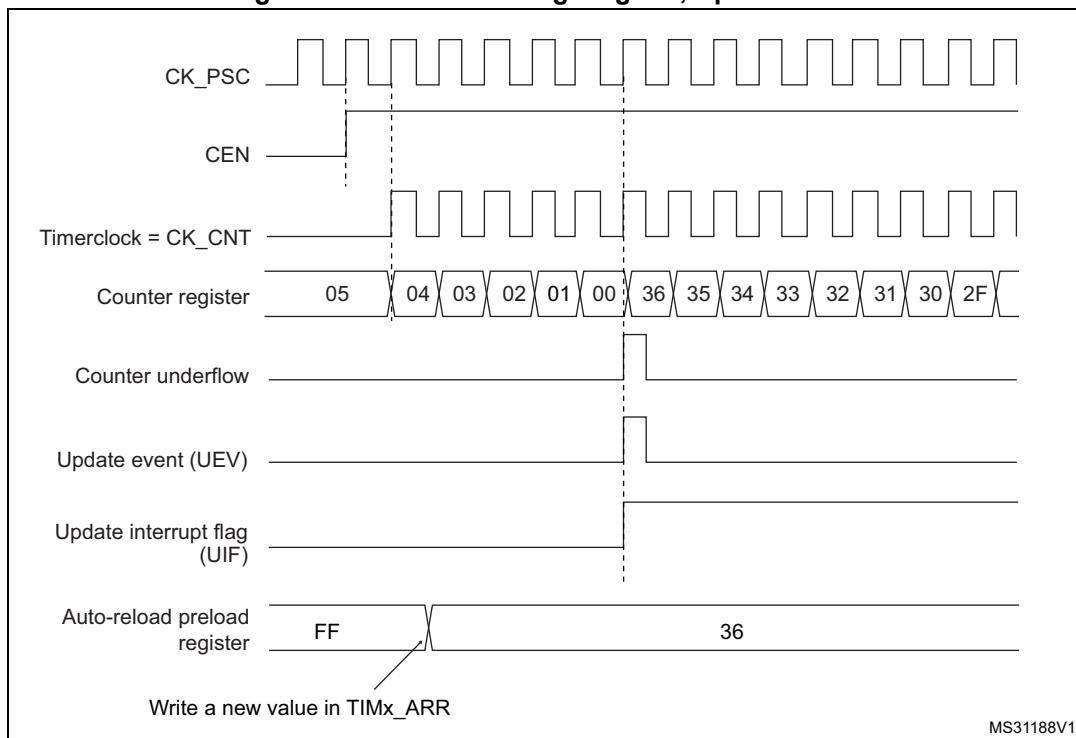
Figure 192. Counter timing diagram, internal clock divided by 4**Figure 193. Counter timing diagram, internal clock divided by N**

Figure 194. Counter timing diagram, Update event

MS31188V1

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") or 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

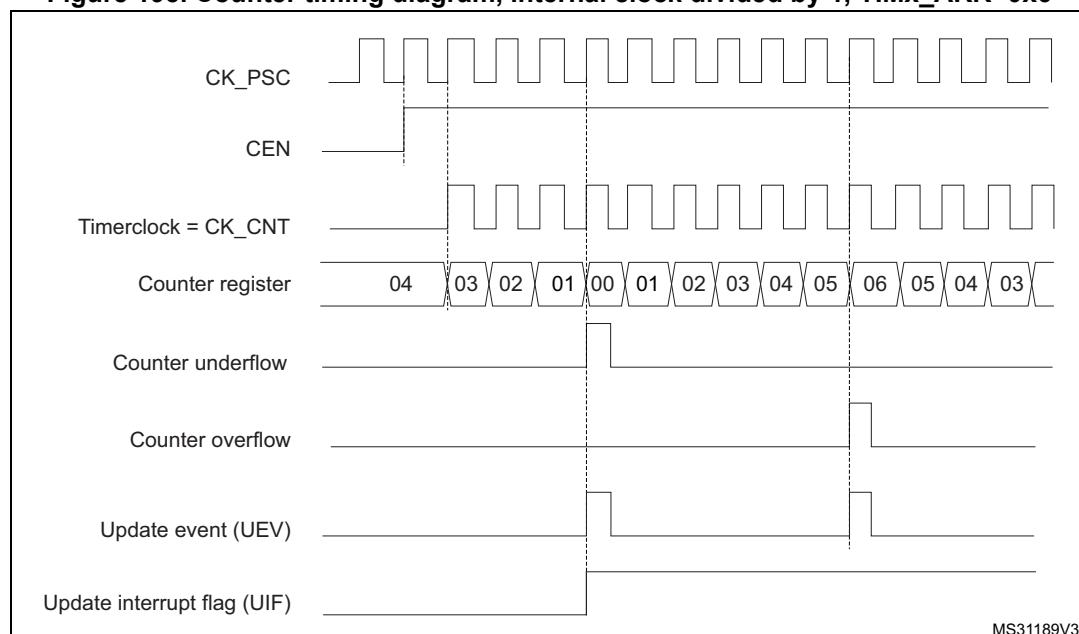
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

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):

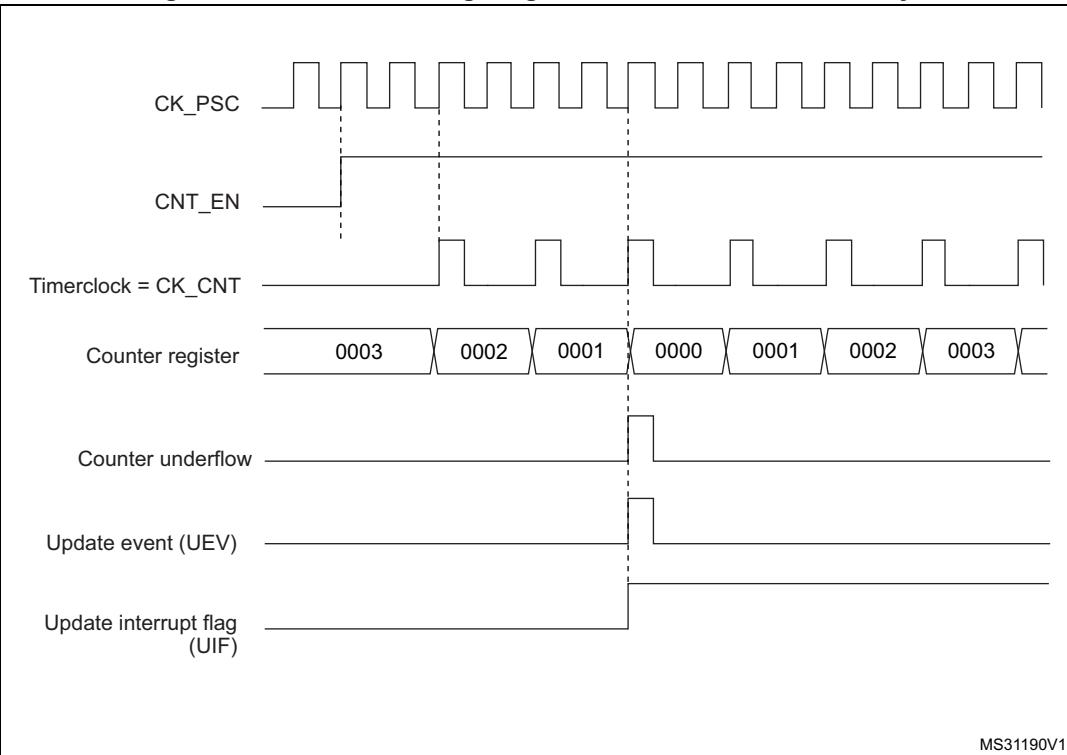
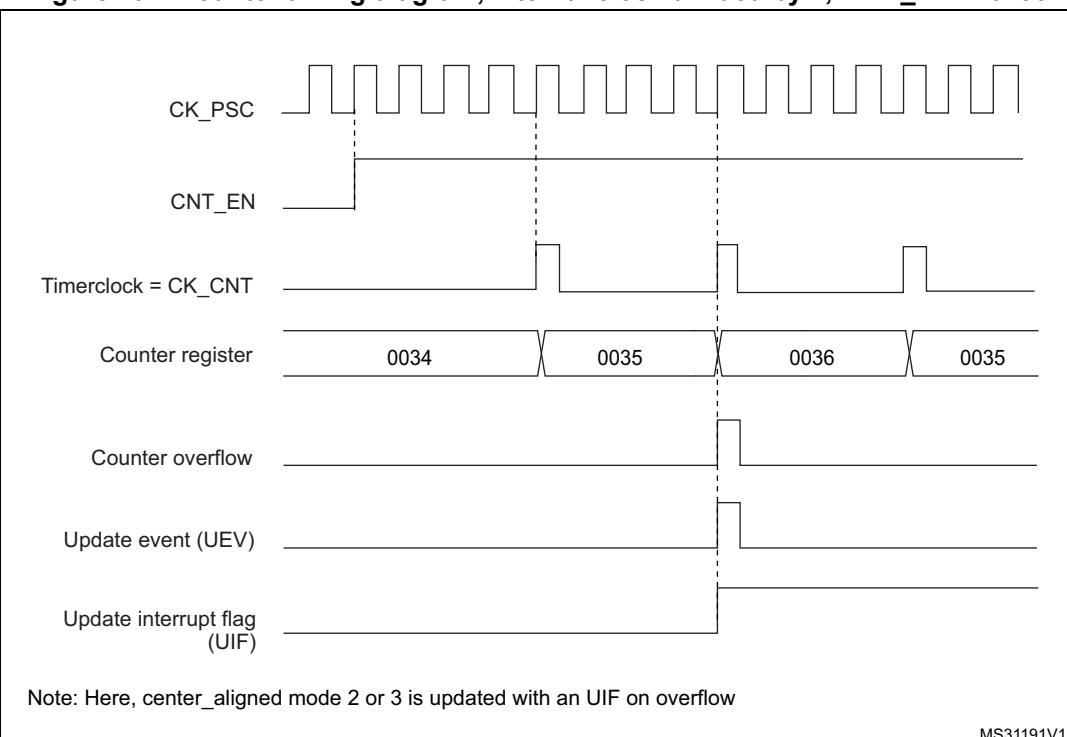
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter 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 195. 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 24.4.1: TIMx control register 1 \(TIMx_CR1\)\(x = 2 to 3\) on page 696](#)).

Figure 196. Counter timing diagram, internal clock divided by 2**Figure 197. 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.

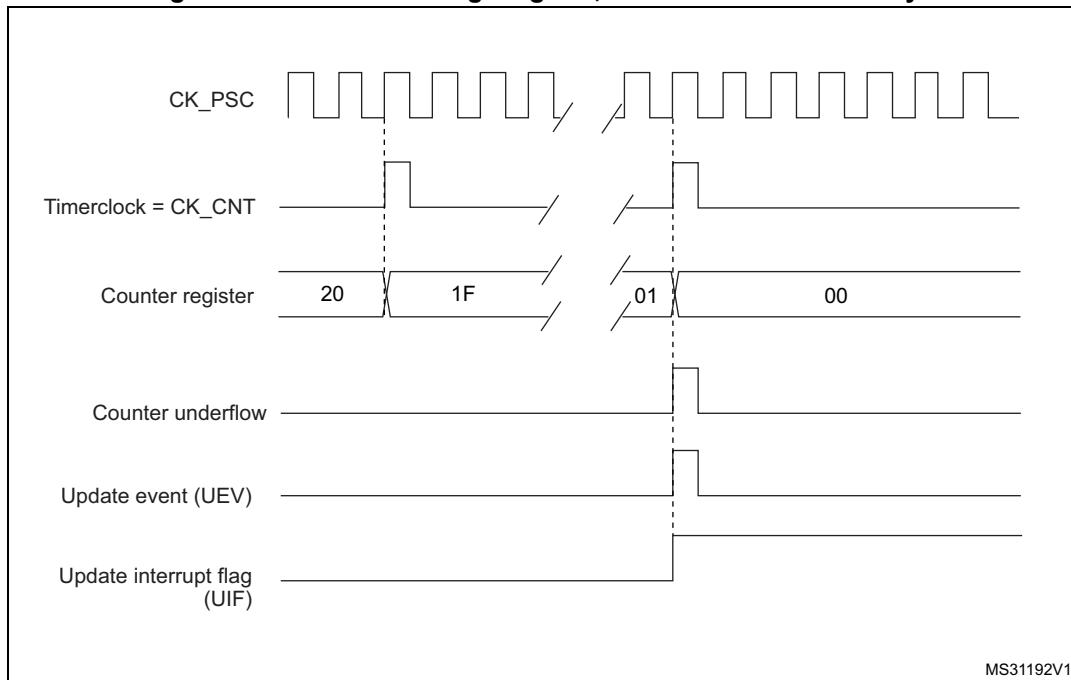
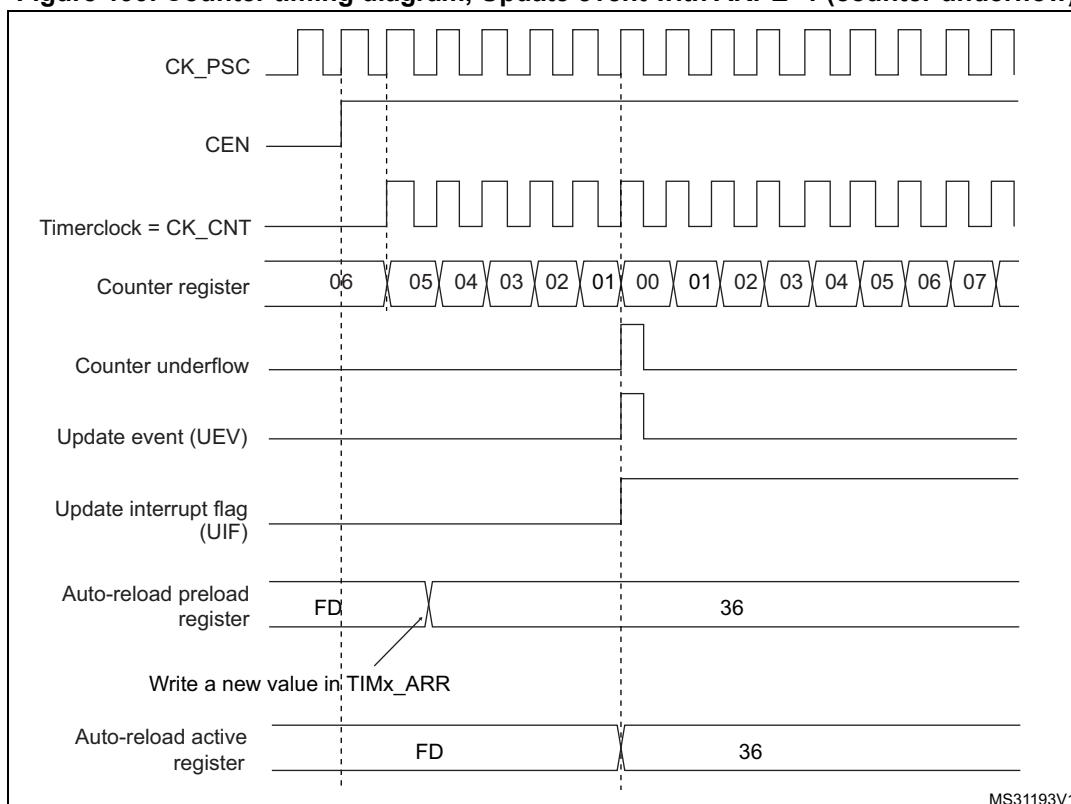
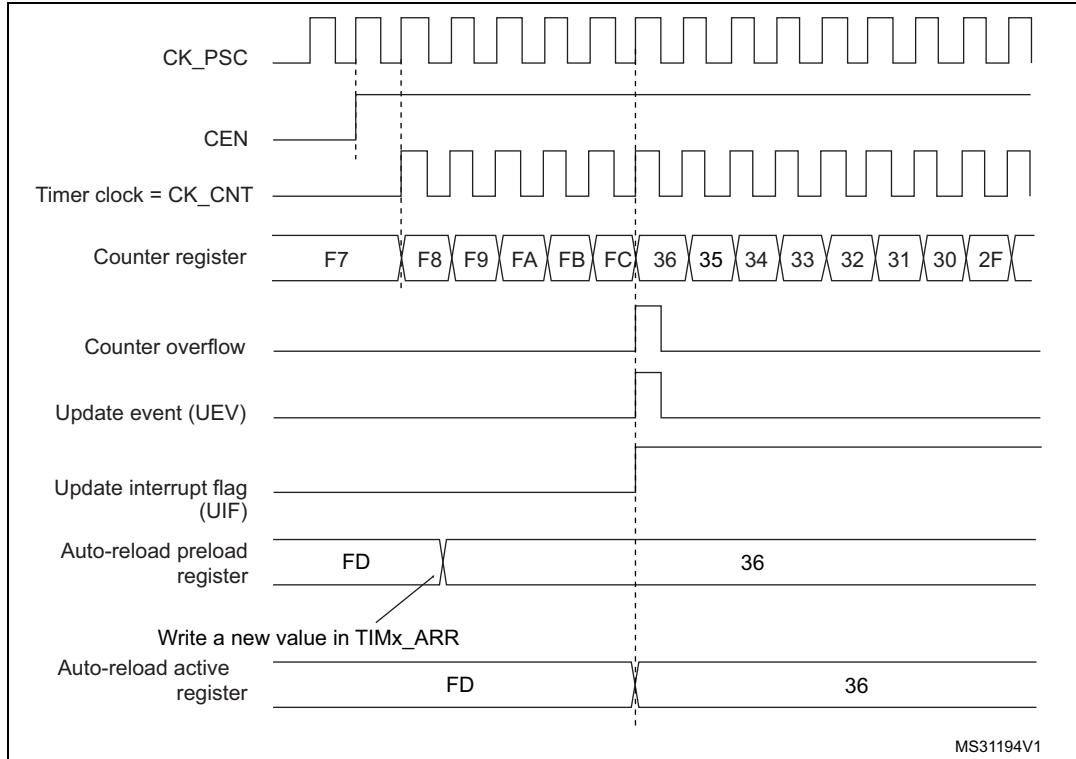
Figure 198. Counter timing diagram, internal clock divided by N**Figure 199. Counter timing diagram, Update event with ARPE=1 (counter underflow)**

Figure 200. Counter timing diagram, Update event with ARPE=1 (counter overflow)

24.3.3 Clock selection

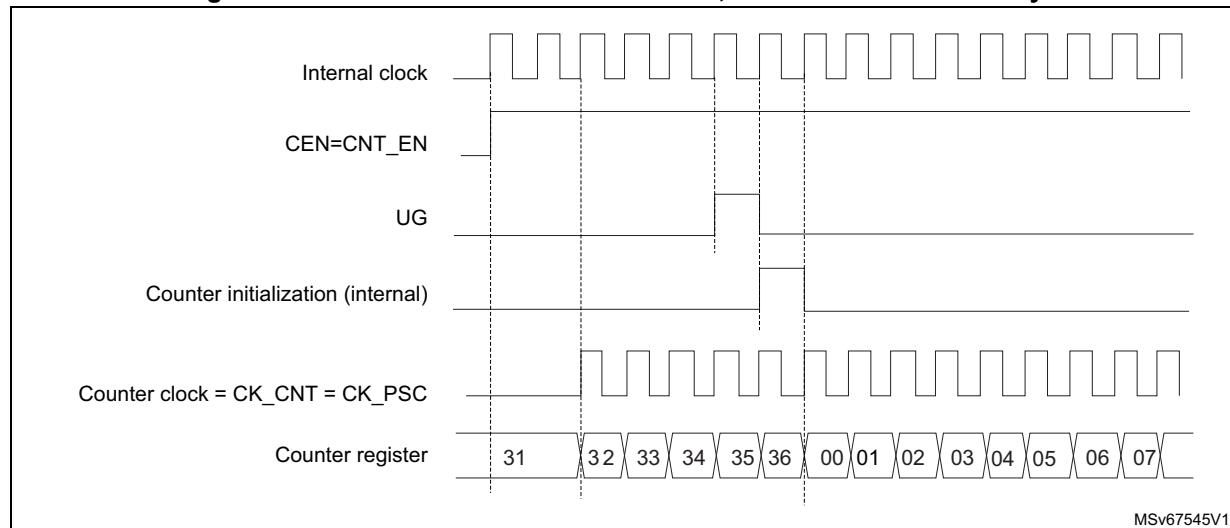
The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin (TI_x)
- 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 690](#) for more details.

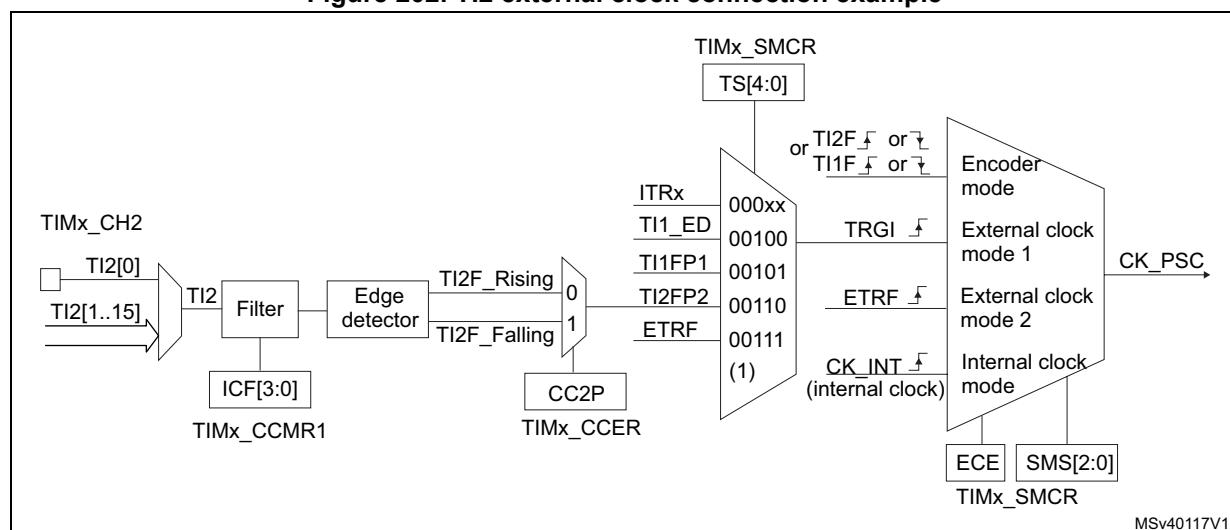
Internal clock source (CK_INT)

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 201](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 201. 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 202. 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).

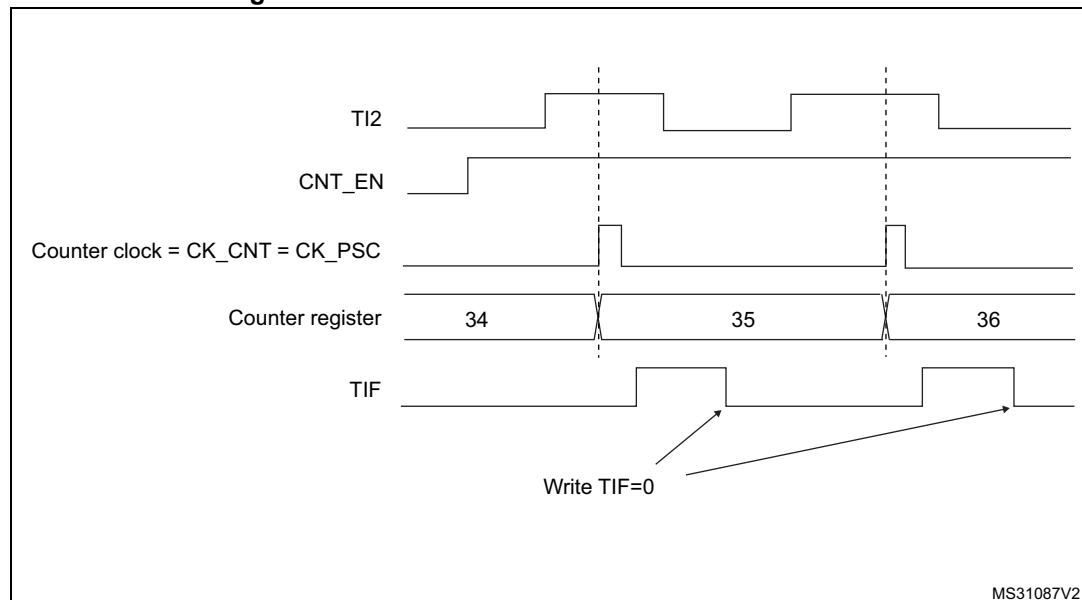
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

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 203. Control circuit in external clock mode 1



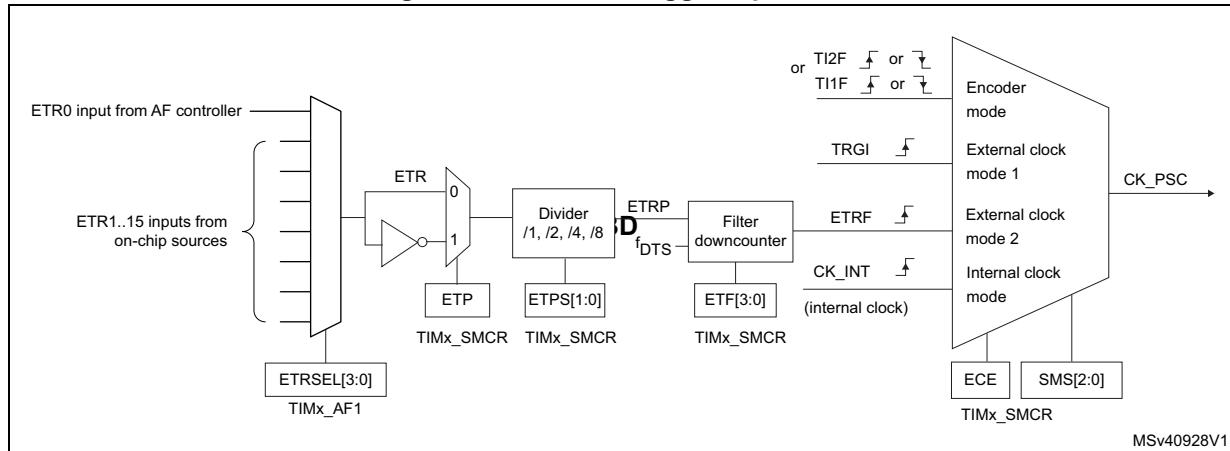
External clock source mode 2

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 204](#) gives an overview of the external trigger input block.

Figure 204. External trigger input block



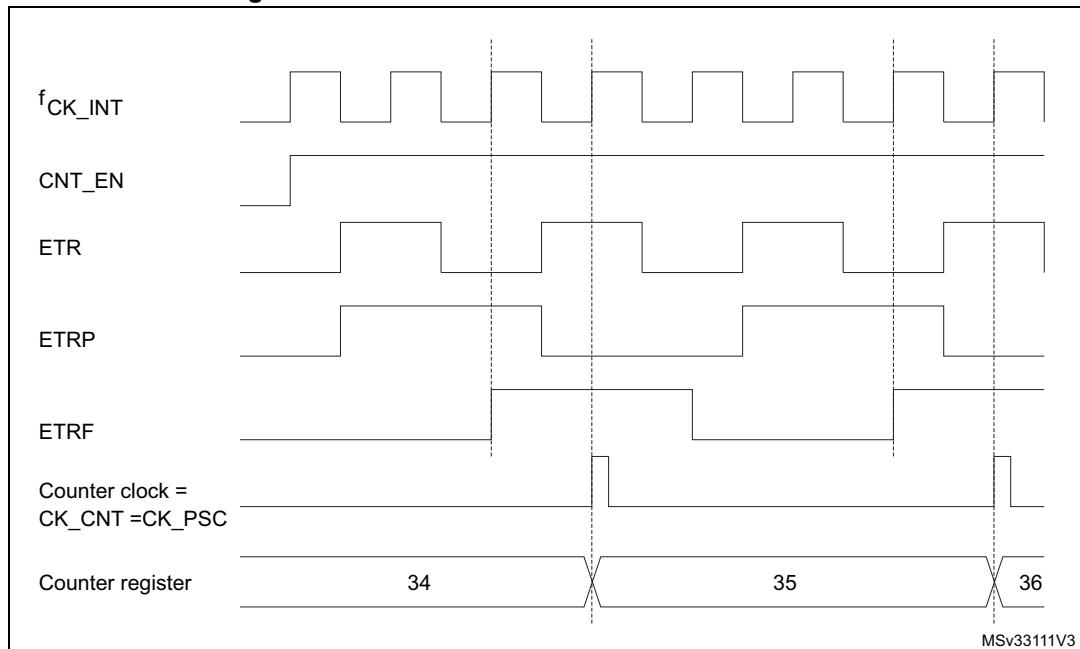
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
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 ETSPS[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 $\frac{1}{4}$ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETSPS prescaler setting.

Figure 205. Control circuit in external clock mode 2



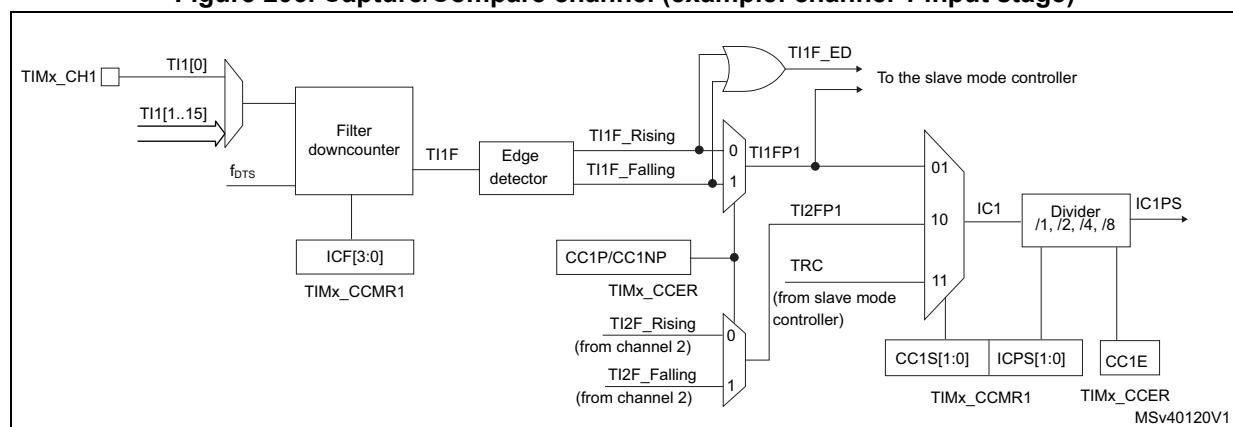
24.3.4 Capture/Compare channels

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 TIx_F . Then, an edge detector with polarity selection generates a signal (TIx_FPx) 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 206. 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 207. Capture/Compare channel 1 main circuit

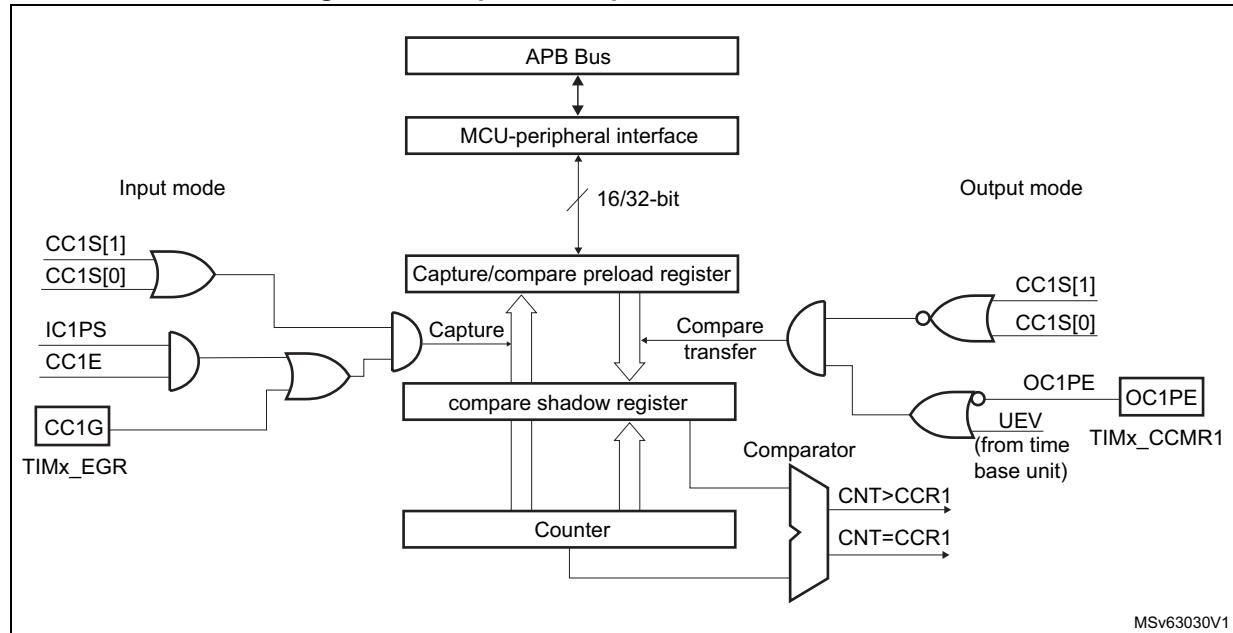
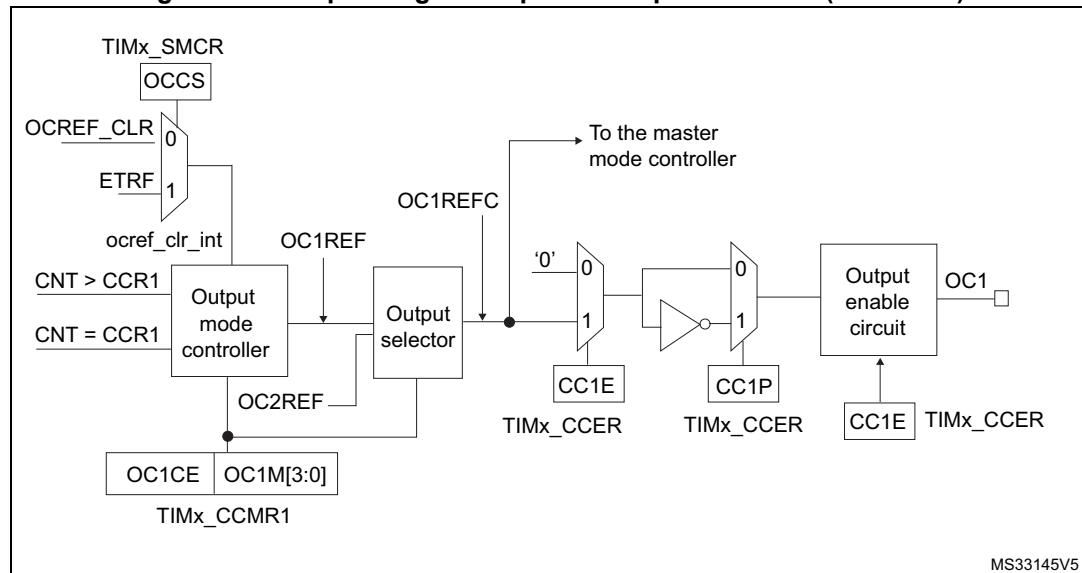


Figure 208. Output stage of Capture/Compare channel (channel 1)



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.

24.3.5 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 IC_x signal. When a capture occurs, the corresponding CC_xIF 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 CC_xIF flag was already high, then the over-capture flag CC_xOF (TIMx_SR register) is set. CC_xIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CC_xOF 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 TI_x (IC_xF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 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 CC_xG bit in the TIMx_EGR register.*

24.3.6 PWM input mode

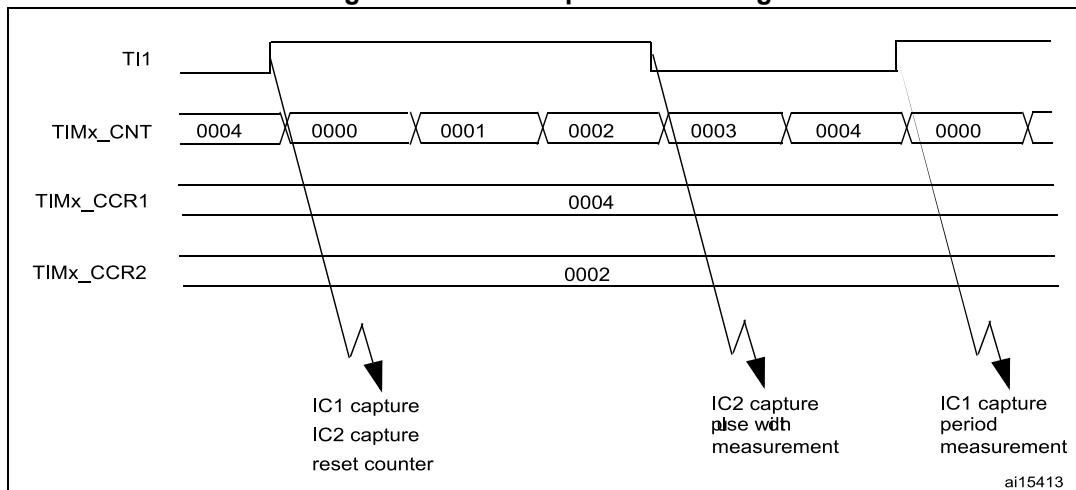
This mode is a particular case of input capture mode. The procedure is the same except:

- 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 209. 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.

24.3.7 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.

24.3.8 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.

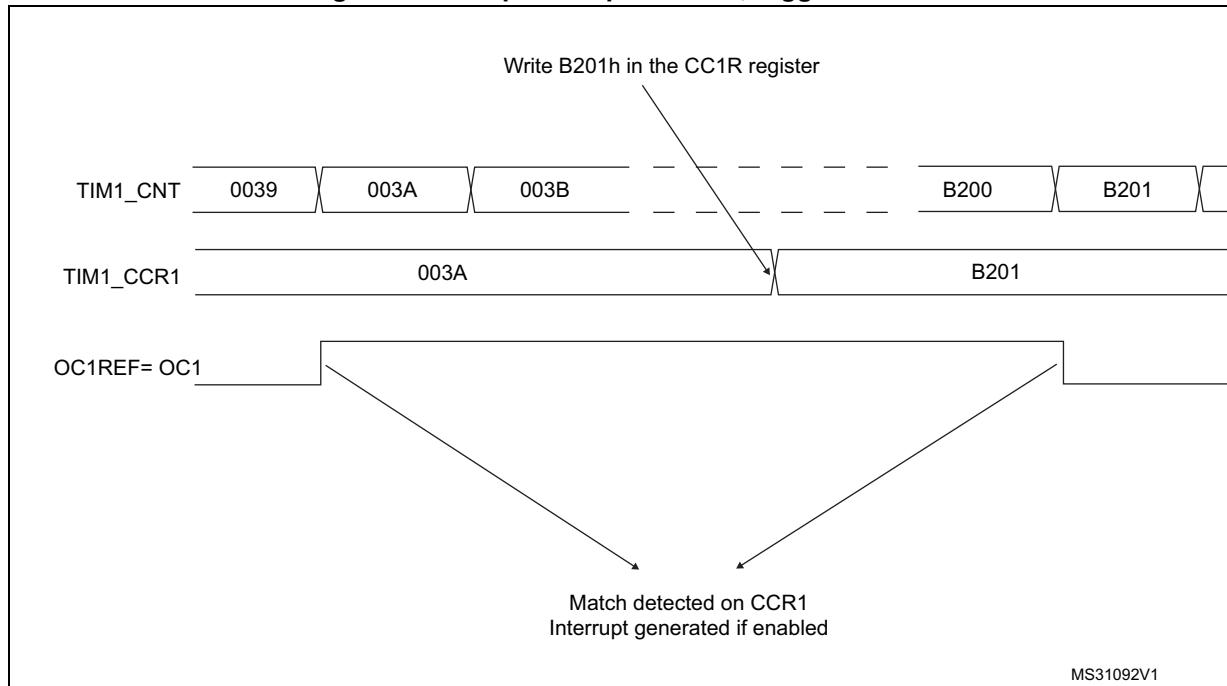
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 210](#).

Figure 210. Output compare mode, toggle on OC1



24.3.9 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 \leq TIMx_CNT or TIMx_CNT \leq 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.

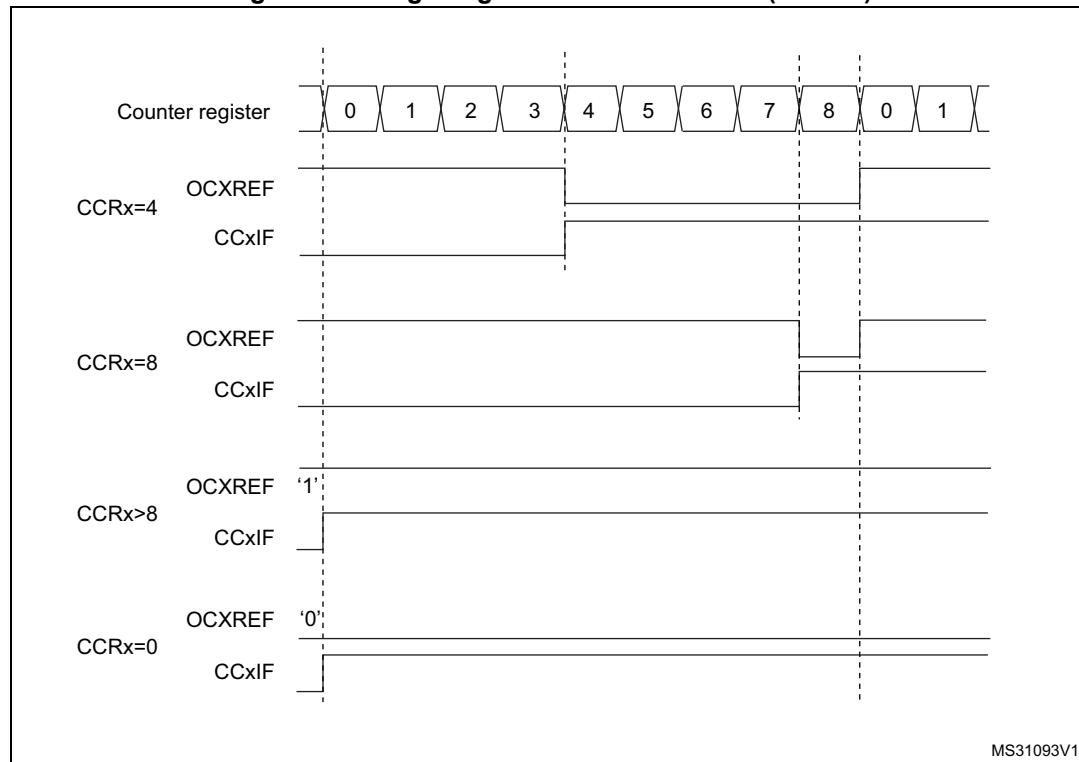
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to [Upcounting mode on page 655](#).

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 211](#) shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 211. Edge-aligned PWM waveforms (ARR=8)



MS31093V1

Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to [Downcounting mode on page 658](#).

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.

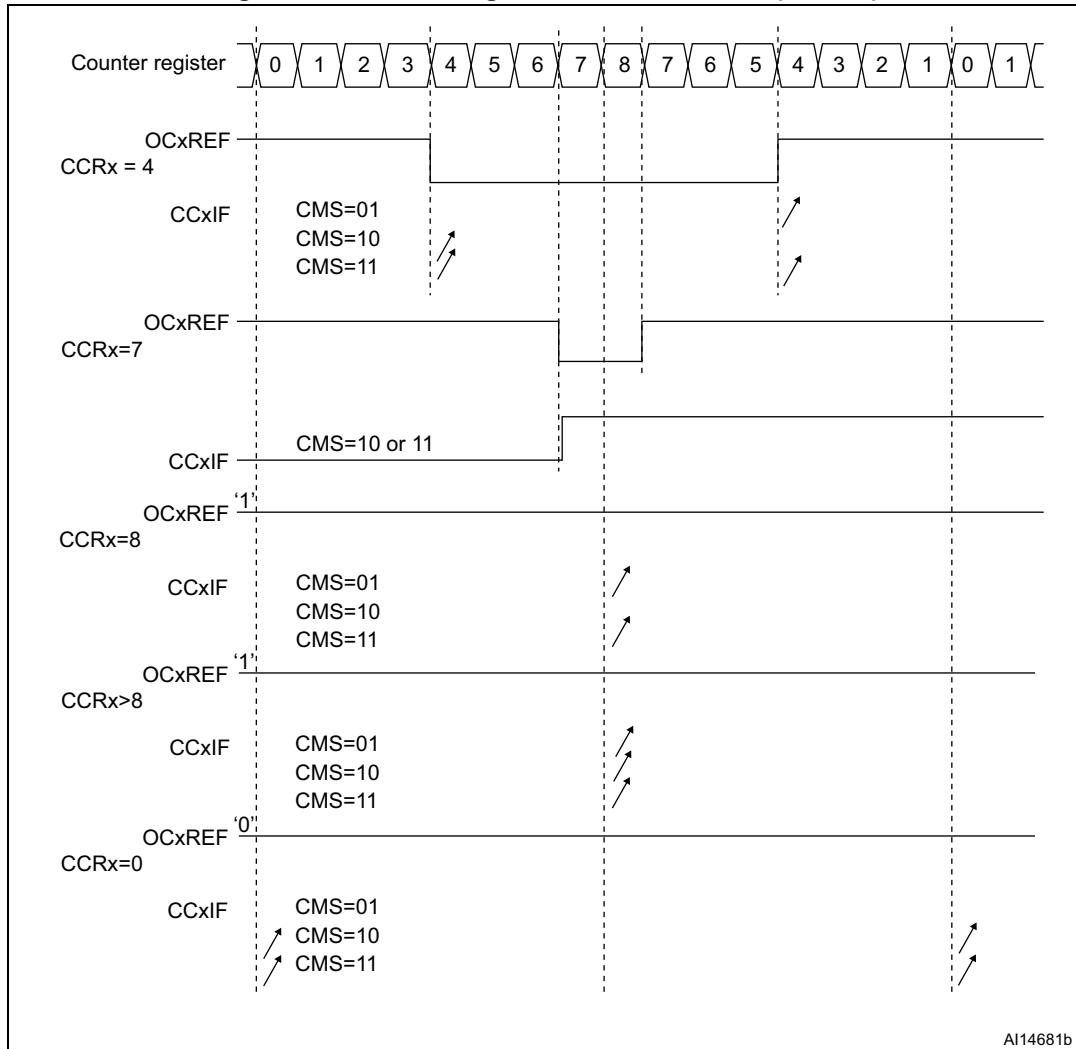
PWM center-aligned mode

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 661](#).

Figure 212 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 212. 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.

24.3.10 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

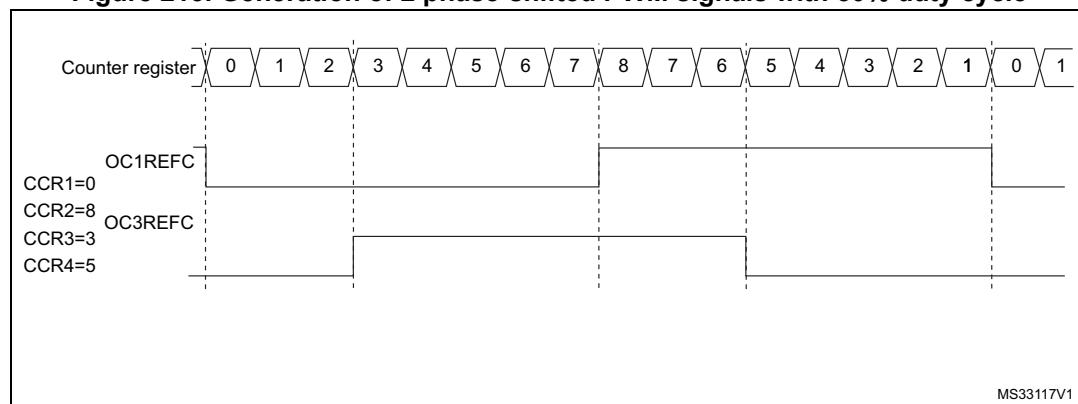
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

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 213 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 213. Generation of 2 phase-shifted PWM signals with 50% duty cycle



24.3.11 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.

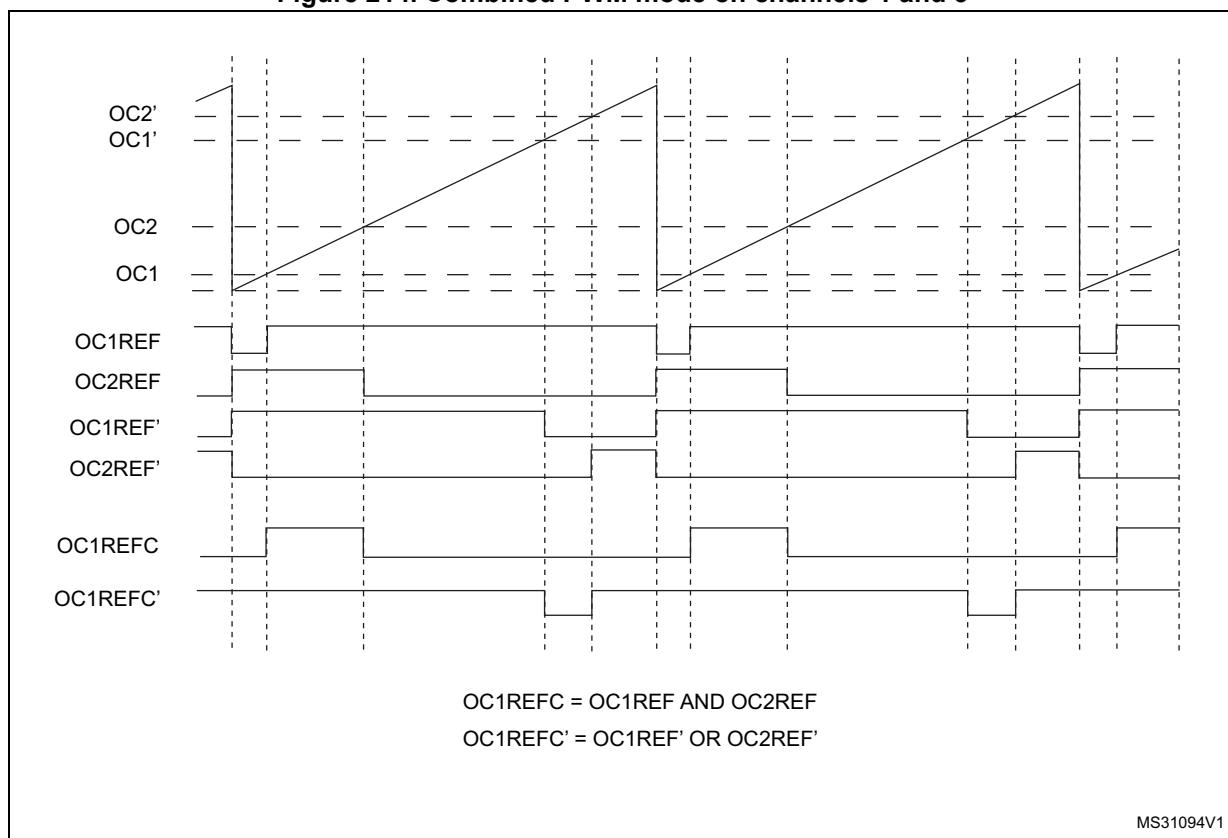
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 214 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 214. Combined PWM mode on channels 1 and 3



24.3.12 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.

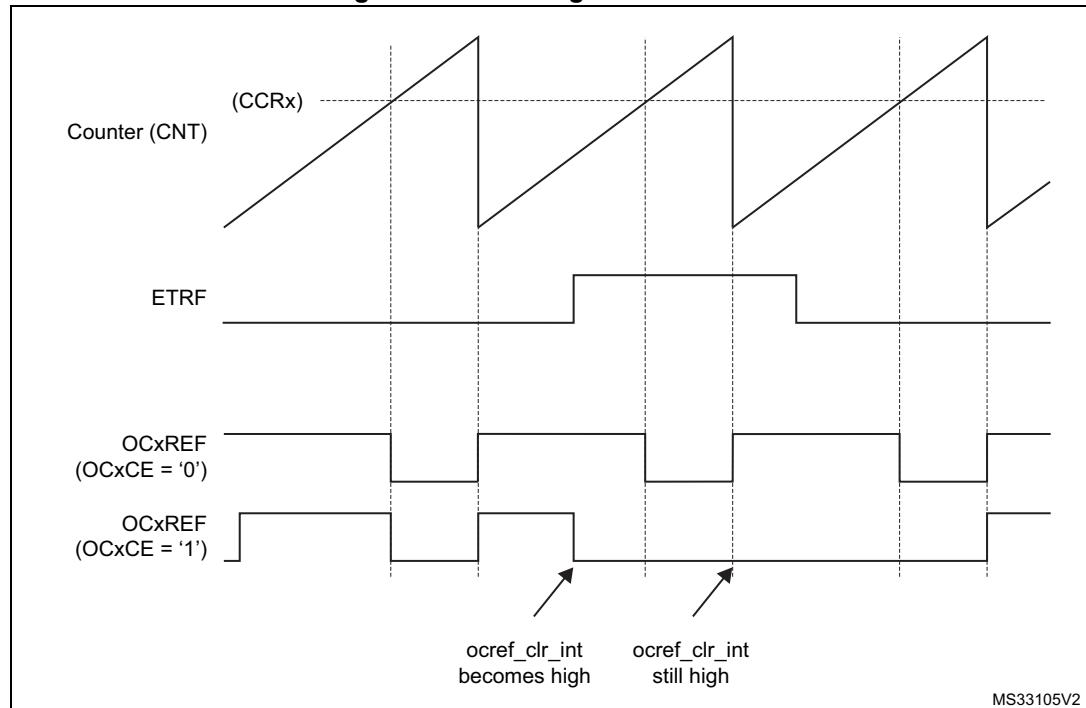
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

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 215 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 215. Clearing TIMx OCxREF



Note:

In case of a PWM with a 100% duty cycle (if CCR_x>ARR), OCxREF is enabled again at the next counter overflow.

24.3.13 One-pulse mode

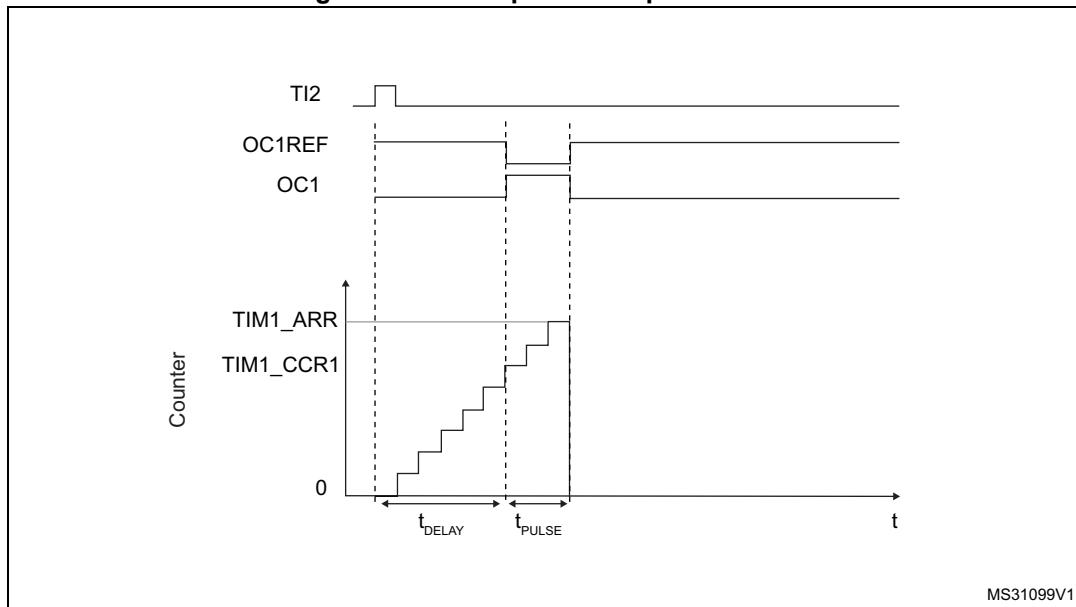
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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<CCR_x ≤ ARR (in particular, 0<CCR_x),

Figure 216. 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.

24.3.14 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 24.3.13](#):

- 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.

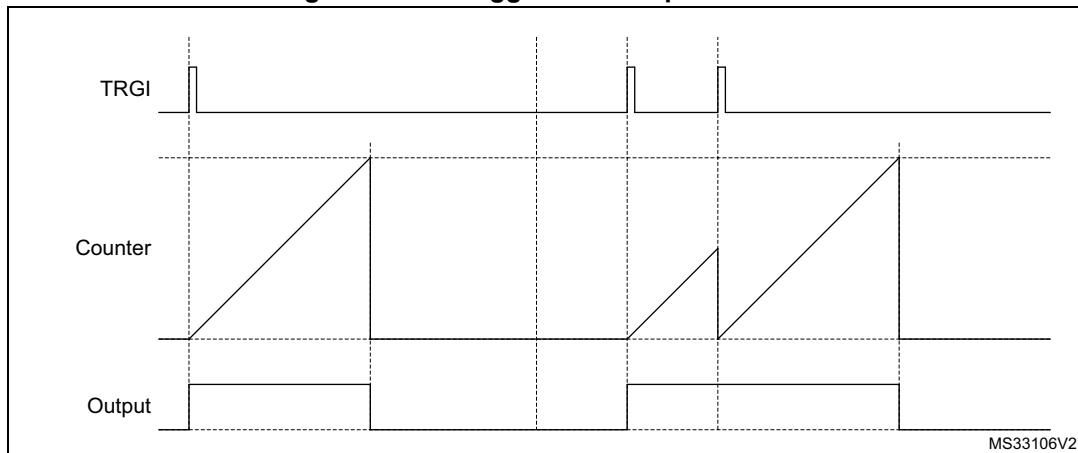
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

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 217. Retriggerable one-pulse mode



24.3.15 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 132](#). 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 132. 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 218 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)

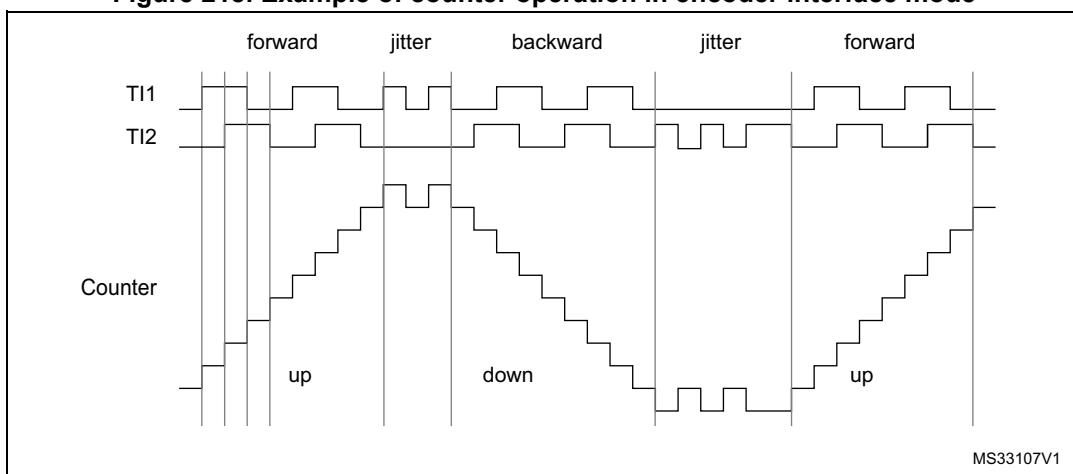
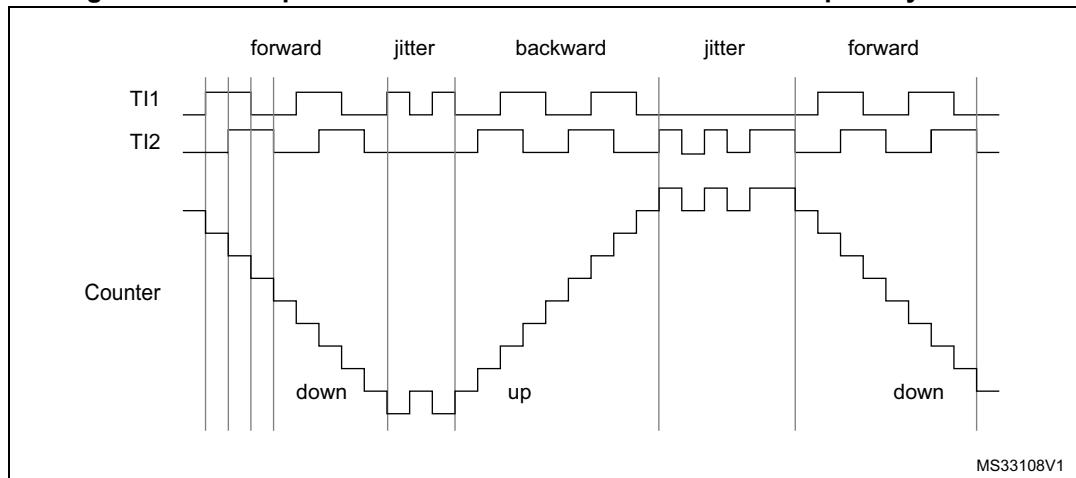
Figure 218. Example of counter operation in encoder interface mode

Figure 219 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 219. 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.

24.3.16 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).

24.3.17 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 23.3.25: Interfacing with Hall sensors on page 602](#).

24.3.18 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.

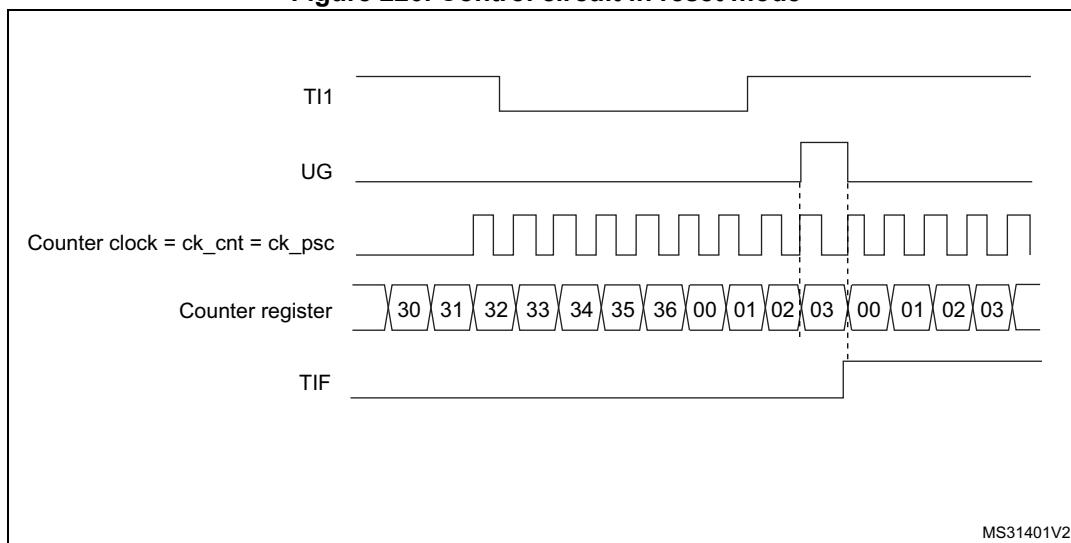
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

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, 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 220. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

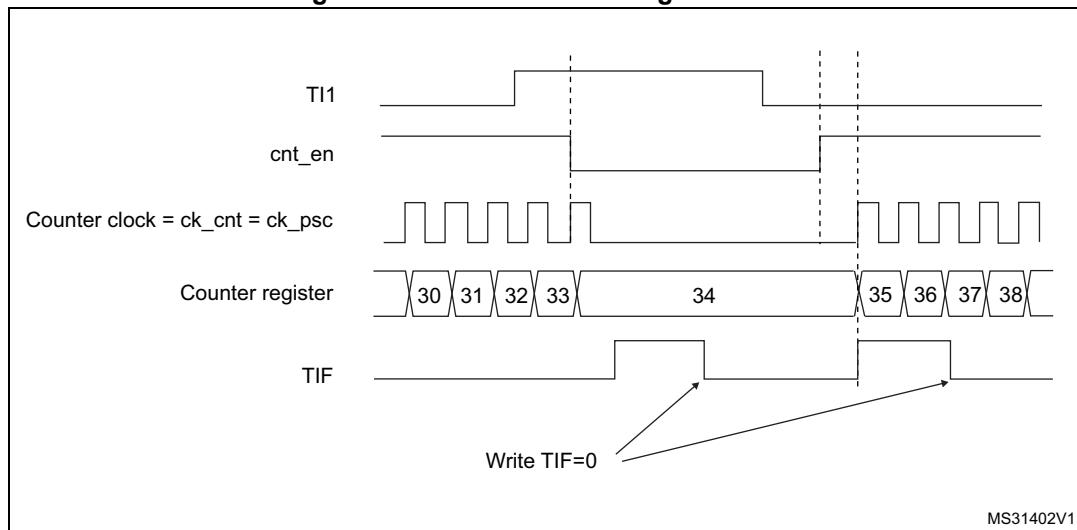
In the following example, the upcounter counts only when TI1 input is low:

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 221. 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

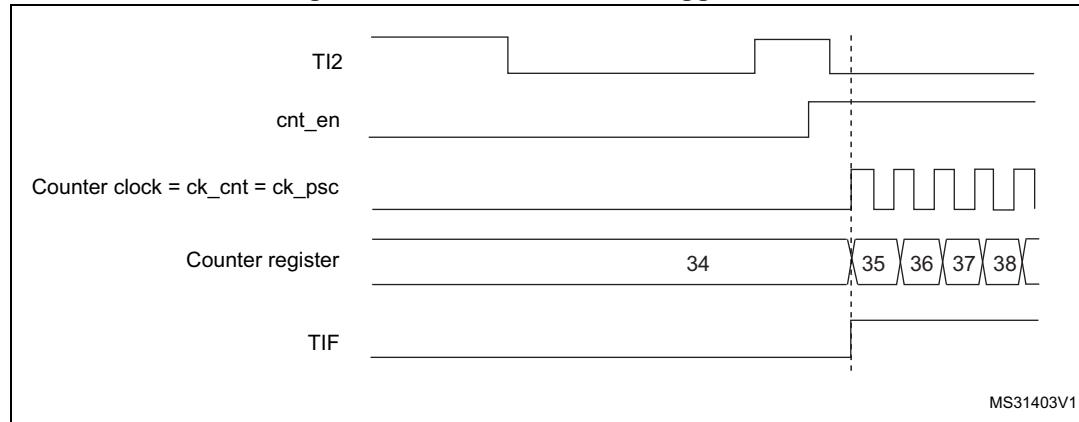
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

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 222. 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 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.

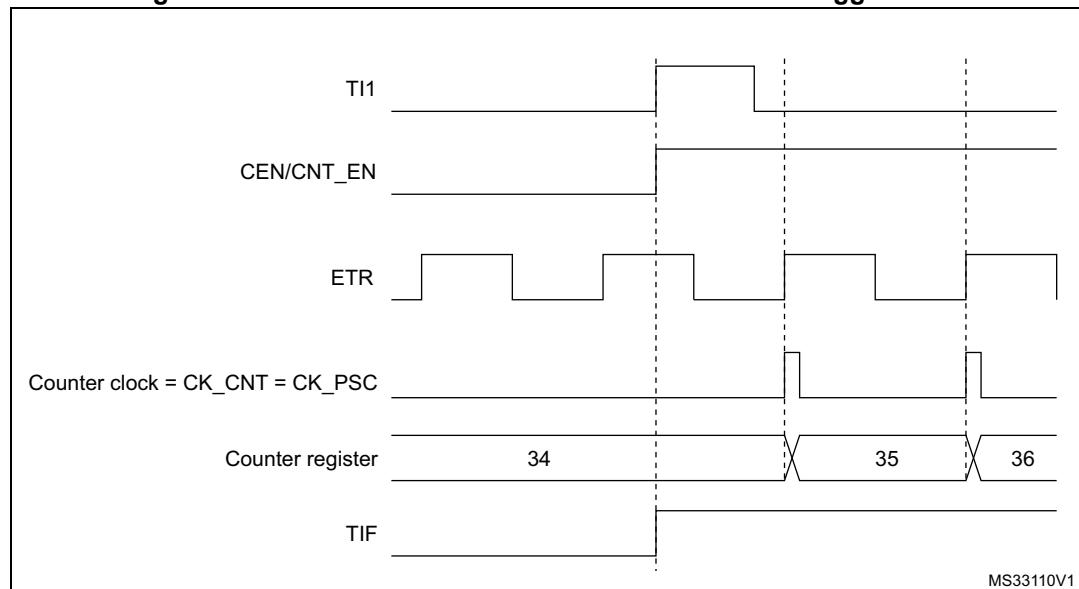
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

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.

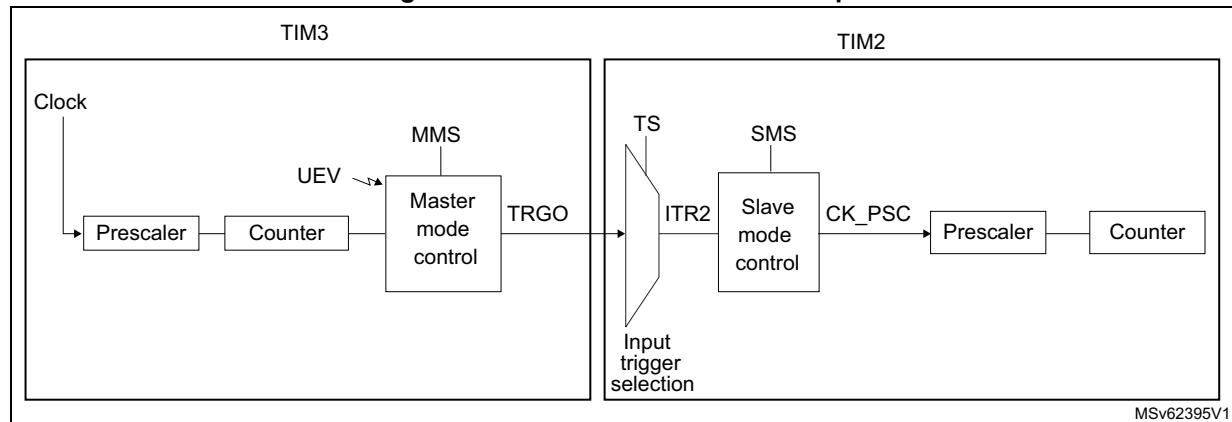
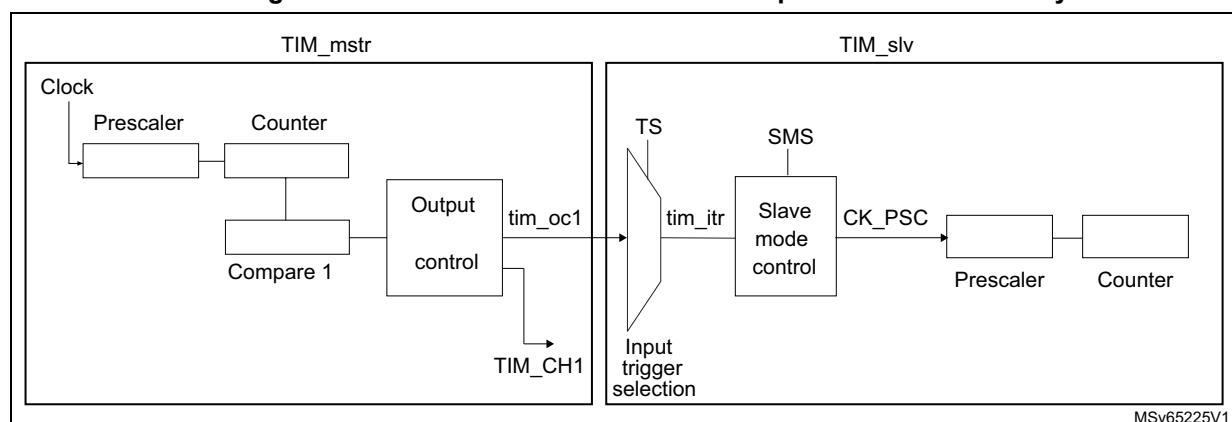
Figure 223. Control circuit in external clock mode 2 + trigger mode



24.3.19 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 224: Master/Slave timer example](#) and [Figure 225: Master/slave connection example with 1 channel only timers](#) present an overview of the trigger selection and the master mode selection blocks.

Figure 224. Master/Slave timer example**Figure 225. Master/slave connection example with 1 channel only timers**

Note: The timers with one channel only (see [Figure 225](#)) 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, TIM3 can be configured to act as a prescaler for TIM2. Refer to [Figure 224](#). To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM3_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM2_SMCR register (writing TS=00010).
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 TIM3 trigger signal (which correspond to the TIM3 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: *If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.*

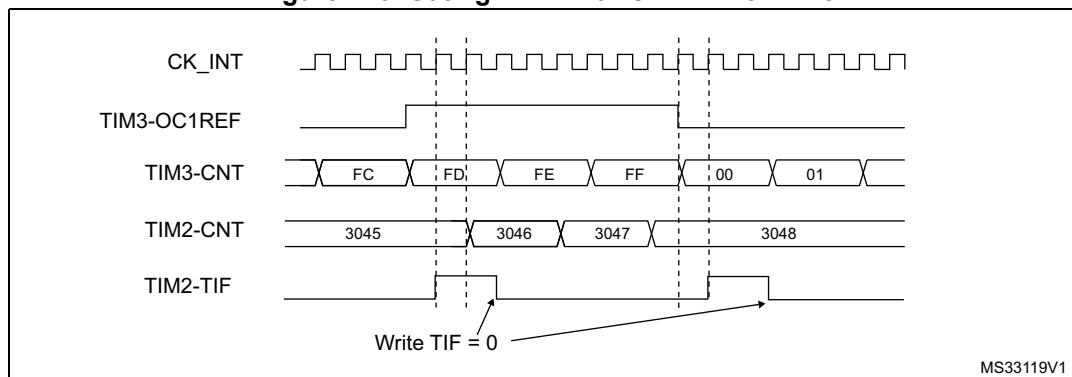
Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to [Figure 224](#) for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 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 TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Enable TIM2 by writing '1 in the CEN bit (TIM2_CR1 register).
6. Start TIM3 by writing '1 in the CEN bit (TIM3_CR1 register).

Note: *The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.*

Figure 226. Gating TIM2 with OC1REF of TIM3

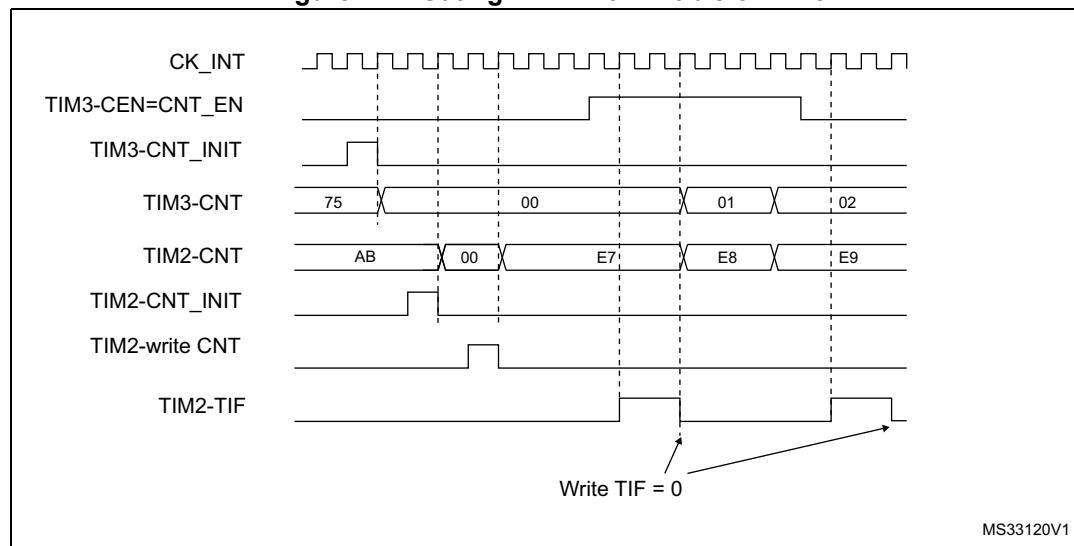


In the example in [Figure 226](#), the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to [Figure 227](#)), we synchronize TIM3 and TIM2. TIM3 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 TIM3 is disabled by writing '0' to the CEN bit in the TIM3_CR1 register:

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM3 by writing '1' in UG bit (TIM3_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_CNT).
8. Enable TIM2 by writing '1' in the CEN bit (TIM2_CR1 register).
9. Start TIM3 by writing '1' in the CEN bit (TIM3_CR1 register).
10. Stop TIM3 by writing '0' in the CEN bit (TIM3_CR1 register).

Figure 227. Gating TIM2 with Enable of TIM3



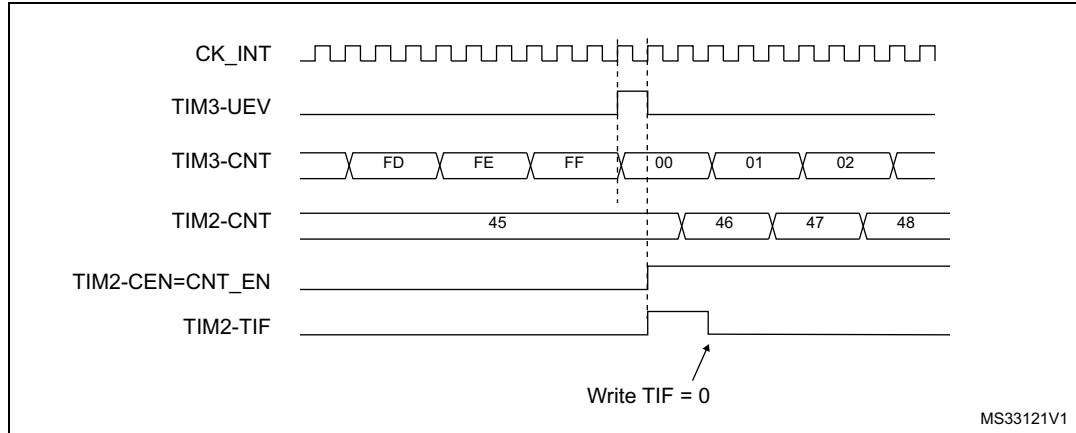
MS33120V1

Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to [Figure 224](#) 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 ($f_{CK_CNT} = f_{CK_INT}/3$).

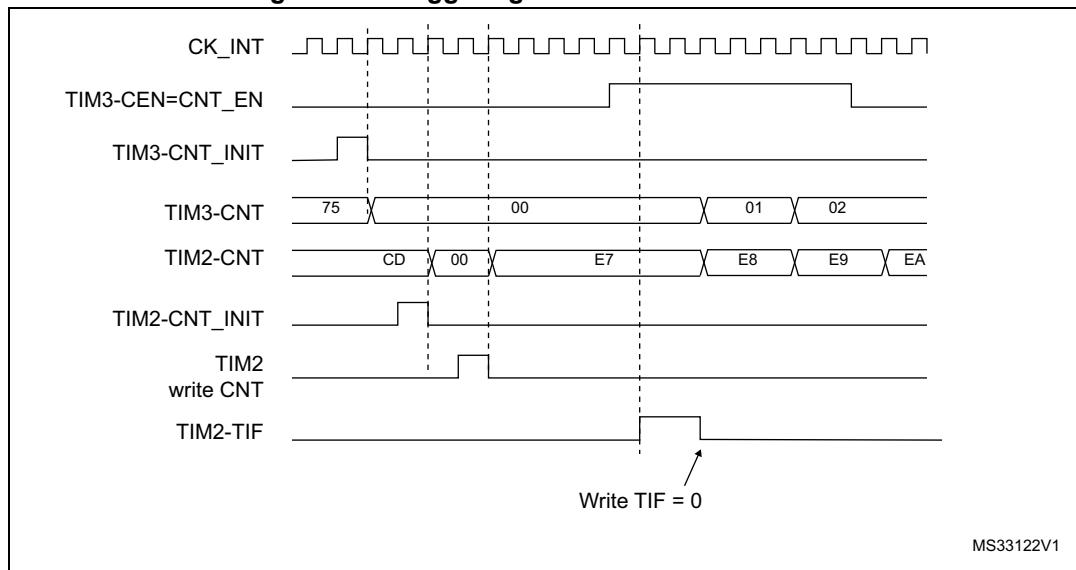
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
2. Configure the TIM3 period (TIM3_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM3 by writing '1 in the CEN bit (TIM3_CR1 register).

Figure 228. Triggering TIM2 with update of TIM3



As in the previous example, both counters can be initialized before starting counting. [Figure 229](#) shows the behavior with the same configuration as in [Figure 228](#) but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 229. Triggering TIM2 with Enable of TIM3



Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to [Figure 224](#) for connections. To ensure the counters are

aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

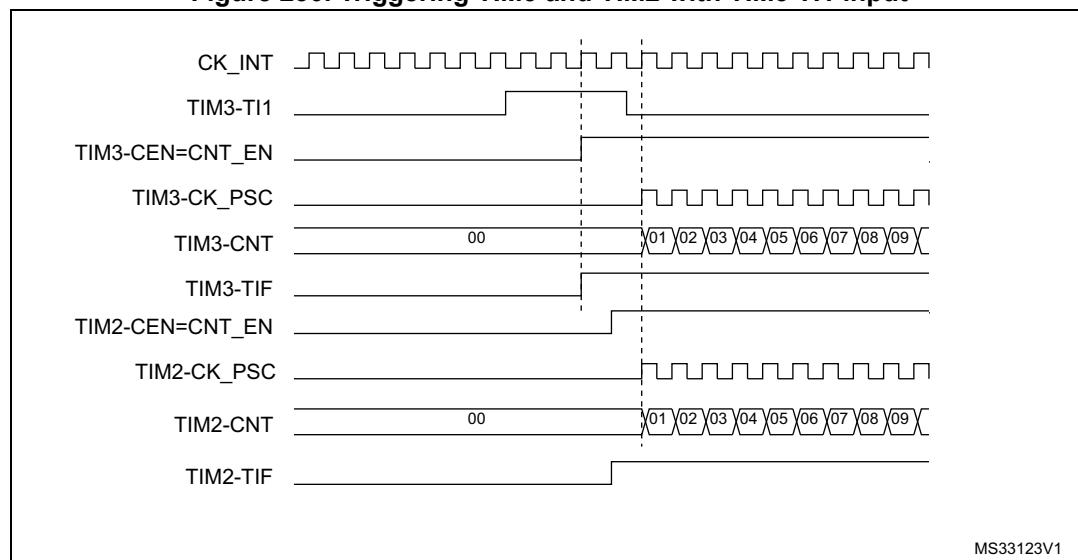
1. Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3_CR2 register).
2. Configure TIM3 slave mode to get the input trigger from TI1 (TS=00100 in the TIM3_SMCR register).
3. Configure TIM3 in trigger mode (SMS=110 in the TIM3_SMCR register).
4. Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3_SMCR register).
5. Configure TIM2 to get the input trigger from TIM3 (TS=00000 in the TIM2_SMCR register).
6. Configure TIM2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note:

In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM3.

Figure 230. Triggering TIM3 and TIM2 with TIM3 TI1 input



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.

24.3.20 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.

24.3.21 Debug mode

When the microcontroller enters debug mode (Cortex®-M0+ 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 : DBGMCU APB1 freeze register \(DBGMCU_APB1FZR\)](#).

24.4 TIM2/TIM3 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).

24.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 3)

Address offset: 0x000

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|--------------|------|----------|----|------|----------|----|-----|-----|-----|------|-----|
| Res. | Res. | Res. | Res. | UIFRE MAP | Res. | CKD[1:0] | | ARPE | CMS[1:0] | | DIR | OPM | URS | UDIS | CEN |
| | | | | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:12 Reserved, 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.

24.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 3)

Address offset: 0x04

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|------|------|------|----------|---|---|------|------|------|------|
| Res. | TI1S | MMS[2:0] | | | CCDS | Res. | Res. | Res. |

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 23.3.25: Interfacing with Hall sensors on page 602](#)

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.

24.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 3)

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 133: TIMx internal trigger connection on page 702](#) 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 COMP1 or COMP2 output depending on
TIMx_OR1.OCREF_CLR[1:0]

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 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 133. TIMx internal trigger connection

| Slave TIM | ITR0 | ITR1 | ITR2 | ITR3 |
|-----------|------|-------|-------|------|
| TIM2 | TIM1 | TIM15 | TIM3 | - |
| TIM3 | TIM1 | TIM2 | TIM15 | - |

24.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 3)

Address offset: 0x0C

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-----|------|-------|-------|-------|-------|-----|------|-----|------|-------|-------|-------|-------|-----|
| Res. | TDE | Res. | CC4DE | CC3DE | CC2DE | CC1DE | UDE | Res. | TIE | Res. | CC4IE | CC3IE | CC2IE | CC1IE | UIE |
| | rw | | rw | rw | rw | rw | rw | | rw | | rw | rw | rw | rw | rw |

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 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.

24.4.5 TIMx status register (TIMx_SR)(x = 2 to 3)

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 | 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.

24.4.6 TIMx event generation register (TIMx_EGR)(x = 2 to 3)

Address offset: 0x14

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|------|------|------|----|------|------|------|------|------|----|
| Res. | TG | Res. | CC4G | CC3G | CC2G | CC1G | UG |

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).

24.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 3)

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).

24.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 3)

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).

24.4.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 3)

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 filterBits 3:2 **IC3PSC[1:0]**: Input capture 3 prescalerBits 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).

24.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) ($x = 2$ to 3)

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).

24.4.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 3)

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 |

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), TIxFP1 is 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 134. 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.

24.4.12 TIMx counter (TIMx_CNT)(x = 2 to 3)

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 31:16 **CNT[31:16]**: Most significant part counter value (TIM2)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

24.4.13 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3)

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 | | | | | | | | | | | | | | | |
| 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[30:16] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 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 30:16 **CNT[30:16]**: Most significant part counter value (TIM2)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

24.4.14 TIMx prescaler (TIMx_PSC)(x = 2 to 3)

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”).

24.4.15 TIMx auto-reload register (TIMx_ARR)(x = 2 to 3)

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 31:16 **ARR[31:16]**: High auto-reload value (TIM2)

Bits 15:0 **ARR[15:0]**: Low Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 24.3.1: Time-base unit on page 653](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

24.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 3)

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 31:16 **CCR1[31:16]**: High Capture/Compare 1 value (TIM2)

Bits 15:0 **CCR1[15:0]**: Low 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). The TIMx_CCR1 register is read-only and cannot be programmed.

24.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 3)

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 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (TIM2)

Bits 15:0 **CCR2[15:0]**: Low 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.

24.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 3)

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 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (TIM2)

Bits 15:0 **CCR3[15:0]**: Low 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.

24.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 3)

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 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (TIM2)

Bits 15:0 **CCR4[15:0]**: Low Capture/Compare value

- 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.

- 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.

24.4.20 TIMx DMA control register (TIMx_DCR)(x = 2 to 3)

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.

24.4.21 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 3)

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) × 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).

24.4.22 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | OCREF_CLR [1:0] |
| | | | | | | | | | | | | | | | rw rw |

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

Bits 1:0 **OCREF_CLR[1:0]**: Ocref_clr source selection

This bit selects the ocref_clr input source.

00: COMP1 output is connected to the OCREF_CLR input

01: COMP2 output is connected to the OCREF_CLR input

Others: Reserved

24.4.23 TIM3 option register 1 (TIM3_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | OCREF_CLR [1:0] |
| | | | | | | | | | | | | | | | rw rw |

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

Bits 1:0 **OCREF_CLR[1:0]**: Ocref_clr source selection

This bit selects the ocref_clr input source.

Bits 1:0 OCREF_CLR[1:0]

00: COMP1 output is connected to the OCREF_CLR input

01: COMP2 output is connected to the OCREF_CLR input

Others: Reserved

24.4.24 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. | Res. | ETRSEL[3:2] |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRSEL[1:0] | 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: ETR legacy mode

0001: COMP1

0010: COMP2

0011: LSE

0100: MCO

0101: MCO2

Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

24.4.25 TIM3 alternate function option register 1 (TIM3_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. |
| 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

0001: COMP1 output

0010: COMP2 output

Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

24.4.26 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. | TI3SEL[3:0] | |
| | | | | | | | | | | | | | | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | TI2SEL[3:0] | Res. | TI1SEL[3:0] | |
| | | | | | rw | rw | rw | rw | | | | | | rw | rw |

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

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection

These bits select the TI3[0] to TI3[15] input source.

0000: TIM2_CH3 input

Others: Reserved

Bits 15:12 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

0001: COMP2 output

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

0001: COMP1 output

Others: Reserved

24.4.27 TIM3 timer input selection register (TIM3_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. | TI3SEL[3:0] | | | |
| | | | | | | | | | | | | 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 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection

These bits select the TI3[0] to TI3[15] input source.

0000: TIM3_CH3 input

Others: Reserved

Bits 15:12 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: TIM3_CH2 input

0001: COMP2 output

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: TIM3_CH1 input

0001: COMP1 output

Others: Reserved

24.4.28 TIMx register map

TIMx registers are mapped as described in the table below:

Table 135. TIM2/TIM3 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 | |
|--------|-----------------------------------|------|------|---------|------|------|---------|------|------|-----------|-------|------|-----------|-------|------|--------|-------|------|--------|------|-------|------|-------|------|-------|------|-------|
| | TIMx_CR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| 0x00 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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. | Res. | Res. |
| 0x04 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_SMCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| 0x08 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_DIER | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| 0x0C | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_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. |
| 0x10 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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. | Res. |
| 0x14 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_CCMR1 Output Compare mode | Res. | 0 | OC2M[3] | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 0 | SMS[3] | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| 0x18 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_CCMR1 Input Capture mode | Res. | 0 | OC4M[3] | Res. | 0 | OC1M[3] | Res. | 0 | OC2CE | Res. | 0 | OC2M | [2:0] | Res. | Res. | Res. | 0 | ECP | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| 0x1C | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_CCMR2 Output Compare mode | Res. | 0 | OC3M[3] | Res. | 0 | OC4CE | 0 | 0 | IC4F[3:0] | 0 | 0 | IC2F[3:0] | 0 | 0 | IC2PSC | [1:0] | 0 | CC4OF | 0 | CC4DE | 0 | CC3DE | 0 | CC2DE | 0 | CC1DE |
| 0x20 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_CCER | Res. | 0 | CC4NP | 0 | 0 | OC4P | 0 | 0 | OC4M | [2:0] | 0 | IC4PSC | [1:0] | 0 | OC4PE | 0 | 0 | CC3NP | 0 | CC3OF | 0 | CC3DE | 0 | CC3F | 0 | CC3S |
| 0x24 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_CCER | Res. | 0 | CC4E | 0 | 0 | CC4NP | 0 | 0 | OC4E | 0 | 0 | IC4PSC | [1:0] | 0 | CC4FE | 0 | 0 | CC4NP | 0 | CC4FE | 0 | CC4DE | 0 | CC4F | 0 | CC4IE |
| 0x28 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_CCER | Res. | 0 | CC3NP | 0 | 0 | CC3P | 0 | 0 | CC3E | 0 | 0 | IC3PSC | [1:0] | 0 | CC4S | [1:0] | 0 | CC3NP | 0 | CC3F | 0 | CC3DE | 0 | CC3F | 0 | CC3S |
| 0x2C | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_CCER | Res. | 0 | CC2NP | 0 | 0 | CC2P | 0 | 0 | CC2E | 0 | 0 | IC2PSC | [1:0] | 0 | CC3PE | 0 | 0 | CC1NP | 0 | CC1FE | 0 | CC1F | 0 | CC1IE | 0 | CC1S |
| 0x30 | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TIMx_CCER | Res. | 0 | CC1NP | 0 | 0 | CC1P | 0 | 0 | CC1E | 0 | 0 | IC1PSC | [1:0] | 0 | CC1F | 0 | 0 | CC1NP | 0 | CC1FE | 0 | CC1F | 0 | CC1IE | 0 | CC1S |

Table 135. TIM2/TIM3 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[30:16] (TIM2 only, reserved on the other timers) | | | | | | | | | | | | | | | CNT[15:0] | | | | | | | | | | | | | | | |
| | | | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| 0x28 | TIMx_PSC | Res. | PSC[15:0] | | | | | | | | | | | | | | | PSC[15:0] | | | | | | | | | | | | | | | |
| | | | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | |
| 0x2C | TIMx_ARR | ARR[31:16] (TIM2 only, reserved on the other timers) | | | | | | | | | | | | | | | ARR[15:0] | | | | | | | | | | | | | | | | |
| | | 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 | | | | | |
| 0x30 | | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x34 | TIMx_CCR1 | CCR1[31:16] (TIM2 only, reserved on the other timers) | | | | | | | | | | | | | | | CCR1[15:0] | | | | | | | | | | | | | | | | |
| | | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | |
| 0x38 | TIMx_CCR2 | CCR2[31:16] (TIM2 only, reserved on the other timers) | | | | | | | | | | | | | | | CCR2[15:0] | | | | | | | | | | | | | | | | |
| | | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | |
| 0x3C | TIMx_CCR3 | CCR3[31:16] (TIM2 only, reserved on the other timers) | | | | | | | | | | | | | | | CCR3[15:0] | | | | | | | | | | | | | | | | |
| | | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | |
| 0x40 | TIMx_CCR4 | CCR4[31:16] (TIM2 only, reserved on the other timers) | | | | | | | | | | | | | | | CCR4[15:0] | | | | | | | | | | | | | | | | |
| | | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | |
| 0x44 | | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x48 | TIMx_DCR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DBL[4:0] | Res. | DBA[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 | 0 | 0 | 0 | | | |
| 0x4C | TIMx_DMAR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DMAB[15:0] | Res. | Res. | | | | |
| | | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| 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. | OCREF_CLR | 0 | 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 | | | |

Table 135. TIM2/TIM3 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 |
|--------|-------------------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|---|
| 0x50 | TIM3_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. | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x60 | TIM2_AF1 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x60 | TIM3_AF1 | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x68 | TIM2_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. | Res. | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x68 | TIM3_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. | Res. | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

25 Basic timers (TIM6/TIM7)

25.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

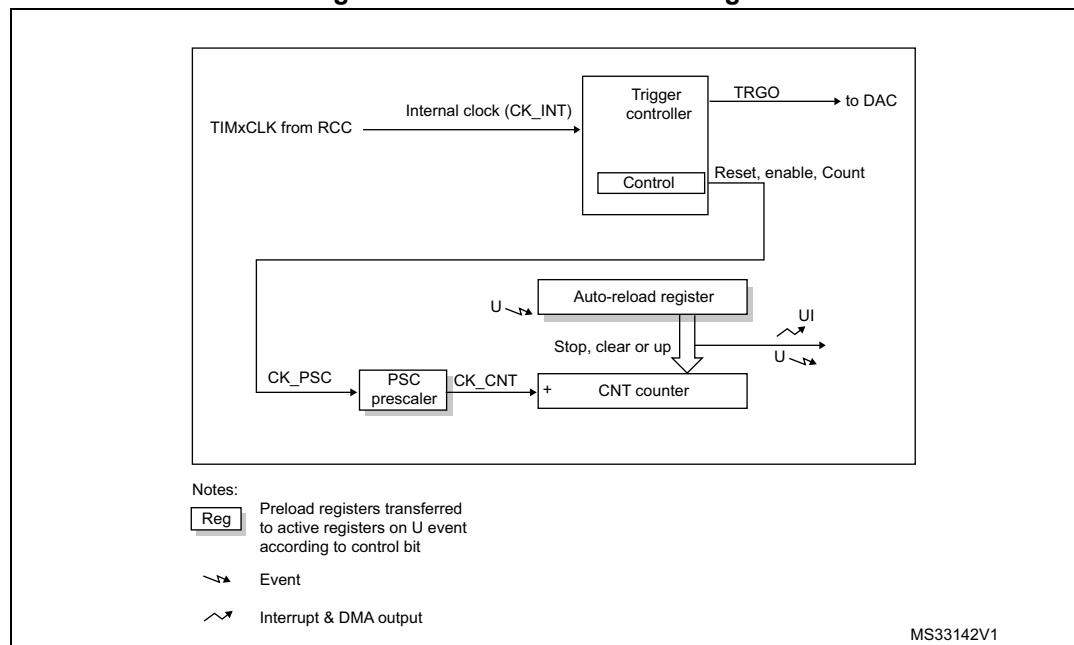
The timers are completely independent, and do not share any resources.

25.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:

- 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
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 231. Basic timer block diagram



25.3 TIM6/TIM7 functional description

25.3.1 Time-base unit

The main block of the programmable 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)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents 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 the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value 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.

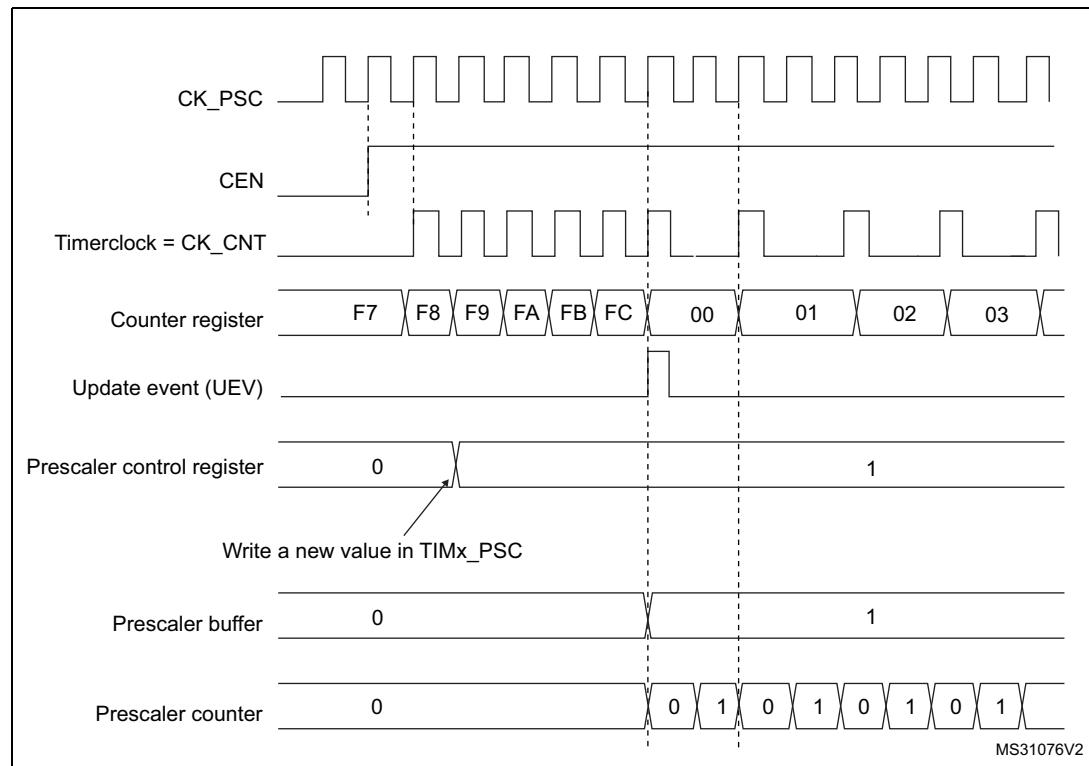
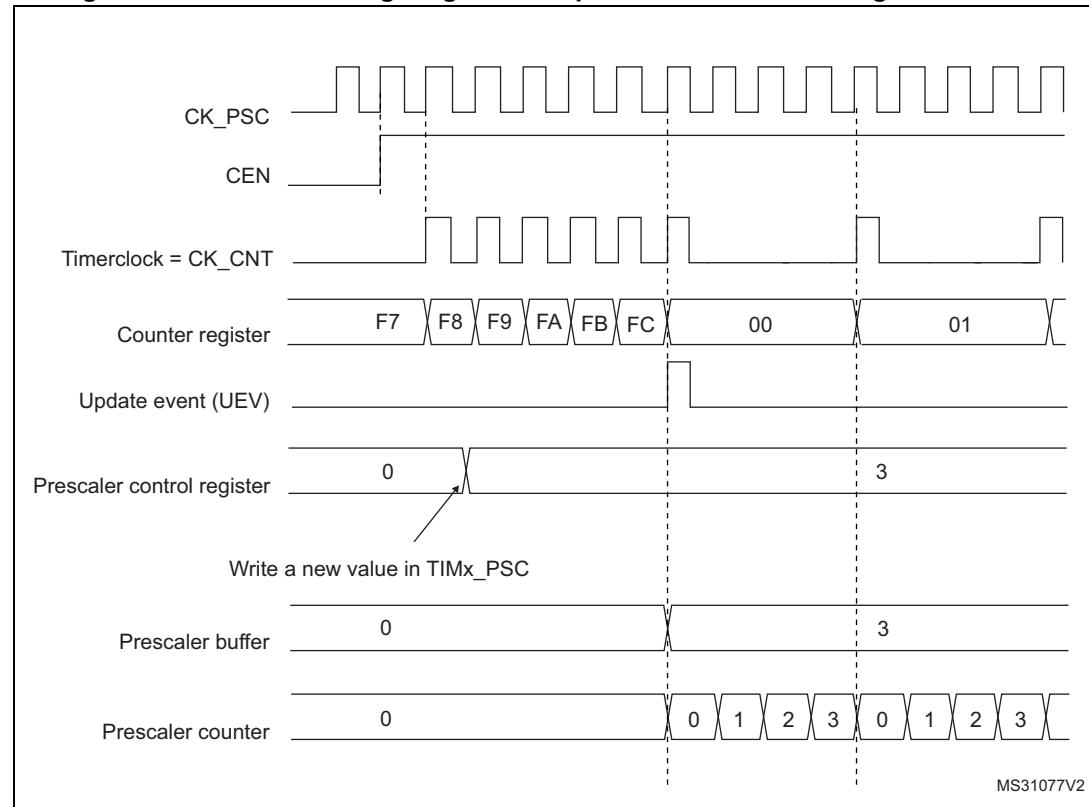
The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

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 register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 232 and *Figure 233* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 232. Counter timing diagram with prescaler division change from 1 to 2**Figure 233. Counter timing diagram with prescaler division change from 1 to 4**

25.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents 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).

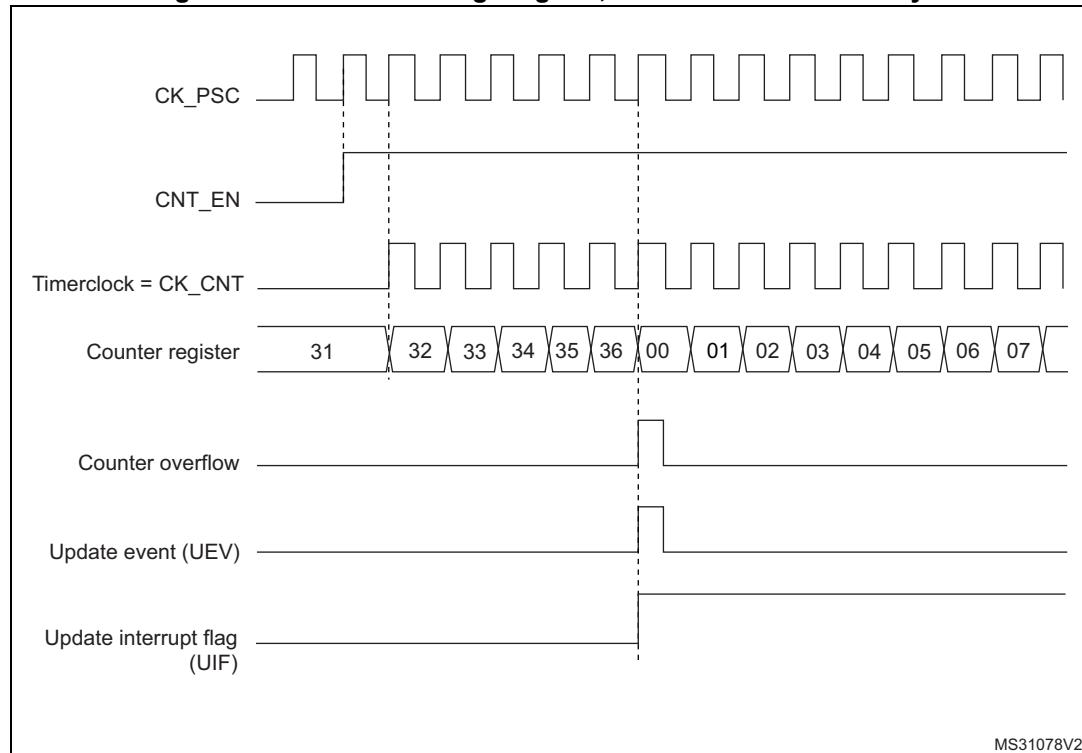
The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been cleared, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents 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 234. Counter timing diagram, internal clock divided by 1



MS31078V2

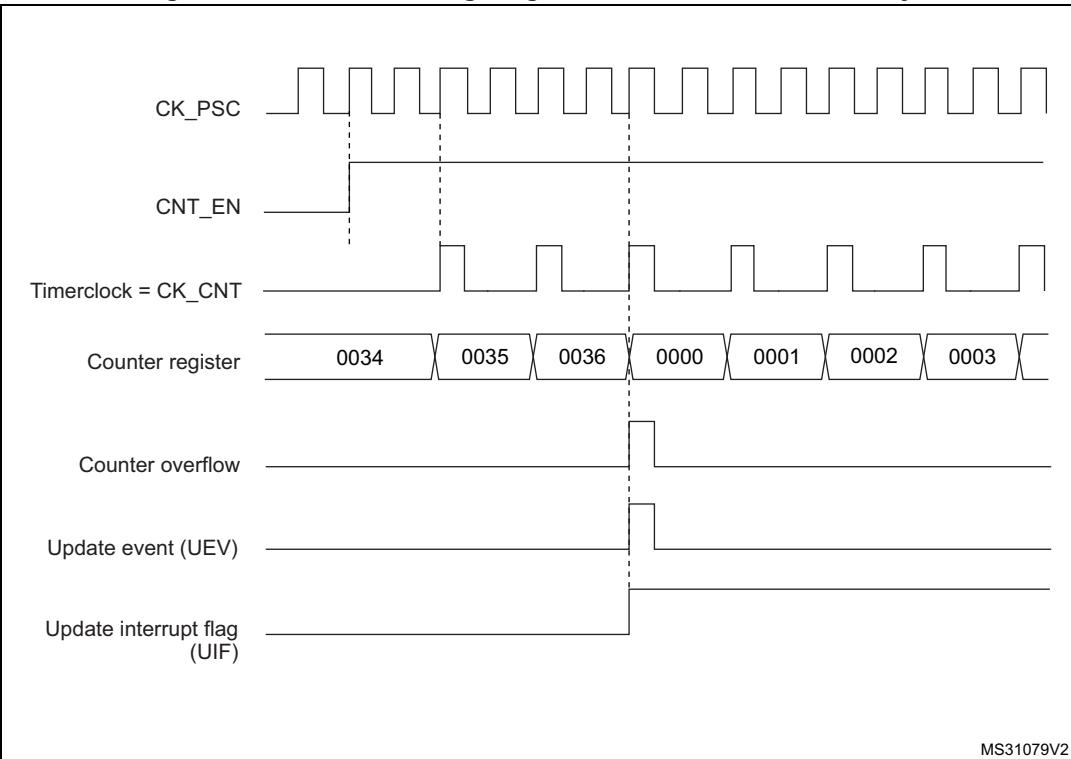
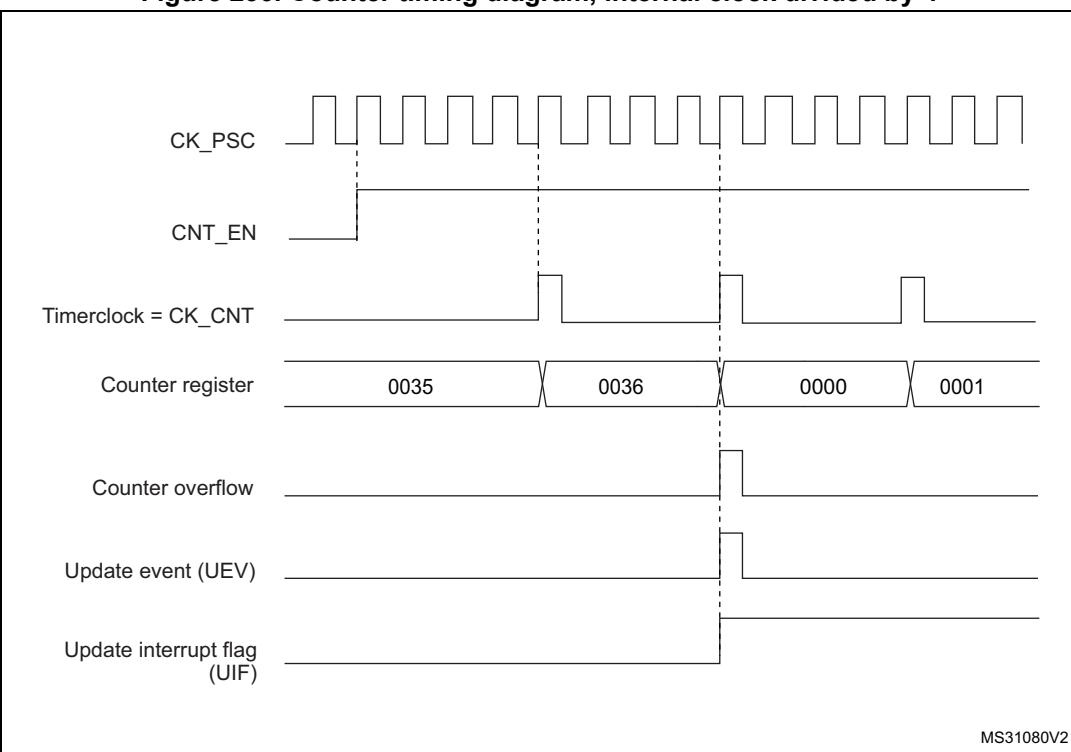
Figure 235. Counter timing diagram, internal clock divided by 2**Figure 236. Counter timing diagram, internal clock divided by 4**

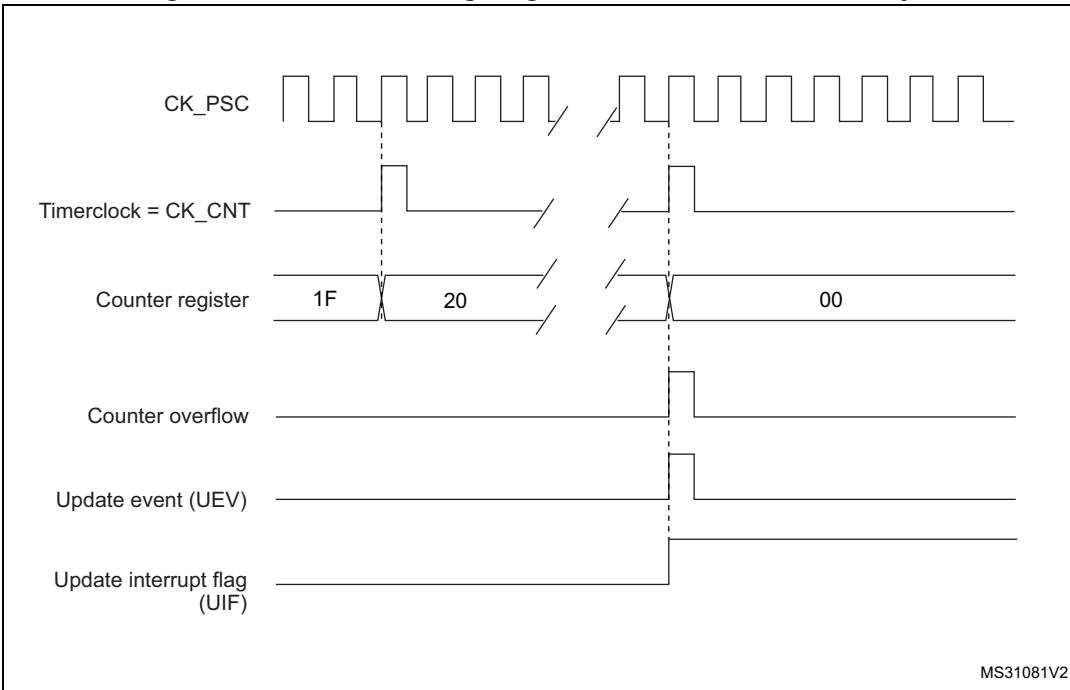
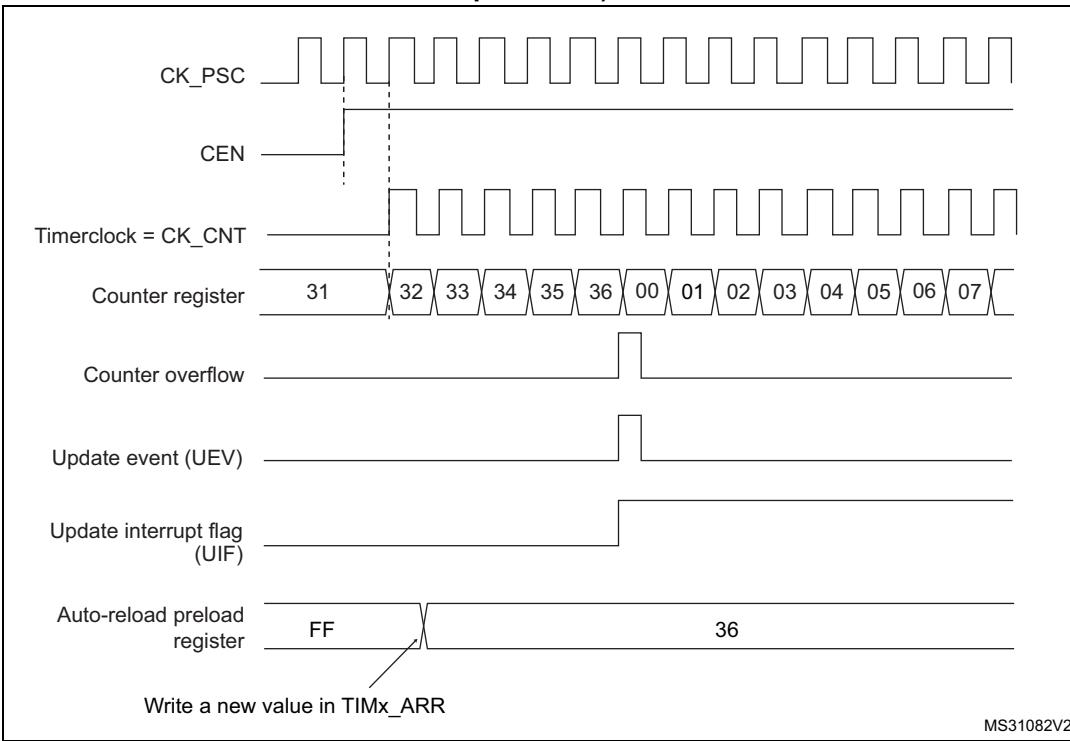
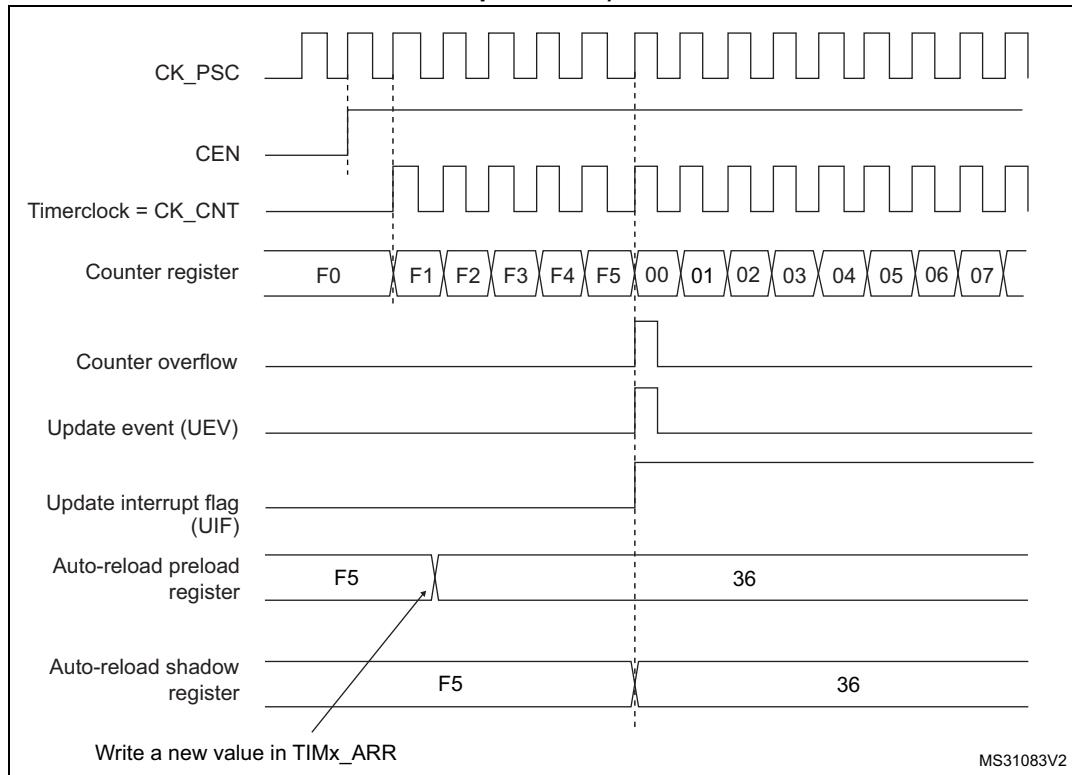
Figure 237. Counter timing diagram, internal clock divided by N**Figure 238. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)**

Figure 239. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



25.3.3 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 to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

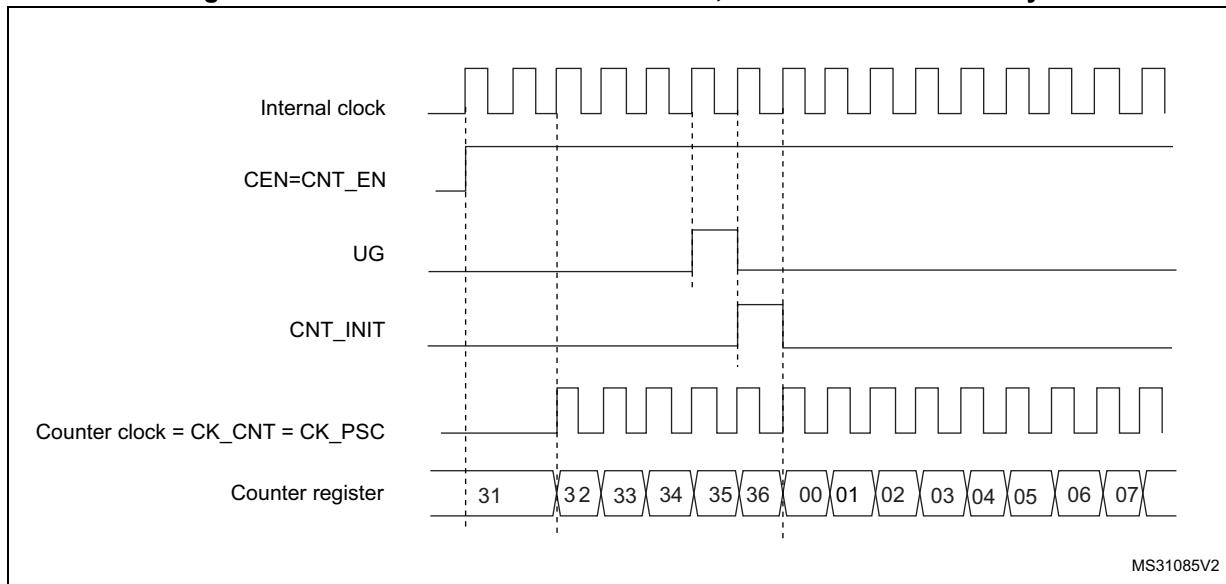
There is no latency between the assertions of the UIF and UIFCPY flags.

25.3.4 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

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 for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 240 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 240. Control circuit in normal mode, internal clock divided by 1

25.3.5 Debug mode

When the microcontroller enters the debug mode (Cortex®-M0+ core - halted), the TIMx counter either continues to work normally or stops, depending on the `DBG_TIMx_STOP` configuration bit in the DBG module. For more details, refer to [Section 37.9: Microcontroller debug unit \(DBGMCU\)](#).

25.4 TIM6/TIM7 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 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x000

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|-----------|------|------|------|------|------|------|------|-----|-----|------|-----|
| Res. | Res. | Res. | Res. | UIFRE MAP | Res. | Res. | Res. | ARPE | Res. | Res. | Res. | OPM | URS | UDIS | CEN |
| | | | | 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.

Bits 10:8 Reserved, must be kept at reset 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 CEN bit).

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 generates 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 controllerBuffered registers are then loaded with their preload values.

- 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). 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: 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.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

25.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04

Reset value: 0x0000

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

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used 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 a trigger output (TRGO). If 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 a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

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.

Bits 3:0 Reserved, must be kept at reset value.

25.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

Address offset: 0x0C

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|------|-----|------|------|------|------|------|------|------|-----|
| Res. | UDE | Res. | UIE |
| | | | | | | | rw | | | | | | | | rw |

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

- 0: Update DMA request disabled.
- 1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled.
- 1: Update interrupt enabled.

25.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10

Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

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 and if UDIS = 0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

25.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|----|
| Res. | UG |
| | | | | | | | | | | | | | | | w |

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

25.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

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. |
| r | | | | | | | | | | | | | | | |
| 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

25.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

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 into 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”).

25.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)

Address offset: 0x2C

Reset value: 0xFFFF

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| ARR[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded into the actual auto-reload register.

Refer to [Section 25.3.1: Time-base unit on page 727](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

25.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 136. TIMx 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 | TIMx_CR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| | | | 0x00 | |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | | | 0x04 | |
| 0x08 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | | | 0x0C |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | | | 0x10 |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | | | 0x14 |
| 0x18-0x20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x24 | TIMx_CNT | UFCOPY or Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
| 0x28 | TIMx_PSC | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | | | 0x28 |
| 0x2C | TIMx_ARR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| | | | 0x2C |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

26 General-purpose timers (TIM15/TIM16)

26.1 TIM15/TIM16 introduction

The TIM15/TIM16 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 TIM15/TIM16 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in [Section 26.4.23: Timer synchronization \(TIM15\)](#).

26.2 TIM15 main features

TIM15 includes the following features:

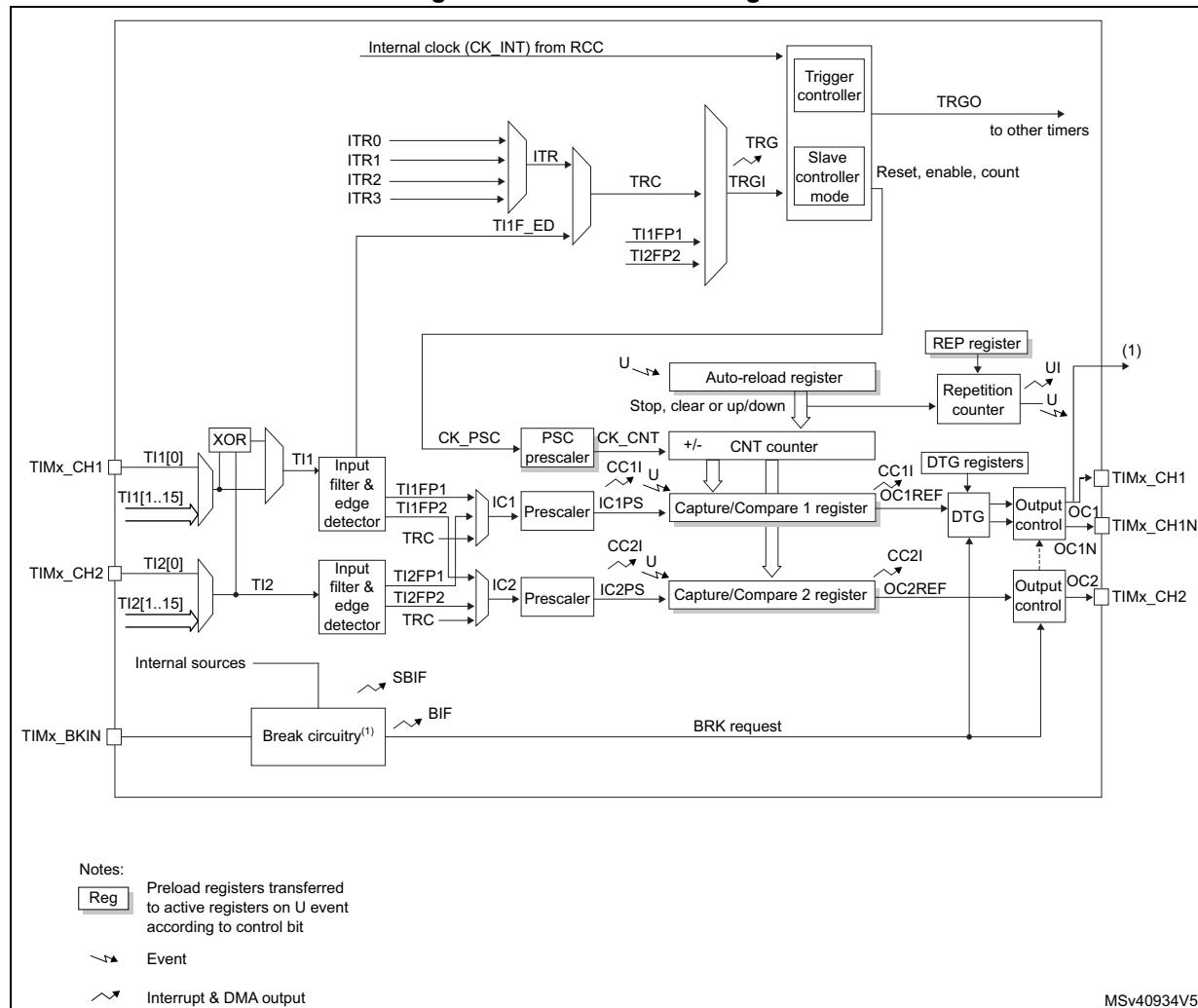
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
 - Input capture
 - Output compare
 - PWM generation (edge mode)
 - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
 - Update: counter overflow, counter initialization (by software or internal/external trigger)
 - Trigger event (counter start, stop, initialization or count by internal/external trigger)
 - Input capture
 - Output compare
 - Break input (interrupt request)

26.3 TIM16 main features

The TIM16 timer 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

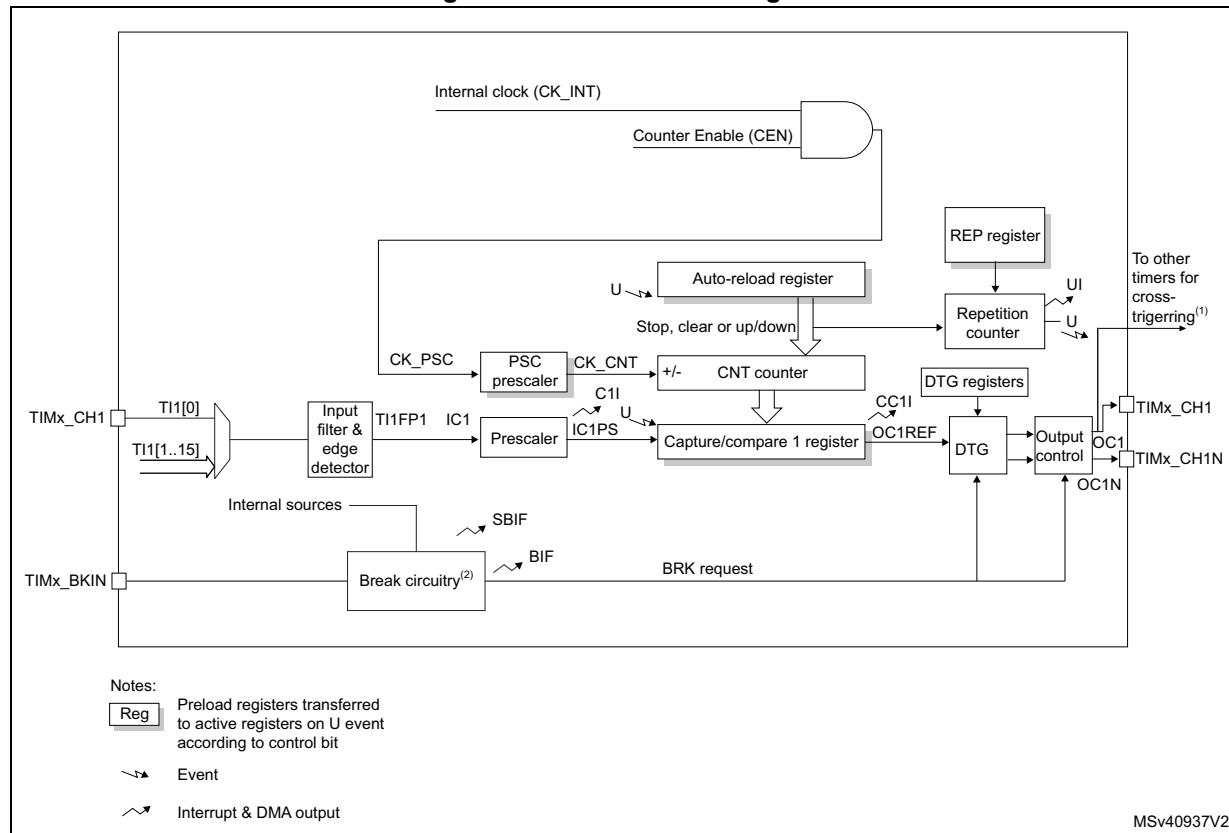
Figure 241. TIM15 block diagram



1. The internal break event source can be:

- A clock failure event generated by CSS. For further information on the CSS, refer to [Section 5.2.10: Clock security system \(CSS\)](#)
- A PVD output
- SRAM parity error signal
- Cortex®-M0+ LOCKUP (Hardfault) output
- COMP output

Figure 242. TIM16 block diagram



1. This signal can be used as trigger for some slave timer, see [Section 26.4.24: Using timer output as trigger for other timers \(TIM16\)](#).
2. The internal break event source can be:
 - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 5.2.10: Clock security system \(CSS\)](#)
 - A PVD output
 - SRAM parity error signal
 - Cortex®-M0+ LOCKUP (Hardfault) output
 - COMP output

26.4 TIM15/TIM16 functional description

26.4.1 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.

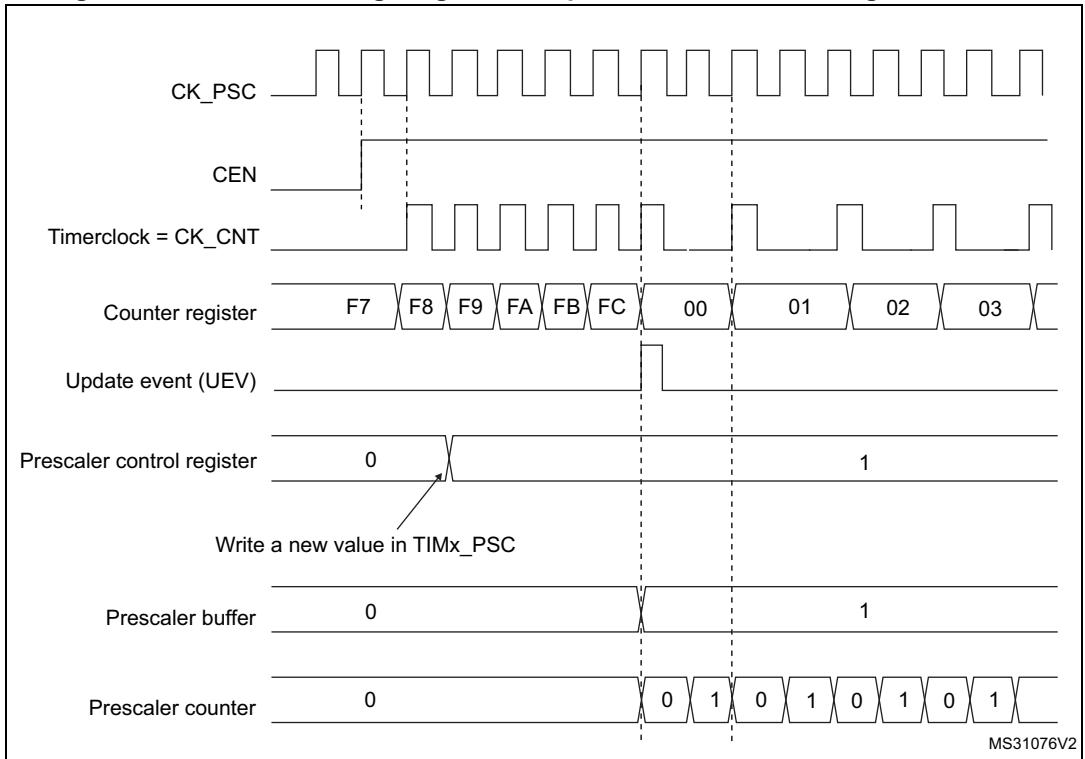
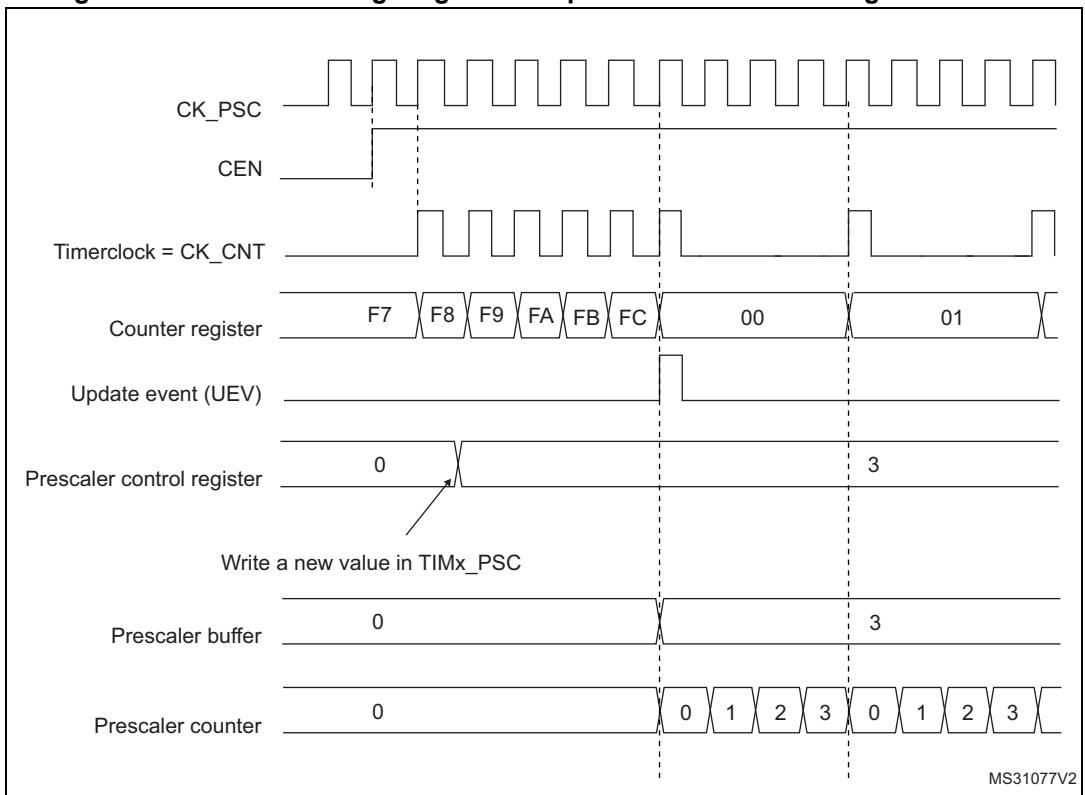
The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

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 243 and *Figure 244* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 243. Counter timing diagram with prescaler division change from 1 to 2**Figure 244. Counter timing diagram with prescaler division change from 1 to 4**

26.4.2 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.

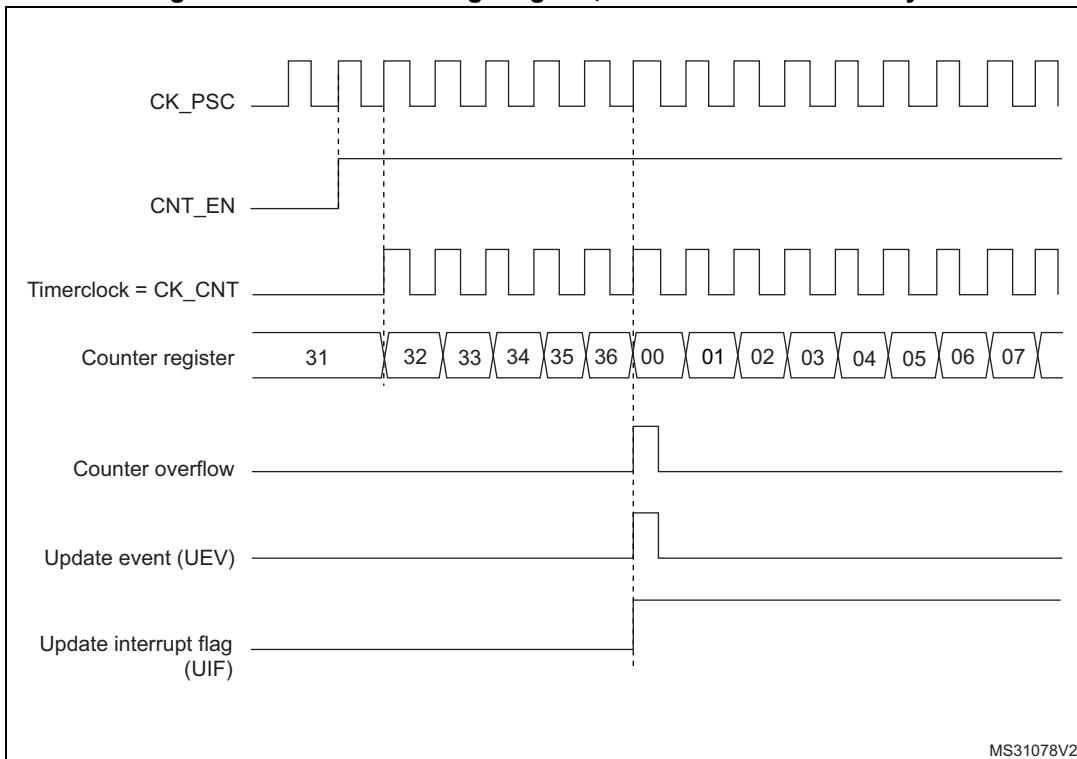
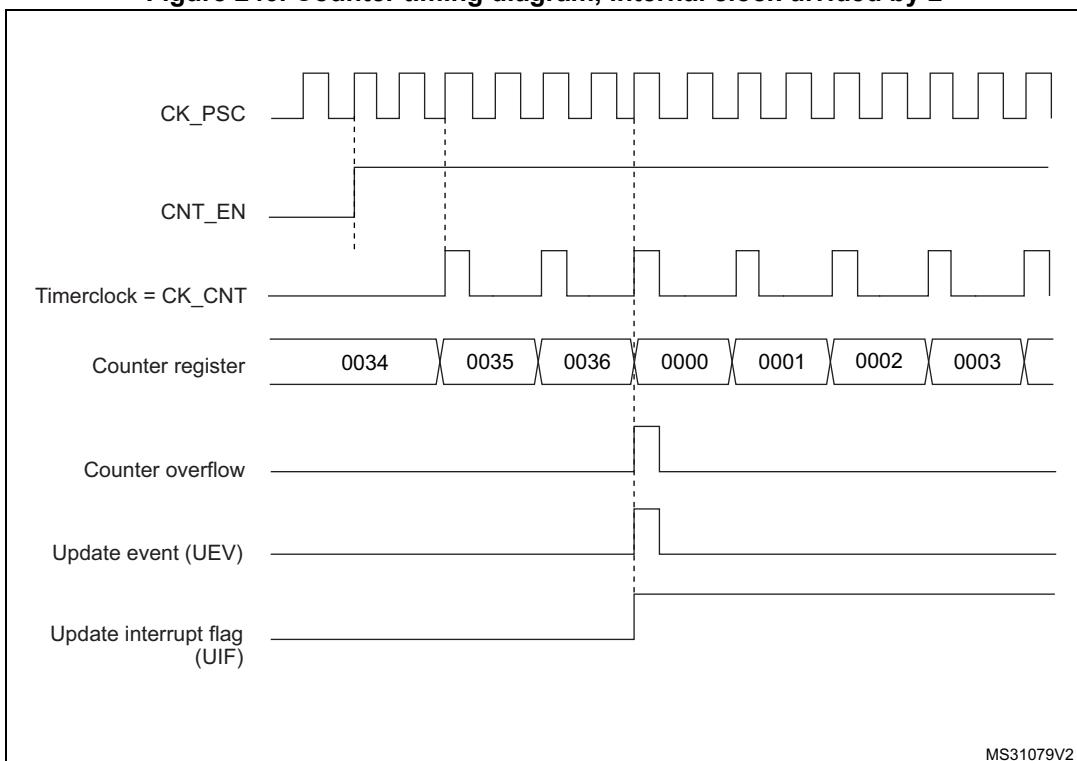
Figure 245. Counter timing diagram, internal clock divided by 1**Figure 246. Counter timing diagram, internal clock divided by 2**

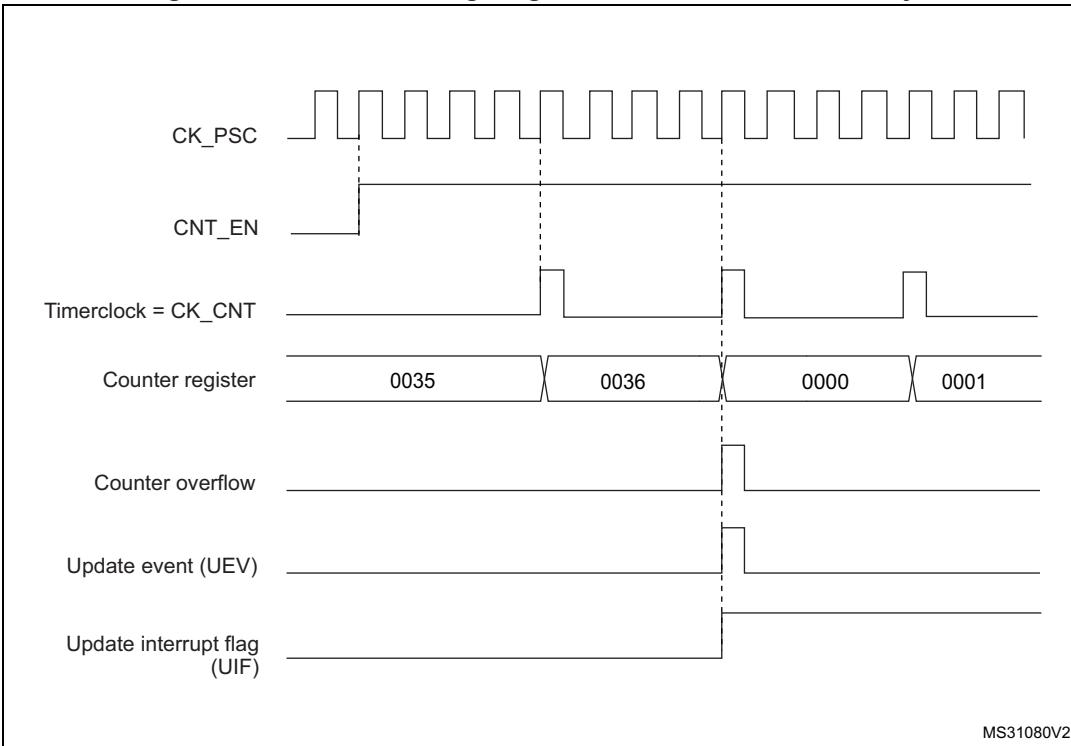
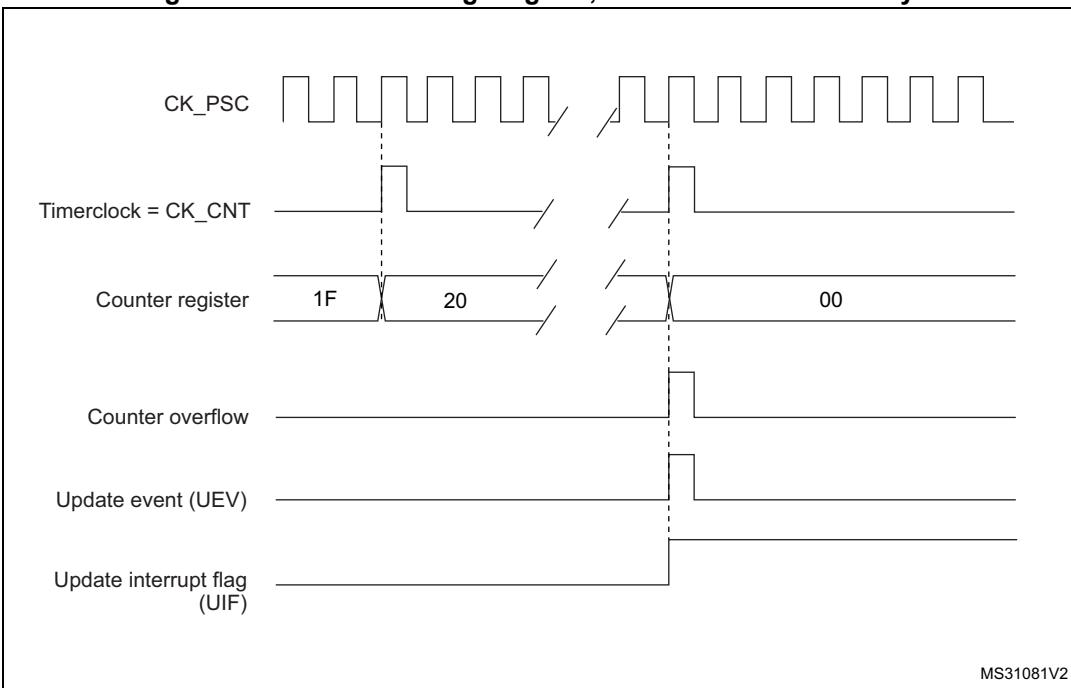
Figure 247. Counter timing diagram, internal clock divided by 4**Figure 248. Counter timing diagram, internal clock divided by N**

Figure 249. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

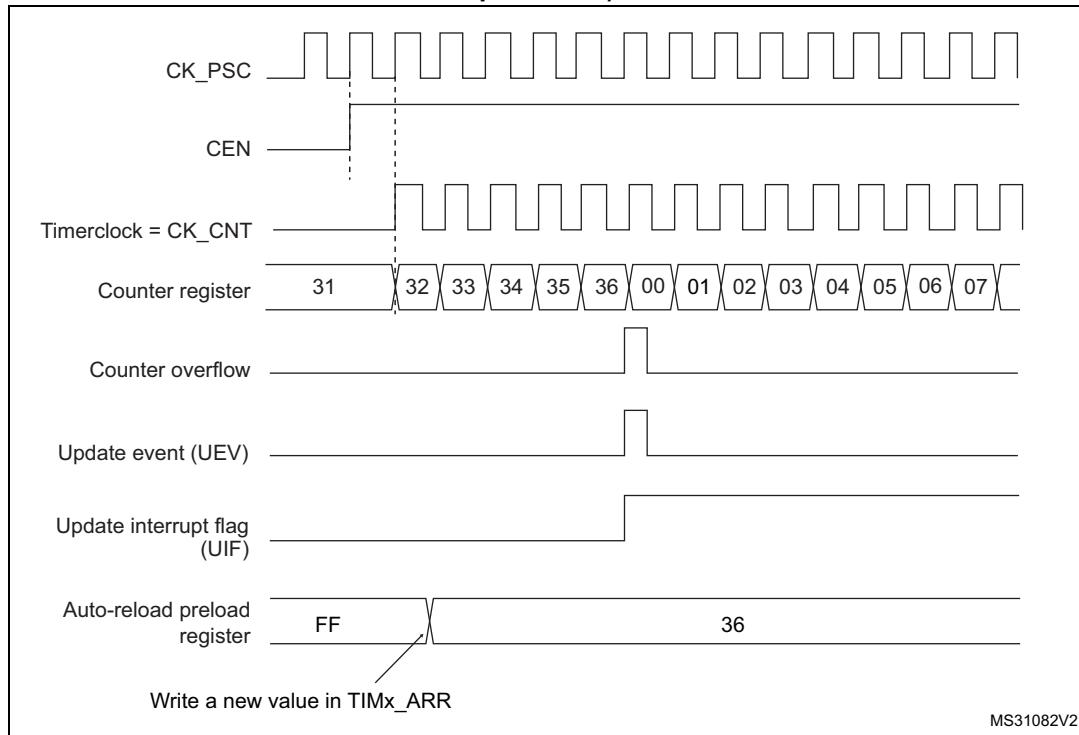
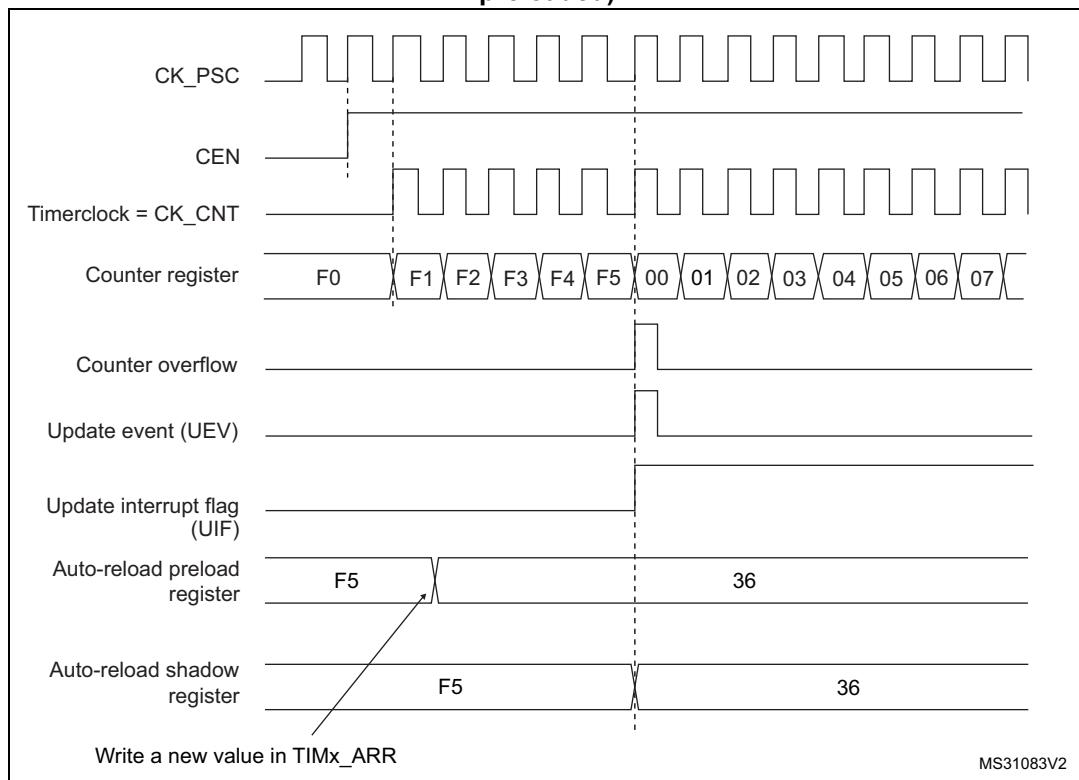


Figure 250. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



26.4.3 Repetition counter

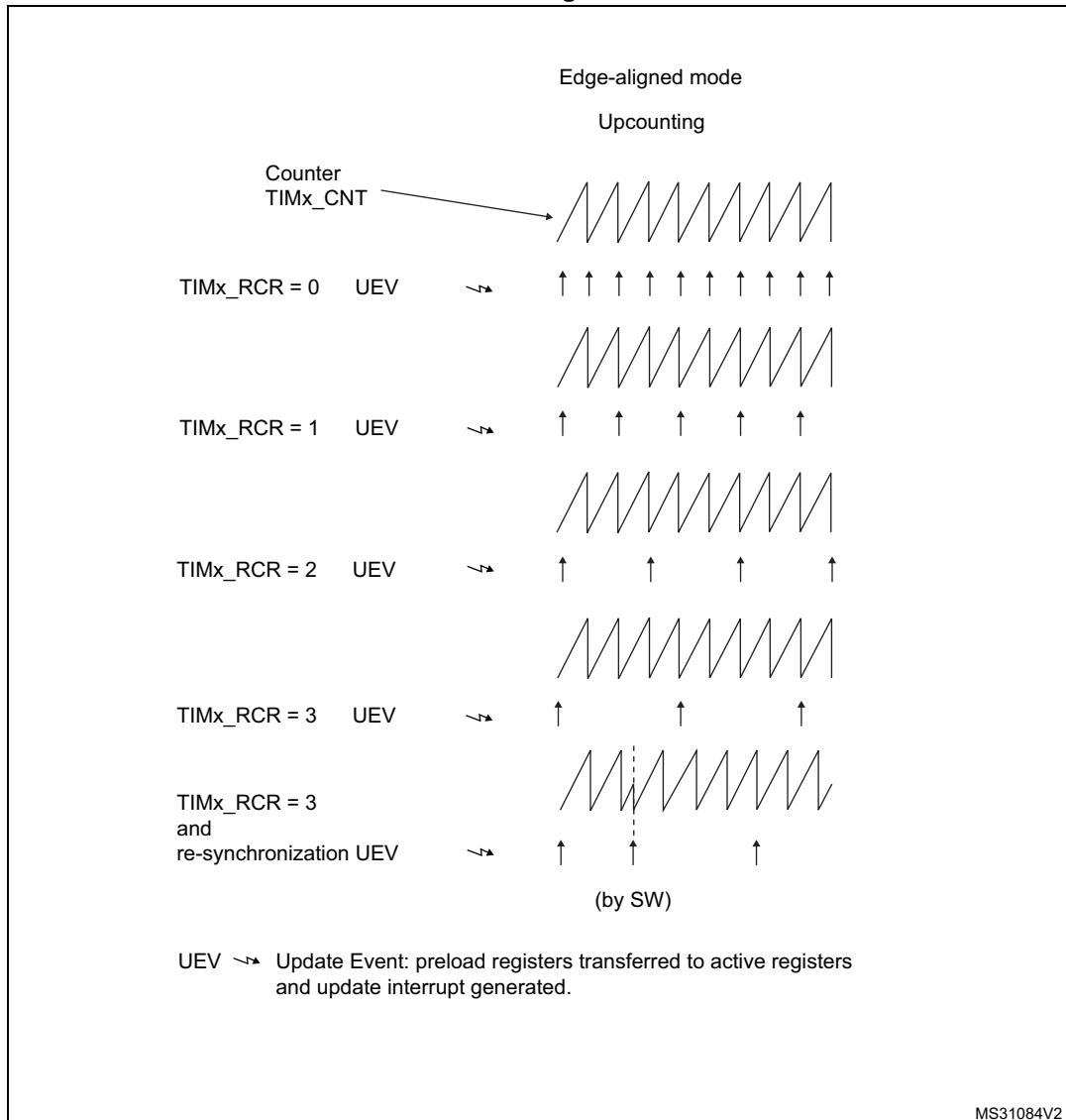
Section 26.4.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

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 251](#)). 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 251. Update rate examples depending on mode and TIMx_RCR register settings



MS31084V2

26.4.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to [Using one timer as prescaler for another timer on page 690](#) for more details.

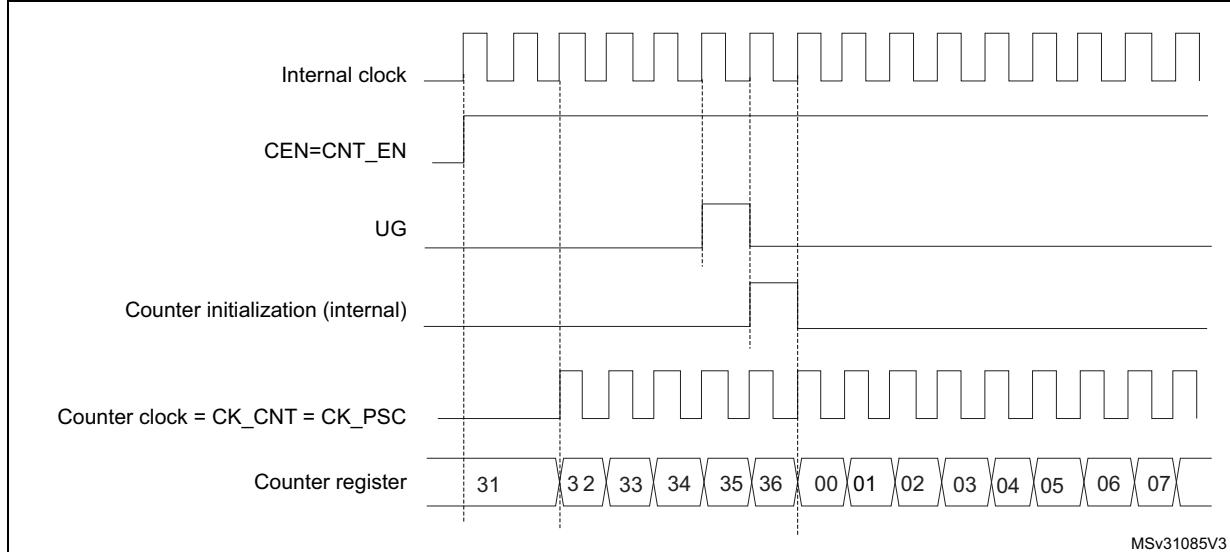
Internal clock source (CK_INT)

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 252 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

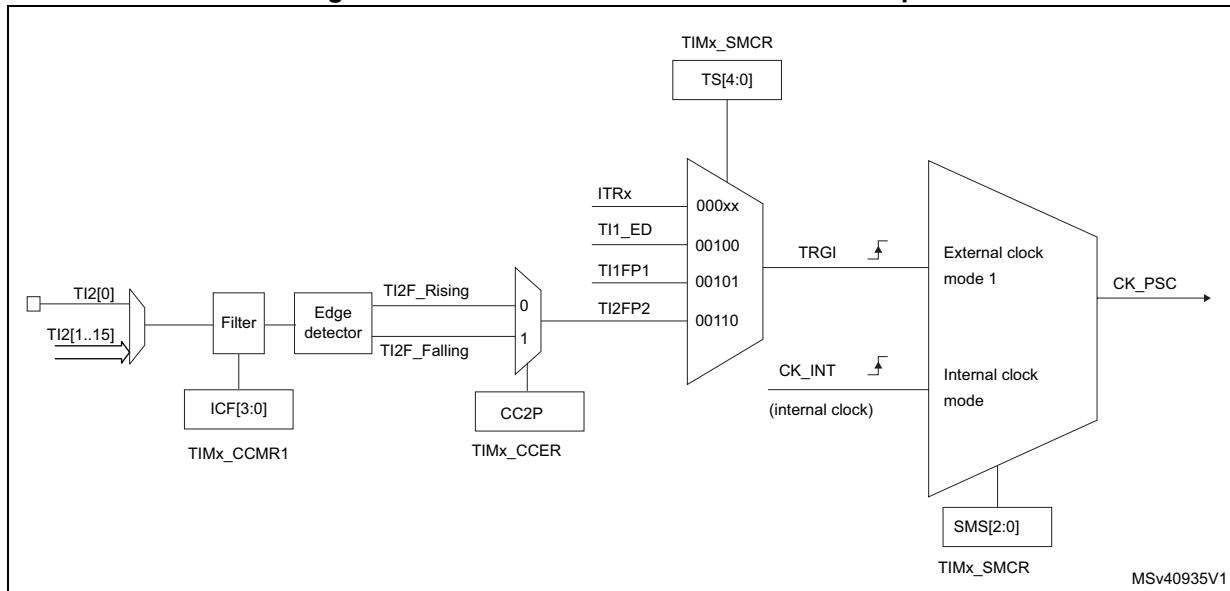
Figure 252. Control circuit in normal mode, internal clock divided by 1



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 253. 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:

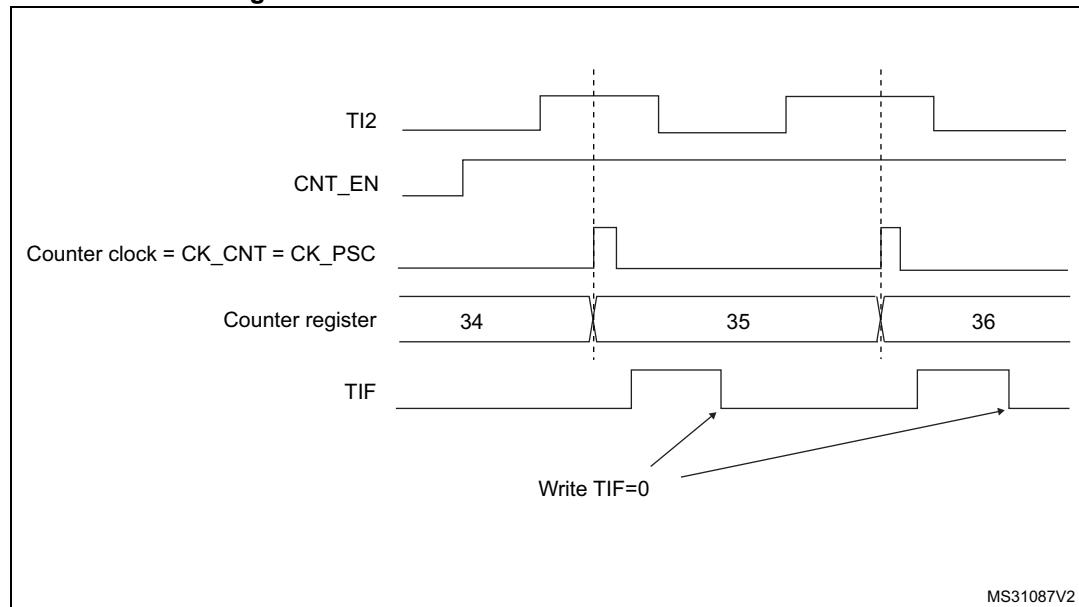
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the 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 254. Control circuit in external clock mode 1

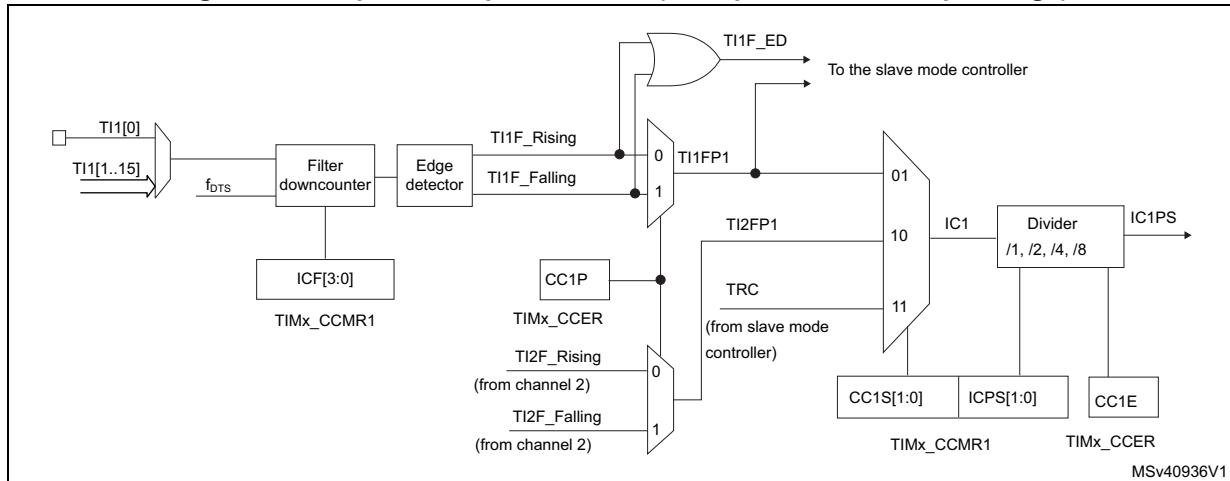


26.4.5 Capture/compare channels

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 255](#) to [Figure 258](#) 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 255. 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.

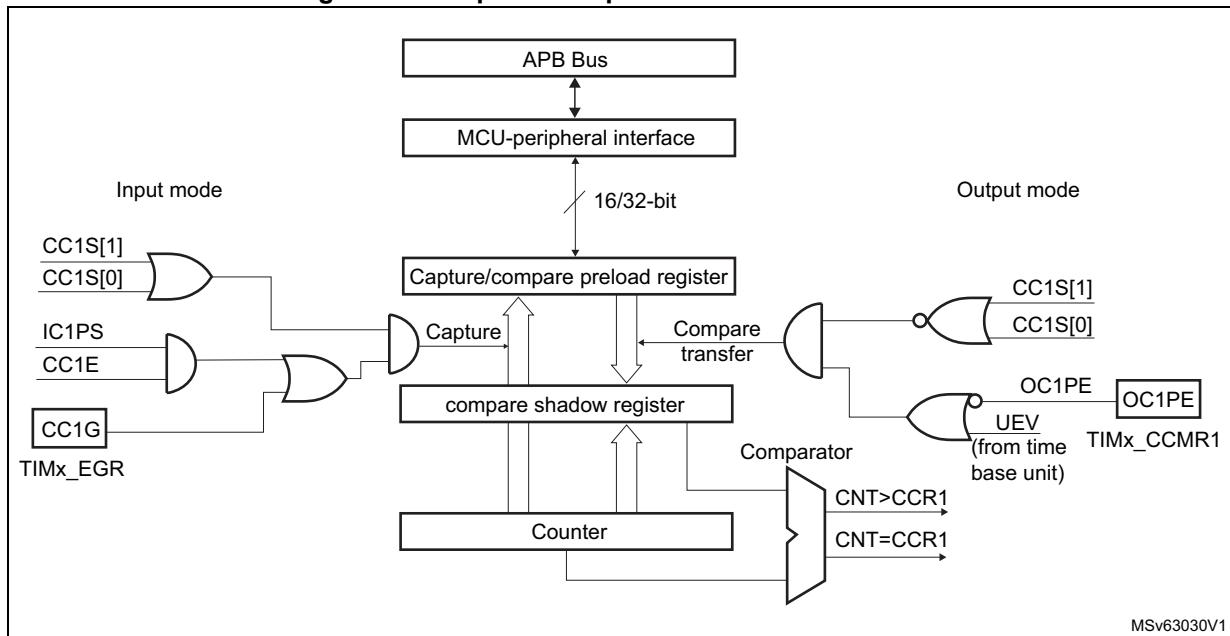
Figure 256. Capture/compare channel 1 main circuit

Figure 257. Output stage of capture/compare channel (channel 1)

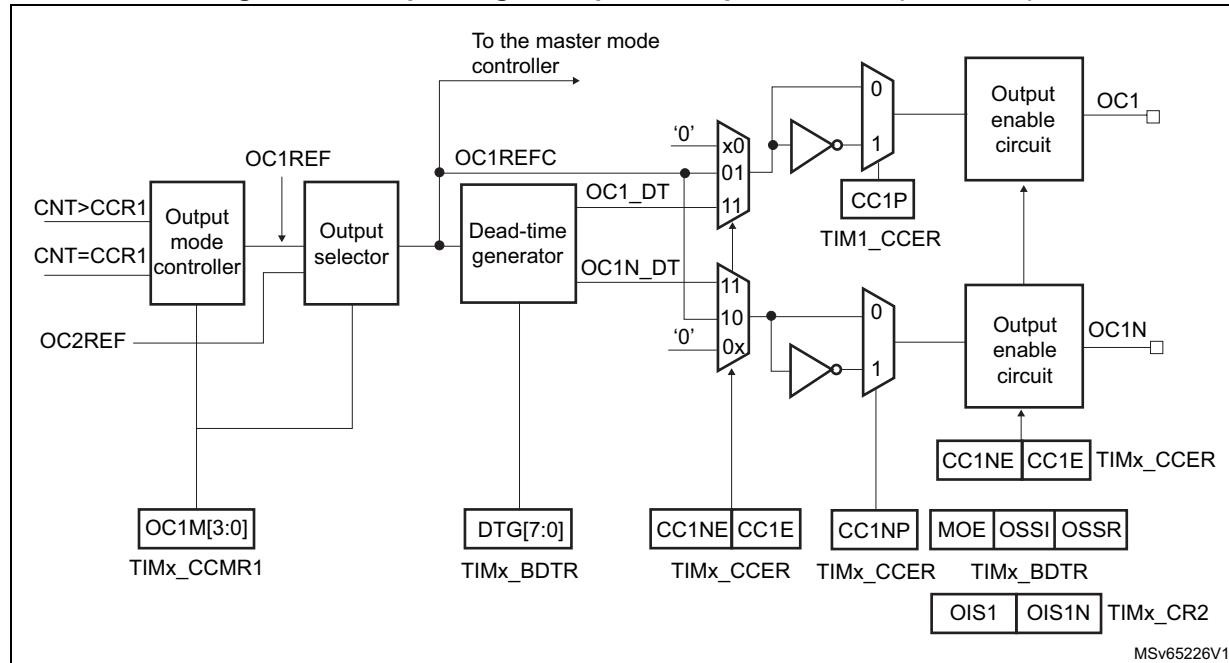
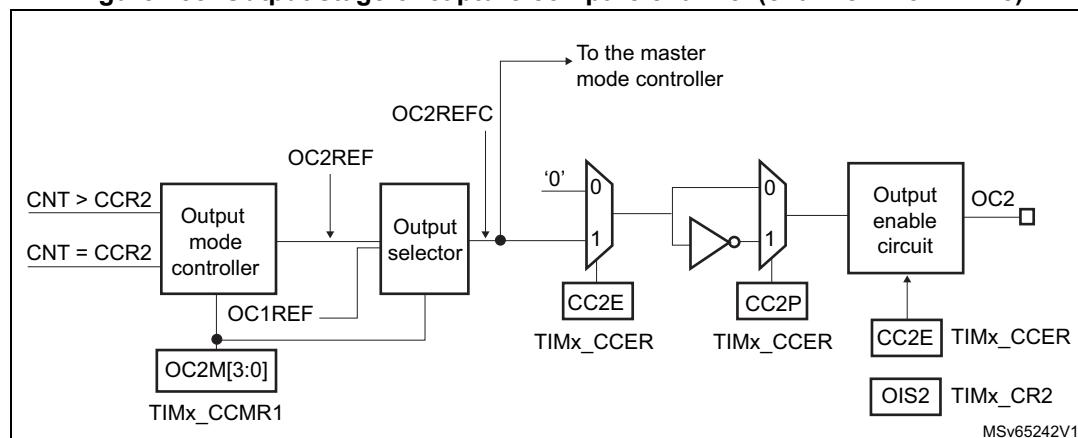


Figure 258. Output stage of capture/compare channel (channel 2 for TIM15)



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.

26.4.6 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.

26.4.7 PWM input mode (only for TIM15)

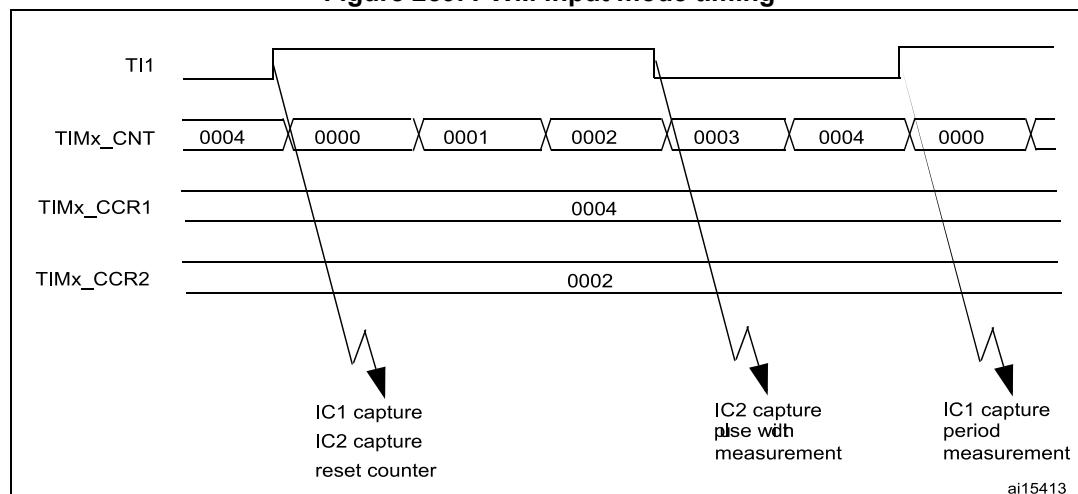
This mode is a particular case of input capture mode. The procedure is the same except:

- 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 TI1[x] 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 '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 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 259. 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.

26.4.8 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.

26.4.9 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.

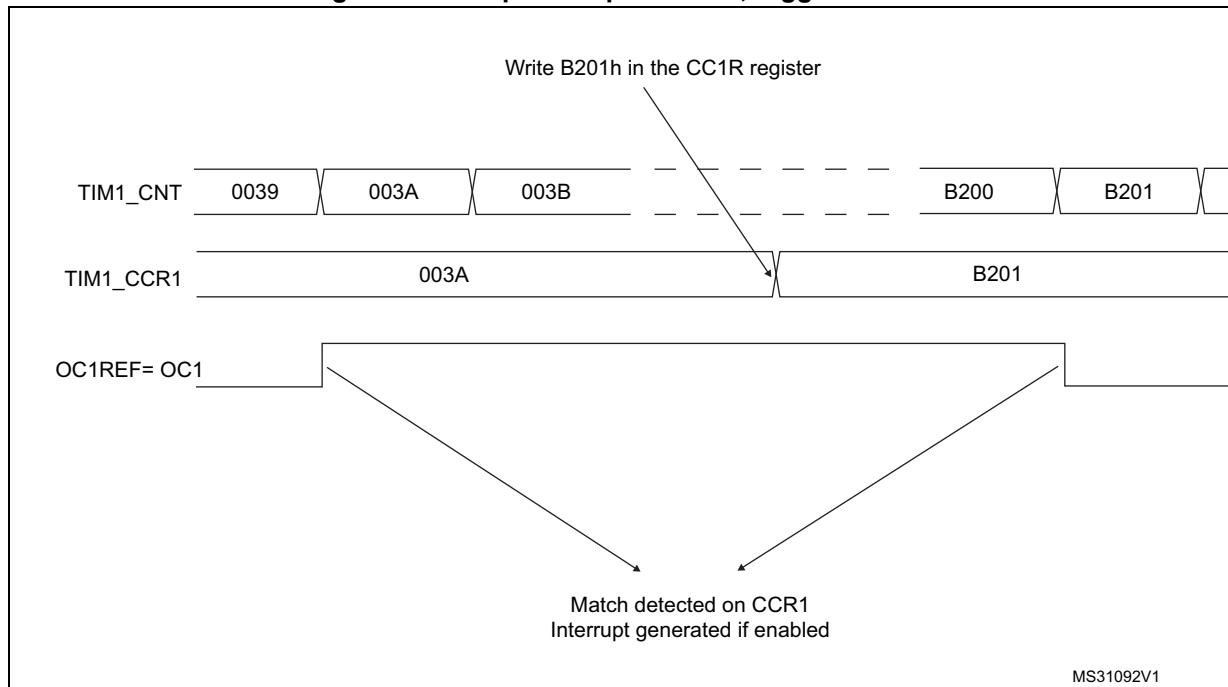
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 260](#).

Figure 260. Output compare mode, toggle on OC1



26.4.10 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.

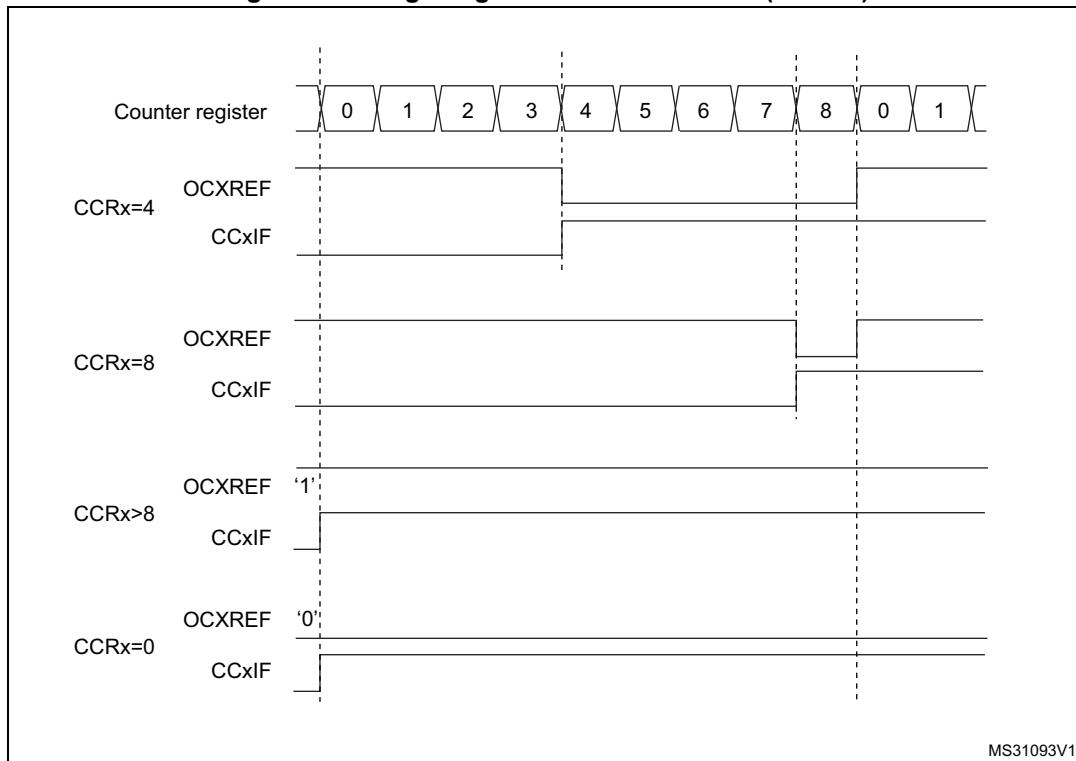
OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSS1 and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx \leq TIMx_CNT or TIMx_CNT \leq TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16 are capable of upcounting only. Refer to [Upcounting mode on page 745](#).

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 261](#) shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 261. Edge-aligned PWM waveforms (ARR=8)



26.4.11 Combined PWM mode (TIM15 only)

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 the TIMx_CCR1 and TIMx_CCR2 registers

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 a 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).

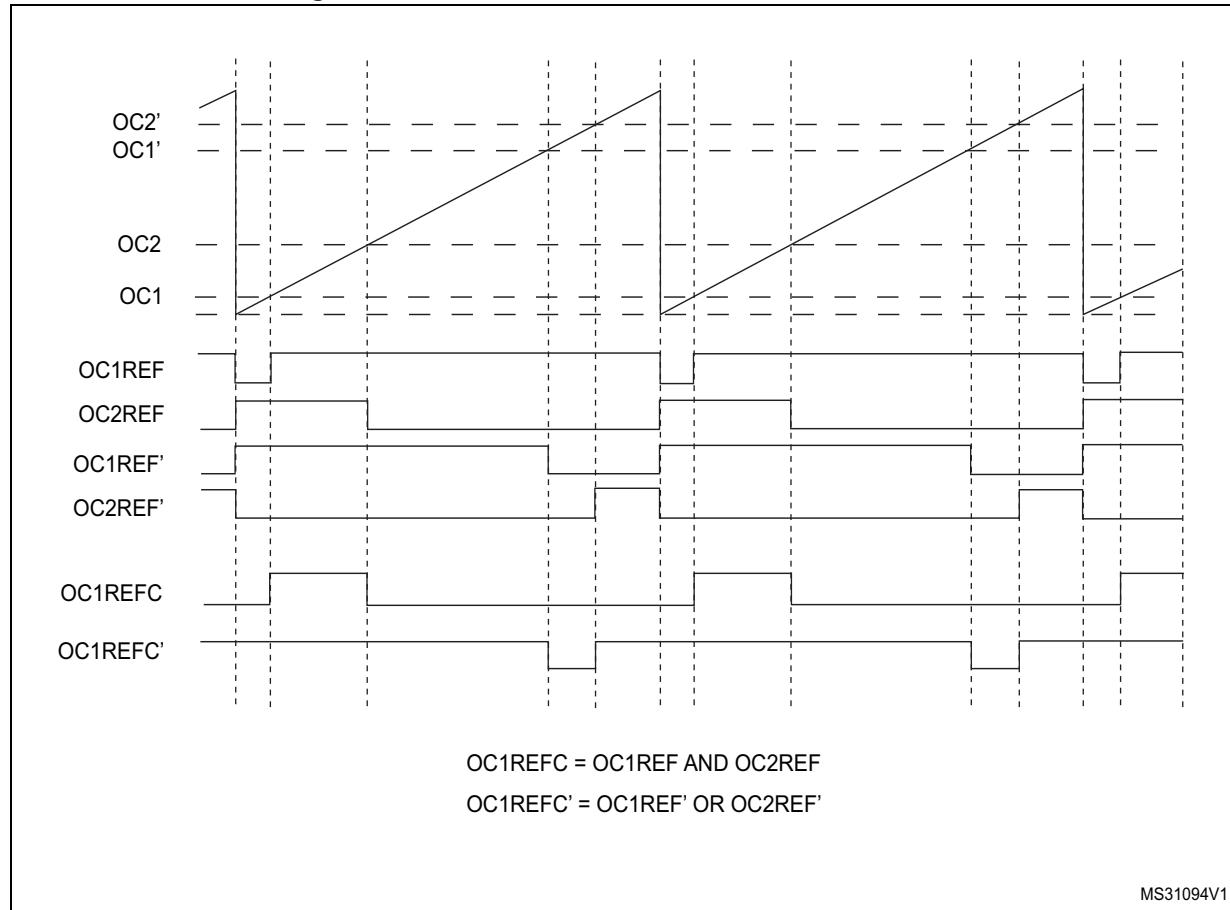
Note:

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 262 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,

Figure 262. Combined PWM mode on channel 1 and 2



26.4.12 Complementary outputs and dead-time insertion

The TIM15/TIM16 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 141: Output control bits for complementary \$OCx\$ and \$OCxN\$ channels with break feature \(TIM16/17\) on page 814](#) 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

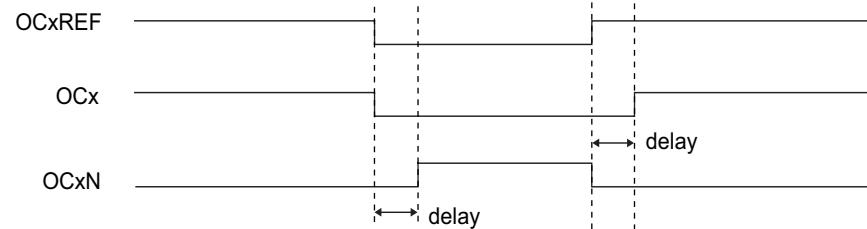
reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- 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.

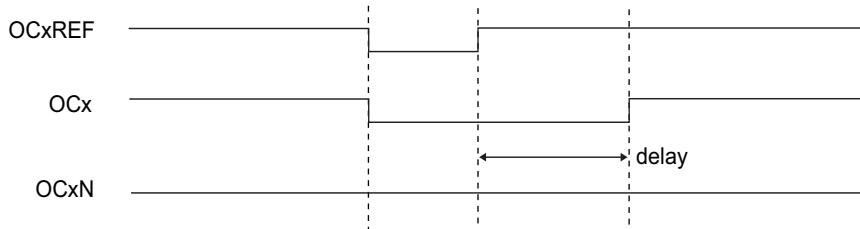
The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 263. Complementary output with dead-time insertion.

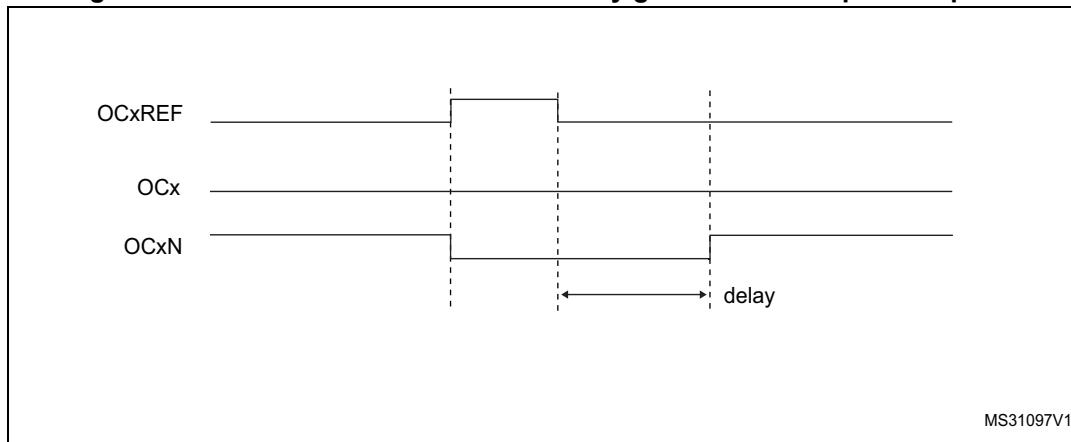


MS31095V1

Figure 264. Dead-time waveforms with delay greater than the negative pulse.



MS31096V1

Figure 265. 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 26.6.14: TIM16 break and dead-time register \(TIM16_BDTR\) on page 817](#) 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.

26.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/TIM16 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.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows to enable /disable the outputs by software and is reset in case of break event.
- the OSS1 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 shutdown 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 139: Output control bits for complementary OCx and OCxN channels with break feature \(TIM15\) on page 793](#) for more details.

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.

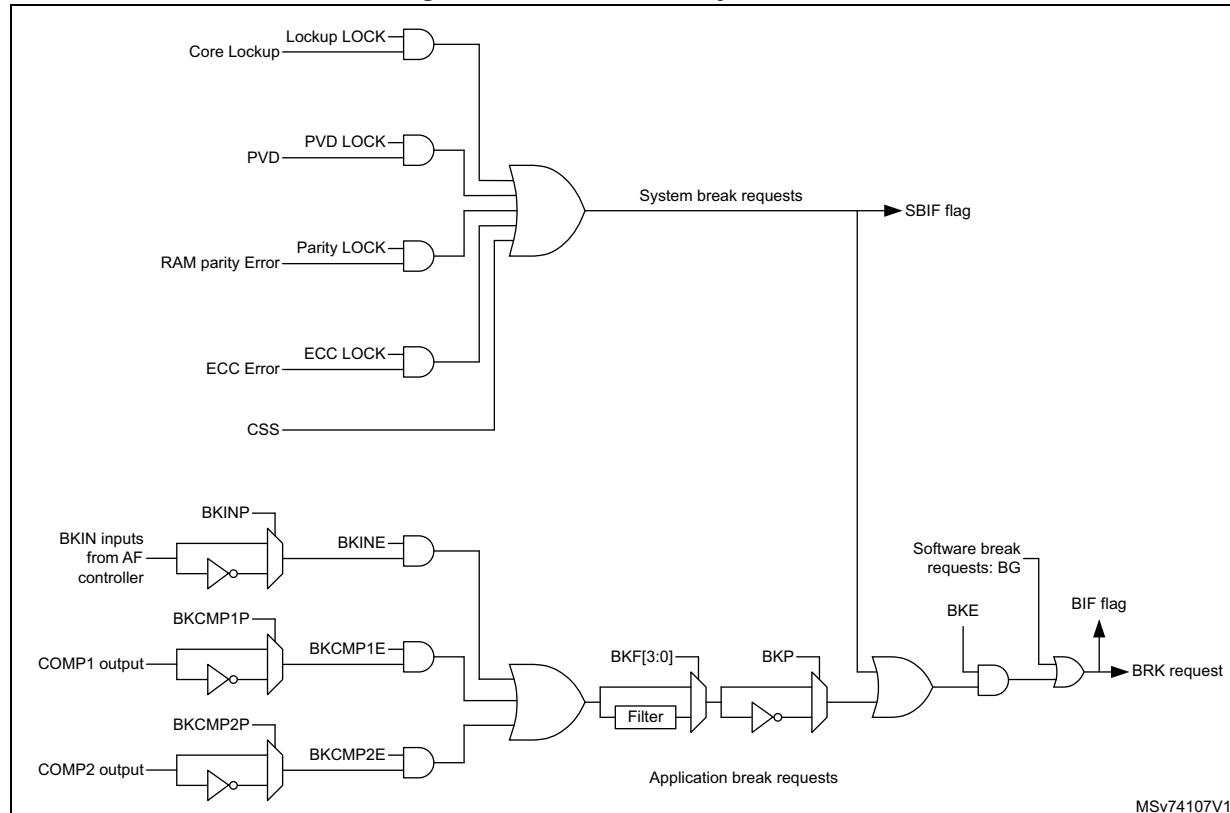
A programmable filter (BKF[3:0] bits in the TIMx_BDTR register allows to filter out spurious events.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering
- An internal source:
 - the output from a comparator, with polarity selection and optional digital filtering
 - A system break:
 - the Cortex®-M0+ LOCKUP output
 - the PVD output
 - the SRAM parity error signal
 - a flash ECC error
 - a clock failure event generated by the CSS detector

Figure 266. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

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 OSS1 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 OSS1=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 OSS1=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 OSS1 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.

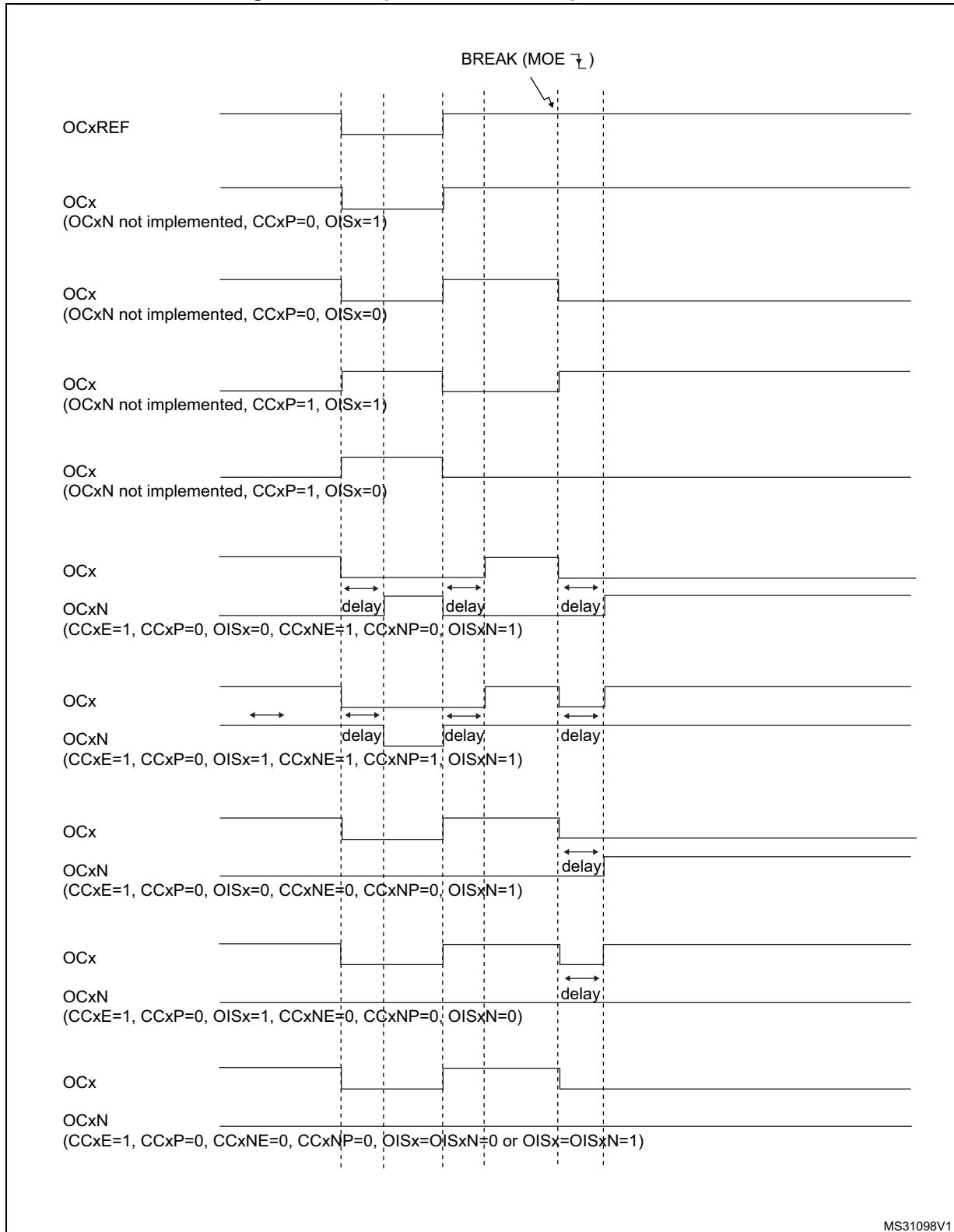
Note: *The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.*

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 26.6.14: TIM16 break and dead-time register \(TIM16_BDTR\) on page 817](#). The LOCK bits can be written only once after an MCU reset.

The [Figure 267](#) shows an example of behavior of the outputs in response to a break.

Figure 267. Output behavior in response to a break



MS31098V1

26.4.14 Bidirectional break inputs

The TIM15/TIM16 are featuring bidirectional break I/Os, as represented on [Figure 268](#).

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 restarted 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 137](#))

Table 137. 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

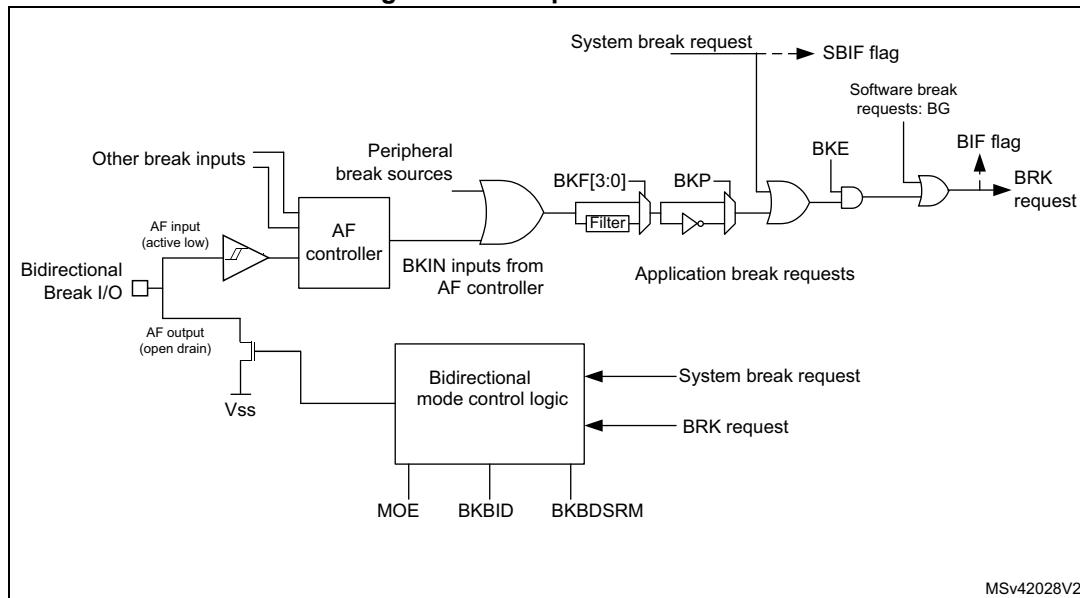
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

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 268. Output redirection



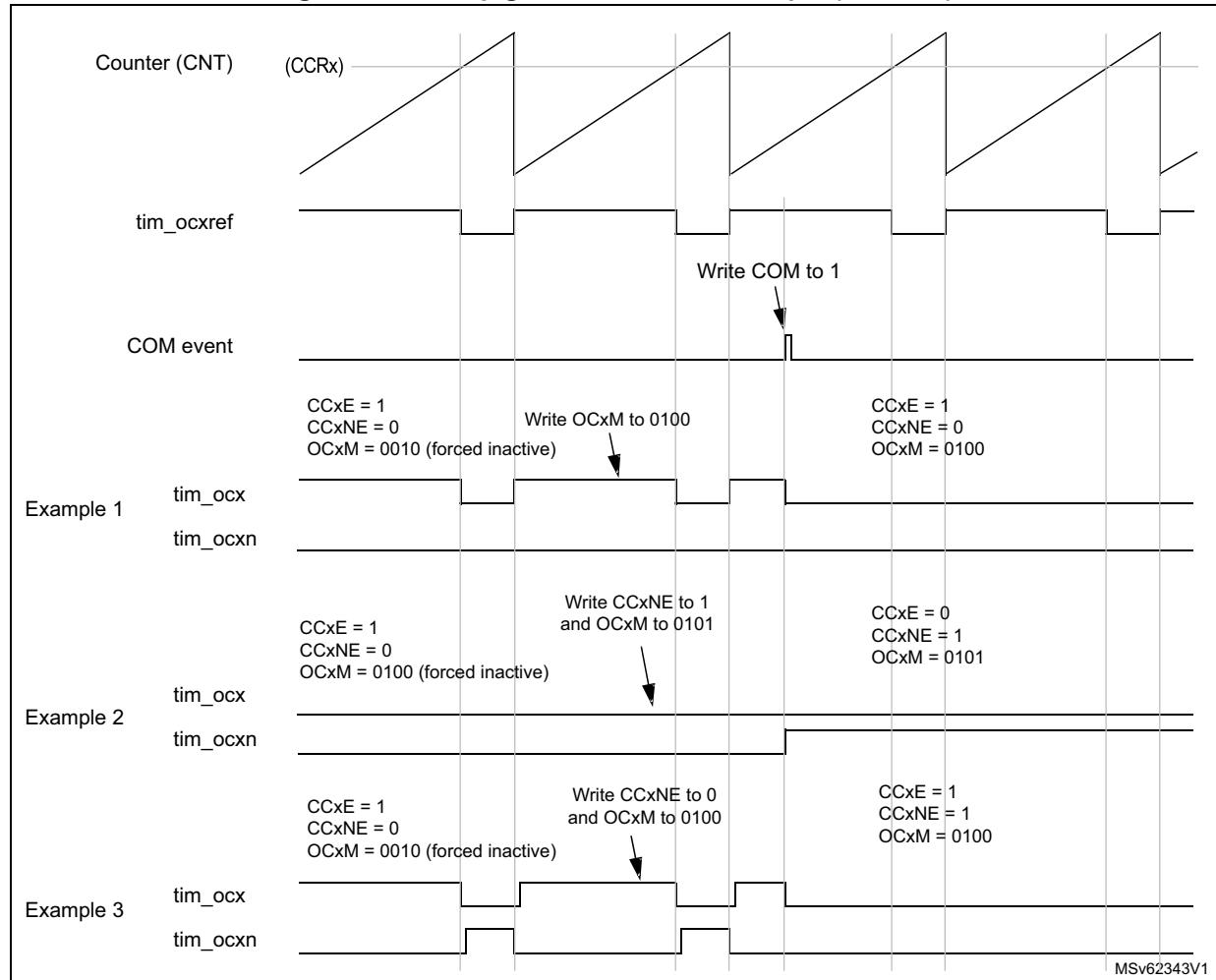
26.4.15 6-step PWM generation

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 269](#) describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 269. 6-step generation, COM example (OSSR=1)



26.4.16 One-pulse mode

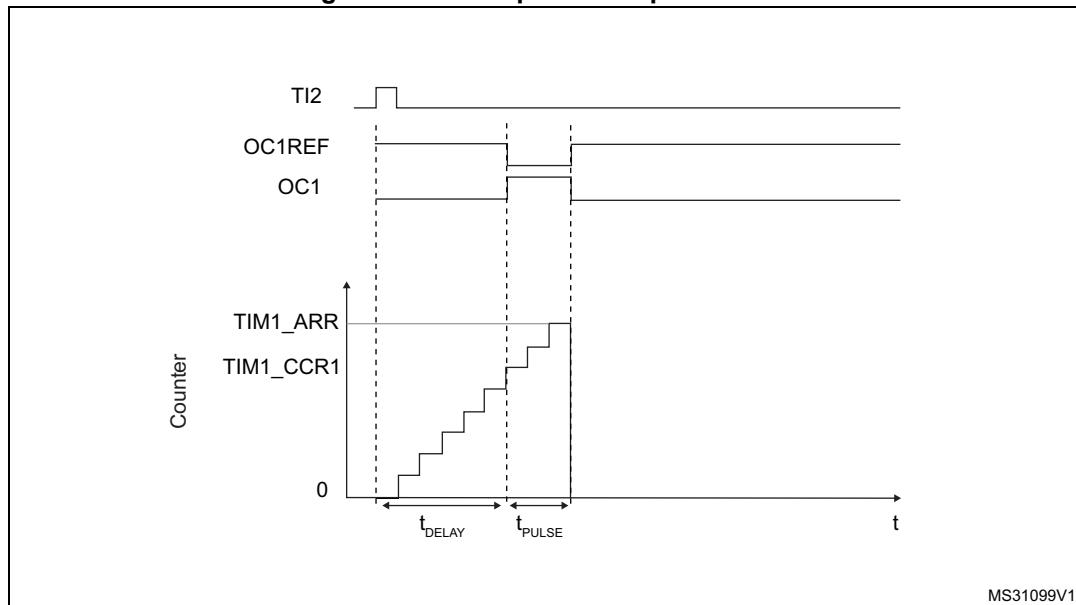
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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:

- $CNT < CCRx \leq ARR$ (in particular, $0 < CCRx$)

Figure 270. 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.

26.4.17 Retriggerable one pulse mode (TIM15 only)

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 26.4.16](#):

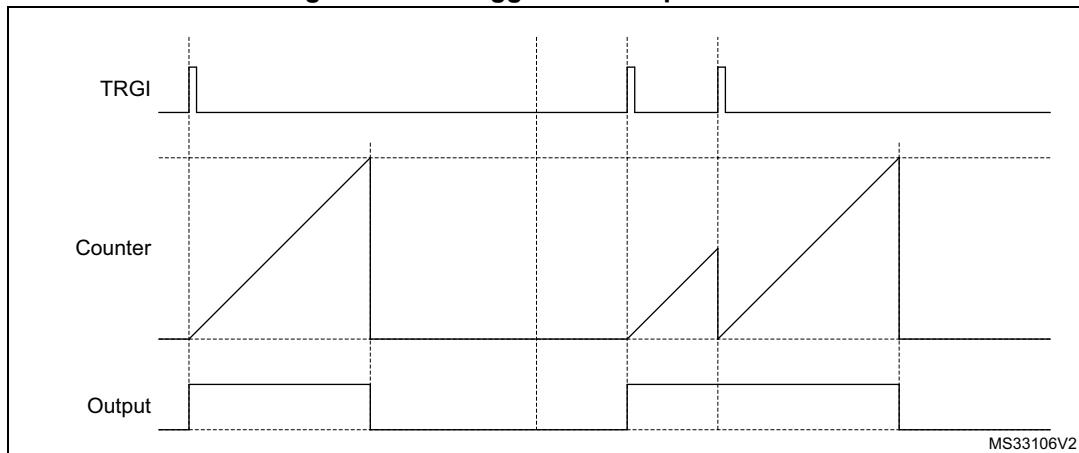
- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

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 271. Retriggerable one pulse mode

26.4.18 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).

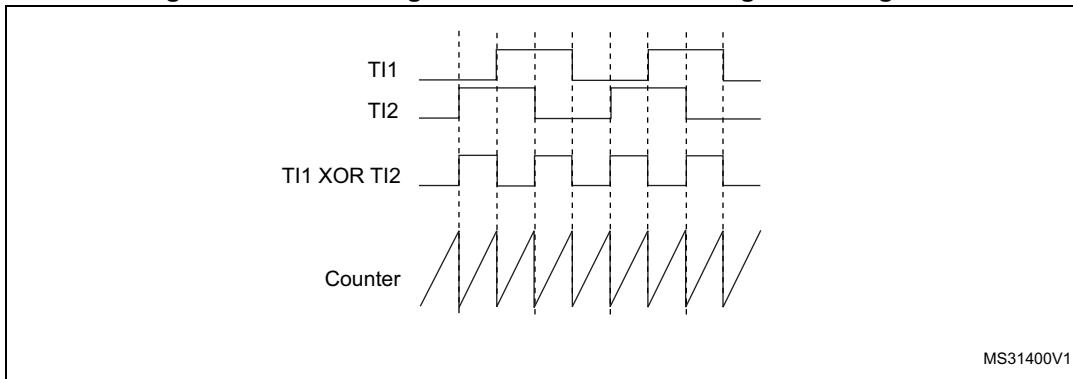
There is no latency between the assertions of the UIF and UIFCPY flags.

26.4.19 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 272*.

Figure 272. Measuring time interval between edges on 2 signals



MS31400V1

26.4.20 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

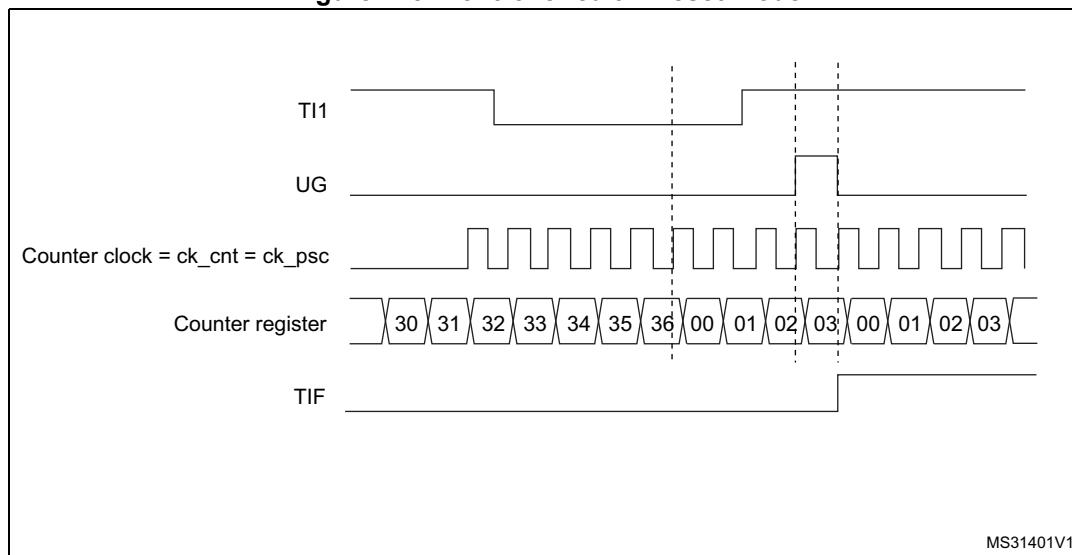
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in the 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, 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 273. Control circuit in reset mode



MS31401V1

Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

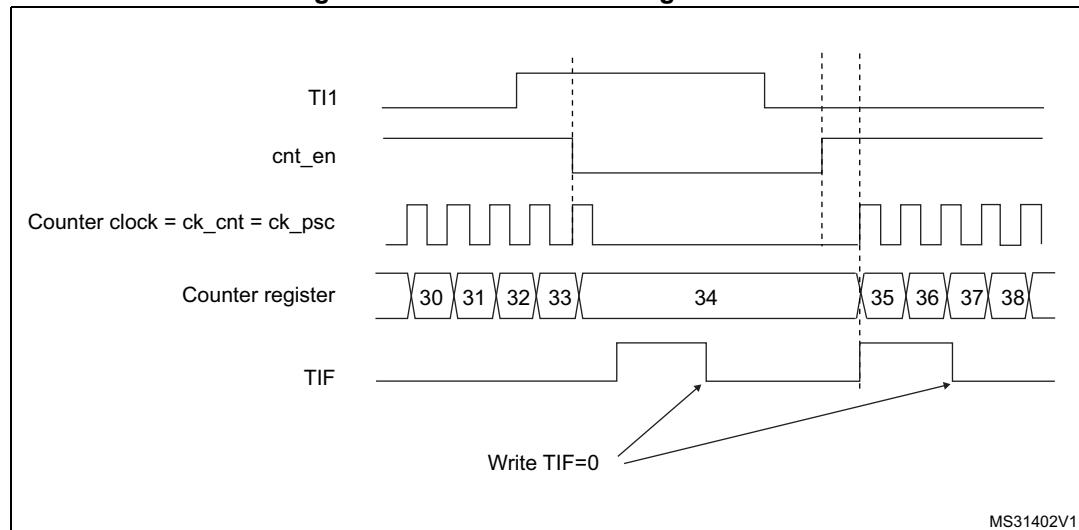
In the following example, the upcounter counts only when TI1 input is low:

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 the 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 274. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

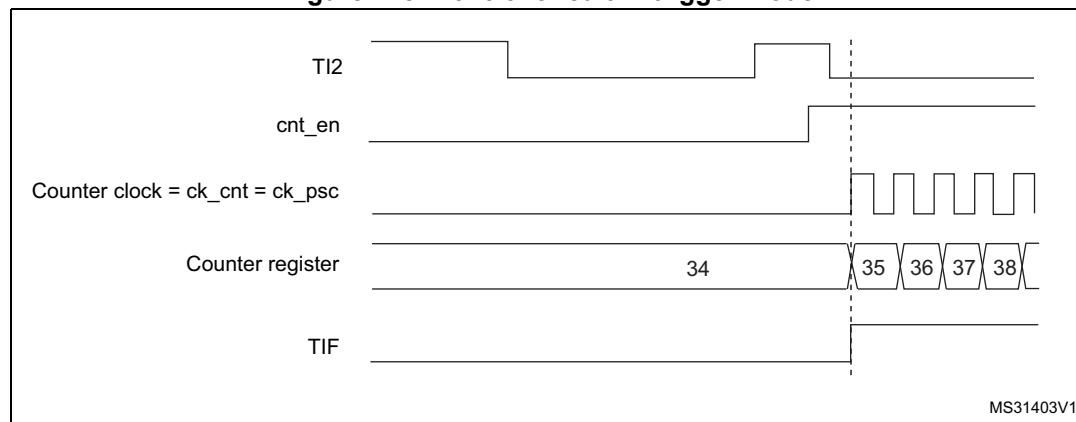
In the following example, the upcounter starts in response to a rising edge on TI2 input:

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. 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 the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in the 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 275. Control circuit in trigger mode



26.4.21 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.

26.4.22 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.

26.4.23 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to [Section 24.3.19: Timer synchronization](#) for details.

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.*

26.4.24 Using timer output as trigger for other timers (TIM16)

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.

26.4.25 Debug mode

When the microcontroller enters debug mode (Cortex®-M0+ 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 37: Debug support \(DBG\)](#).

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.

26.5 TIM15 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).

26.5.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x000

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|--------------|------|----------|----|------|------|------|------|-----|-----|------|-----|
| Res. | Res. | Res. | Res. | UIFRE MAP | Res. | CKD[1:0] | | ARPE | 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 bitfield 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 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 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.

26.5.2 TIM15 control register 2 (TIM15_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. | OIS2 | OIS1N | OIS1 | TI1S | MMS[2:0] | CCDS | CCUS | Res. | CCPC | | |

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

Bit 10 OIS2: Output idle state 2 (OC2 output)

0: OC2=0 when MOE=0

1: OC2=1 when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BDTR register).

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 TIM15_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 TIM15_BDTR register).

Bit 7 **TI1S**: TI1 selection

- 0: The TIMx_CH1 pin is connected to TI1 input
- 1: The TIMx_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

- 000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
- 001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic 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).

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.

26.5.3 TIM15 slave mode control register (TIM15_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. | 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 |
| Res. | MSM | TS[2:0] | Res. | Res. | Res. | Res. | SMS[2:0] | |
| | | | | | | | | 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.

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

Bit 7 **MSM**: Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 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)

Other: Reserved

See [Table 138: TIMx Internal trigger connection on page 783](#) 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 Reserved, must be kept at reset value.

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 (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Other codes: reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='00100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (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 138. TIMx Internal trigger connection

| Slave TIM | ITR0 (TS = 00000) | ITR1 (TS = 00001) | ITR2 (TS = 00010) | ITR3 (TS = 00011) |
|-----------|-------------------|-------------------|-------------------|-------------------|
| TIM15 | TIM2 | TIM3 | TIM16_OC1 | - |

26.5.4 TIM15 DMA/interrupt enable register (TIM15_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 | COMD E | Res. | Res. | Res. | CC1DE | UDE | BIE | TIE | COMIE | Res. | Res. | CC2IE | CC1IE | UIE |
| | 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

Bits 12:10 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 **TIE**: Trigger interrupt enable

- 0: Trigger interrupt disabled
- 1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable

- 0: COM interrupt disabled
- 1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

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

26.5.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|-------|-------|------|-----|-----|-------|------|------|-------|-------|-----|
| Res. | Res. | Res. | Res. | Res. | CC2OF | CC1OF | Res. | BIF | TIF | COMIF | Res. | Res. | CC2IF | CC1IF | UIF |

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

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 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is 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 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 4:3 Reserved, must be kept at reset value.

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 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 26.5.3: TIM15 slave mode control register \(TIM15_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx_CR1 register.

26.5.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|------|------|----|----|------|------|------|------|------|----|
| Res. | BG | TG | COMG | Res. | Res. | CC2G | CC1G | UG |

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

26.5.7 TIM15 capture/compare mode register 1 (TIM15_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 CC_xS 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).

26.5.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_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 |
| Res. | OC2M[2:0] | | | OC2 PE | OC2 FE | CC2S[1:0] | | Res. | OC1M[2:0] | | | OC1 PE | OC1 FE | CC1S[1:0] | |
| | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 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.
- 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.
- 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: Reserved,
- 1111: 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, 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.

On channels that have 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.

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. 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).

26.5.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|------|------|-------|------|------|------|-------|-------|------|------|
| Res. | CC2NP | Res. | CC2P | CC2E | CC1NP | CC1NE | CC1P | CC1E |

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

Bit 7 **CC2NP**: Capture/Compare 2 complementary 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 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" (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, OSS1, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, 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), TIxFP1 is 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, OSS1, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 139](#) for details.

Table 139. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)

| 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) OCx=0 OCxN=0, OCxN_EN=0 | |
| | | 0 | 0 | 1 | Output Disabled (not driven by the timer: Hi-Z) OCx=0 | OCxREF + Polarity OCxN=OCxREF XOR CCxNP |
| | | 0 | 1 | 0 | OCxREF + Polarity OCx=OCxREF XOR CCxP | Output Disabled (not driven by the timer: Hi-Z) 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) OCx=CCxP | OCxREF + Polarity OCxN=OCxREF XOR CCxNP |
| | | 1 | 1 | 0 | OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1 | Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1 |
| 0 | 0 | X | X | X | Output disabled (not driven by the timer: Hi-Z) | |
| | 0 | | 0 | 0 | | |
| | 1 | | 0 | 1 | Off-State (output enabled with inactive state) | |
| | 1 | | 1 | 0 | Asynchronously: OCx=CCxP, OCxN=CCxNP | |
| | 1 | | 1 | 1 | 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 | |

- 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.

26.5.10 TIM15 counter (TIM15_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. |
| r | | | | | | | | | | | | | | | |
| 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 in the TIMx_ISR register.

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

Bits 15:0 **CNT[15:0]**: Counter value

26.5.11 TIM15 prescaler (TIM15_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”).

26.5.12 TIM15 auto-reload register (TIM15_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 26.4.1: Time-base unit on page 743](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

26.5.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | |
|------|------|------|------|------|------|------|------|----------|----|----|----|----|----|----|----|--|--|--|--|--|--|
| 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.

26.5.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CCR1[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

26.5.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CCR2[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

26.5.16 TIM15 break and dead-time register (TIM15_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. | Res. | BKBID | Res. | BKDSRM | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| | | | rw | | rw | | | | | | | rw | rw | rw | rw | |
| BKF[3:0] | | | | | | | | | | | | | | | | |
| 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, BKF[3:0], AOE, BKP, BKE, 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 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample the BRK input signal and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

- 0000: No filter, BRK acts asynchronously
- 0001: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=2
- 0010: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=4
- 0011: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=8
- 0100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N=6
- 0101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N=8
- 0110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N=6
- 0111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N=8
- 1000: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N=6
- 1001: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N=8
- 1010: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=5
- 1011: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=6
- 1100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=8
- 1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N=5
- 1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N=6
- 1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, 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 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 OSS1 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 26.5.9: TIM15 capture/compare enable register \(TIM15_CCER\) on page 791](#)).

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 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
- 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 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 26.5.9: TIM15 capture/compare enable register \(TIM15_CCER\) on page 791](#)).

- 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 26.5.9: TIM15 capture/compare enable register \(TIM15_CCER\) on page 791](#)).

- 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.

$DTG[7:5] = 0xx \Rightarrow DT = DTG[7:0] \times t_{dtg}$ with $t_{dtg} = t_{DTS}$

$DTG[7:5] = 10x \Rightarrow DT = (64+DTG[5:0]) \times t_{dtg}$ with $t_{dtg} = 2 \times t_{DTS}$

$DTG[7:5] = 110 \Rightarrow DT = (32+DTG[4:0]) \times t_{dtg}$ with $t_{dtg} = 8 \times t_{DTS}$

$DTG[7:5] = 111 \Rightarrow DT = (32+DTG[4:0]) \times t_{dtg}$ 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 μ s to 31750 ns by 250 ns steps,

32 μ s to 63 μ s by 1 μ s steps,

64 μ s to 126 μ s by 2 μ s 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).

26.5.17 TIM15 DMA control register (TIM15_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] | | | |

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).

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,

...

26.5.18 TIM15 DMA address for full transfer (TIM15_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
 $(\text{TIMx_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$

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).

26.5.19 TIM15 alternate register 1 (TIM15_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. | BKCM P2P | BKCM P1P | BKINP | Res. | Res. | Res. | Res. | Res. | Res. | BKCM P2E | BKCM P1E | BKINE |
| | | | | rw | rw | rw | | | | | | | rw | rw | rw |

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCM2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low
1: COMP2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCM1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low
1: COMP1 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low
1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCM2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCM1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN 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).

26.5.20 TIM15 input selection register (TIM15_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]**: selects TI2[0] to TI2[15] input

0000: TIM15_CH2 input

0001: TIM2_IC2

0010: TIM3_IC2

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: TIM15_CH1 input

0001: TIM2_IC1

0010: TIM3_IC1

Others: Reserved

26.5.21 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 140. TIM15 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 | TIM15_CR1 | Res. | |
| | Reset value | Res. | |
| 0x04 | TIM15_CR2 | Res. | |
| | Reset value | Res. | |
| 0x08 | TIM15_SMCR | Res. | |
| | Reset value | Res. | |
| 0x0C | TIM15_DIER | Res. | |
| | Reset value | Res. | |
| 0x10 | TIM15_SR | Res. | |
| | Reset value | Res. | |
| 0x14 | TIM15_EGR | Res. | |
| | Reset value | Res. | |
| 0x18 | TIM15_CCMR1 Output Compare mode | Res. | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | TIM15_CCMR1 Input Capture mode | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TIM15_CCER | Res. |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 140. TIM15 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 | TIM15_CNT | UIFCPY or Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | 0 | Res. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x28 | TIM15_PSC | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2C | TIM15_ARR | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x30 | TIM15_RCR | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x34 | TIM15_CCR1 | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x38 | TIM15_CCR2 | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x44 | TIM15_BDTR | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x48 | TIM15_DCR | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4C | TIM15_DMAR | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x60 | TIM15_AF1 | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x68 | TIM15_TISEL | Res. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

26.6 TIM16 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).

26.6.1 TIM16 control register 1 (TIM16_CR1)

Address offset: 0x000

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|--------------|------|----------|----|------|------|------|------|-----|-----|------|-----|
| Res. | Res. | Res. | Res. | UIFRE MAP | Res. | CKD[1:0] | | ARPE | 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.

26.6.2 TIM16 control register 2 (TIM16_CR2)

Address offset: 0x04

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|-------|------|------|------|------|------|------|------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | OIS1N | OIS1 | Res. | Res. | Res. | Res. | CCDS | CCUS | Res. | CCPC |

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.

26.6.3 TIM16 DMA/interrupt enable register (TIM16_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. | Res. | Res. | Res. | CC1DE | UDE | BIE | Res. | COMIE | Res. | Res. | Res. | CC1IE | UIE |

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

26.6.4 TIM16 status register (TIM16_SR)

Address offset: 0x10

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|-------|------|-----|------|-------|------|------|------|-------|-----|
| Res. | Res. | Res. | Res. | Res. | Res. | CC1OF | Res. | BIF | Res. | COMIF | Res. | Res. | Res. | CC1IF | UIF |

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.

26.6.5 TIM16 event generation register (TIM16_EGR)

Address offset: 0x14

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|----|-----|------|-----|-----|-----|------|----|
| 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).

26.6.6 TIM16 capture/compare mode register 1 (TIM16_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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).

26.6.7 TIM16 capture/compare mode register 1 [alternate] (TIM16_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. | OC1M [3] |
| | | | | | | | | | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | rw | rw | rw | rw | rw | rw | CC1S[1:0] |
| | | | | | | | | | 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).

26.6.8 TIM16 capture/compare enable register (TIM16_CCER)

Address offset: 0x20

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|------|------|
| Res. | CC1NP | CC1NE | CC1P | CC1E |

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, OSS1, OSSR, OIS1, OIS1N and CC1E bits.
- 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, 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), TIxFP1 is 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, OSS1, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 141](#) for details.

Table 141. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)

| 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) OCx=0 OCxN=0, OCxN_EN=0 | |
| | | 0 | 0 | 1 | Output Disabled (not driven by the timer: Hi-Z) OCx=0 | OCxREF + Polarity OCxN=OCxREF XOR CCxNP |
| | | 0 | 1 | 0 | OCxREF + Polarity OCx=OCxREF XOR CCxP | Output Disabled (not driven by the timer: Hi-Z) 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) OCx=CCxP | OCxREF + Polarity OCxN=OCxREF XOR CCxNP |
| | | 1 | 1 | 0 | OCxREF + Polarity OCx=OCxREF XOR CCxP, OCx_EN=1 | Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1 |
| 0 | 0 | X | X | X | Output disabled (not driven by the timer: Hi-Z). | |
| | 0 | | 0 | 0 | | |
| | 1 | | 0 | 1 | Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP | |
| | 1 | | 1 | 0 | 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 | |
| | 1 | | 1 | 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.

26.6.9 TIM16 counter (TIM16_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. |
| r | | | | | | | | | | | | | | | |
| 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

26.6.10 TIM16 prescaler (TIM16_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”).

26.6.11 TIM16 auto-reload register (TIM16_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 26.4.1: Time-base unit on page 743](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

26.6.12 TIM16 repetition counter register (TIM16_RCR)

Address offset: 0x30

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | |
|------|------|------|------|------|------|------|------|----------|----|----|----|----|----|----|----|--|--|--|--|--|--|
| 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.

26.6.13 TIM16 capture/compare register 1 (TIM16_CCR1)

Address offset: 0x34

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CCR1[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

26.6.14 TIM16 break and dead-time register (TIM16_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. | Res. | BKBID | Res. | BKDSRM | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | BKF[3:0] |
| | | | 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 BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, 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 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 events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously

0001: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=2

0010: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=4

0011: $f_{\text{SAMPLING}} = f_{\text{CK_INT}}$, N=8

0100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N=6

0101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N=8

0110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N=6

0111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N=8

1000: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N=6

1001: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N=8

1010: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=5

1011: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=6

1100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N=8

1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N=5

1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N=6

1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N=8

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 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 OSS1 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 26.6.8: TIM16 capture/compare enable register \(TIM16_CCER\) on page 812](#)).

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 26.6.8: TIM16 capture/compare enable register \(TIM16_CCER\) on page 812](#)).

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 26.6.8: TIM16 capture/compare enable register \(TIM16_CCER\) on page 812](#)).

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 x t_{DTS}

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t_{dtg} with t_{dtg} = 8 x t_{DTS}

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t_{dtg} with t_{dtg} = 16 x 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).

26.6.15 TIM16 DMA control register (TIM16_DCR)

Address offset: 0x48

Reset value: 0x0000

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

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

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,

00001: 2 transfers,

00010: 3 transfers,

...

10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: 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.

26.6.16 TIM16 DMA address for full transfer (TIM16_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) × 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).

26.6.17 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. | BKCM P2P | BKCM P1P | BKINP | Res. | Res. | Res. | Res. | Res. | Res. | BKCM P2E | BKCM P1E | BKINE |
| | | | | rw | rw | rw | | | | | | | rw | rw | rw |

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCM2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: COMP2 input is active low
- 1: COMP2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCM1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: COMP1 input is active low
- 1: COMP1 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: BKIN input is active low
- 1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCM2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCM1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN 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).

26.6.18 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. | 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
- 0001: LSI
- 0010: LSE
- 0011: RTC wakeup
- 0100: MCO2
- Others: Reserved

26.6.19 TIM16 register map

TIM16 register is mapped as 16-bit addressable register as described in the table below:

Table 142. TIM16 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 |
|------------|--|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|
| 0x00 | TIMx_CR1 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 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. | Res. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 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. | Res. | Res. | Res. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 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. | Res. | Res. | Res. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 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. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 0x18 | TIMx_CCMR1 Output Compare 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. | Res. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 0x20 | 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. | Res. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 0x24 | 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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 0x28 | TIMx_CNT | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 0x2C | TIMx_PSC | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 0x24 | TIMx_ARR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| CNT[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PSC[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ARR[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| OC1M[2:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| IC1F[3:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| IC1NP[1:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| IC1NE[1:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CC1S[1:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CC1G[0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CC1IE[0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CC1UF[0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| UDIS[0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CEN[0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 142. TIM16 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. | | |
| | 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 | | |
| 0x34 | TIMx_CCR1 | Res. | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x44 | TIMx_BDTR | Res. | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x48 | TIMx_DCR | Res. | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x4C | TIMx_DMAR | Res. | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x60 | TIM16_AF1 | Res. | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x68 | TIM16_TISEL | Res. | | |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

27 Low-power timer (LPTIM)

27.1 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.

27.2 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 LP oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit capture/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
- Repetition counter
- Up to 4 independent channels for:
 - Input capture
 - PWM generation (edge-aligned mode)
 - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
 - Update event
 - Input capture

27.3 LPTIM implementation

The table below describes LPTIM implementation on STM32U0 series devices. The full set of features is implemented in LPTIM1 and LPTIM2. LPTIM3 supports a smaller set of features.

Table 143. STM32U0 series LPTIM features

| LPTIM modes/features ⁽¹⁾ | LPTIM1 | LPTIM2 | LPTIM3 ⁽²⁾ |
|-------------------------------------|--------|--------|-----------------------|
| Encoder mode | X | X | X |
| PWM mode | X | X | X |
| Input Capture | X | X | X |
| Number of channels | 4 | 2 | 4 |
| Number of DMA requests | 5 | 5 | 3 |
| Wake-up in Stop mode | X | X | X |

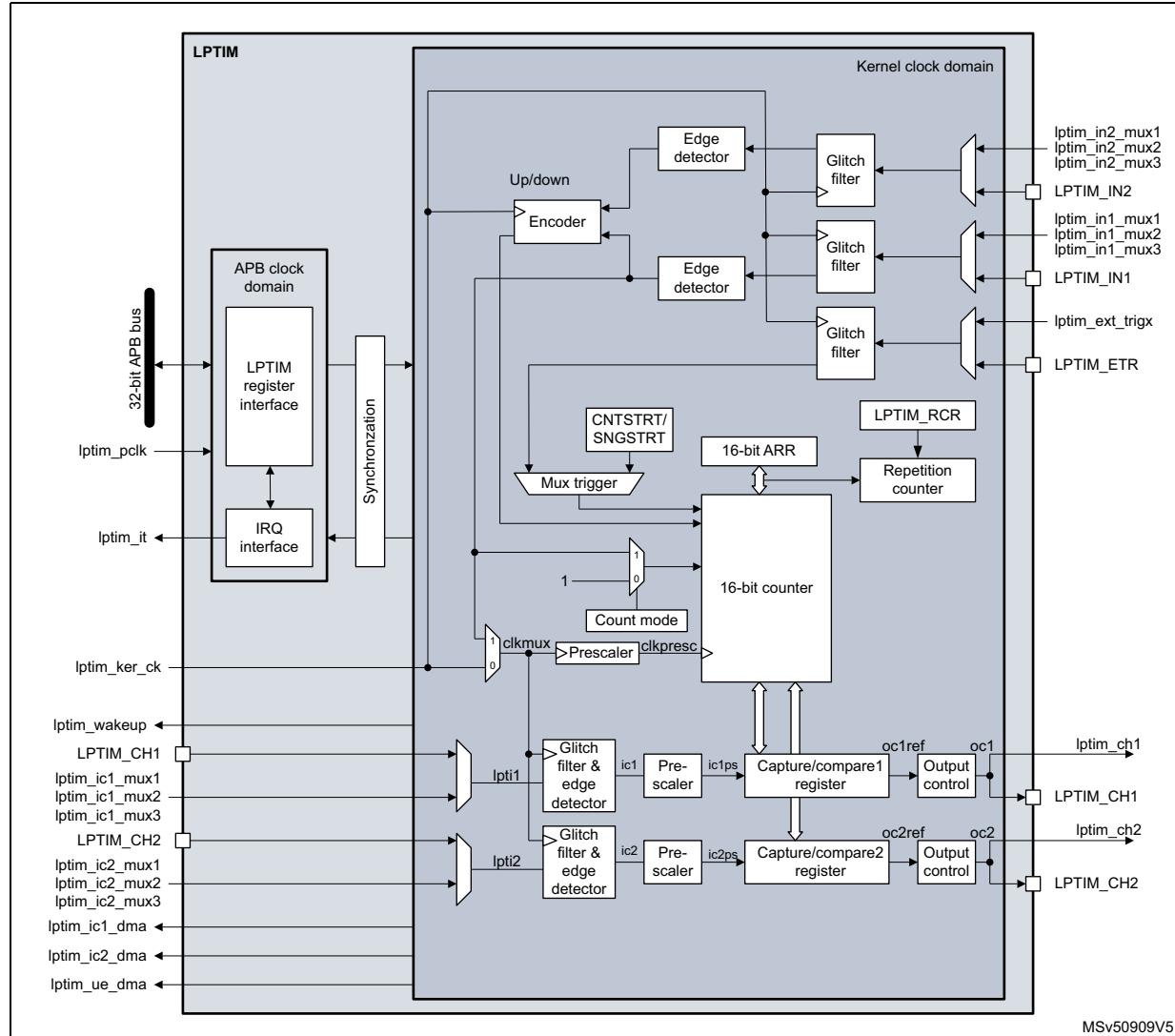
1. X = supported.

2. LPTIM3 is only available on STM32U073/U083 devices.

27.4 LPTIM functional description

27.4.1 LPTIM block diagram

Figure 276. LPTIM1/2/3 timer block diagram⁽¹⁾



1. Some IOs may not be available, refer to [Section 27.4.2: LPTIM pins and internal signals](#).

27.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 144. LPTIM1/2/3 input/output pins

| Names | Signal type | Description |
|-----------|---------------|--|
| LPTIM_IN1 | Digital input | LPTIM Input 1 from GPIO pin on mux input 0 |
| LPTIM_IN2 | Digital input | LPTIM Input 2 from GPIO pin on mux input 0 |
| LPTIM_ETR | Digital input | LPTIM external trigger GPIO pin |

Table 144. LPTIM1/2/3 input/output pins (continued)

| Names | Signal type | Description |
|-----------|----------------------|---------------------------------------|
| LPTIM_CH1 | Digital input/output | LPTIM channel 1 input/output GPIO pin |
| LPTIM_CH2 | Digital input/output | LPTIM channel 2 input/output GPIO pin |
| LPTIM_CH3 | Digital input/output | LPTIM channel 3 input/output GPIO pin |
| LPTIM_CH4 | Digital input/output | LPTIM channel 4 input/output GPIO pin |

Table 145. LPTIM1/2/3 internal signals

| Names | Signal type | Description |
|-----------------|----------------|---|
| lptim_pcclk | Digital input | LPTIM APB clock domain |
| lptim_ker_ck | Digital input | LPTIM kernel clock |
| lptim_in1_mux1 | Digital input | Internal LPTIM input 1 connected to mux input 1 |
| lptim_in1_mux2 | Digital input | Internal LPTIM input 1 connected to mux input 2 |
| lptim_in1_mux3 | Digital input | Internal LPTIM input 1 connected to mux input 3 |
| lptim_in2_mux1 | Digital input | Internal LPTIM input 2 connected to mux input 1 |
| lptim_in2_mux2 | Digital input | Internal LPTIM input 2 connected to mux input 2 |
| lptim_in2_mux3 | Digital input | Internal LPTIM input 2 connected to mux input 3 |
| lptim_ic1_mux1 | Digital input | Internal LPTIM input capture 1 connected to mux input 1 |
| lptim_ic1_mux2 | Digital input | Internal LPTIM input capture 1 connected to mux input 2 |
| lptim_ic1_mux3 | Digital input | Internal LPTIM input capture 1 connected to mux input 3 |
| lptim_ic2_mux1 | Digital input | Internal LPTIM input capture 2 connected to mux input 1 |
| lptim_ic2_mux2 | Digital input | Internal LPTIM input capture 2 connected to mux input 2 |
| lptim_ic2_mux3 | Digital input | Internal LPTIM input capture 2 connected to mux input 3 |
| lptim_ext_trigx | Digital input | LPTIM external trigger input x |
| lptim_it | Digital output | LPTIM global interrupt |
| lptim_wakeup | Digital output | LPTIM wake-up event |
| lptim_ic1_dma | Digital output | LPTIM input capture 1 DMA request |
| lptim_ic2_dma | Digital output | LPTIM input capture 2 DMA request |
| lptim_ue_dma | Digital output | LPTIM update event DMA request |

27.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 146. LPTIM1/2/3 external trigger connection

| TRIGSEL | External trigger | | |
|-----------------|------------------|--------------|----------|
| | LPTIM1 | LPTIM2 | LPTIM3 |
| Iptim_ext_trig0 | GPIO | | |
| Iptim_ext_trig1 | rtc_alra_trg | Iptim2_ch1 | |
| Iptim_ext_trig2 | rtc_alrb_trg | Reserved | |
| Iptim_ext_trig3 | tamp_trg1 | Iptim4_out | |
| Iptim_ext_trig4 | tamp_trg2 | Iptim5_out | |
| Iptim_ext_trig5 | tamp_trg3 | sai1_fs_a_in | |
| Iptim_ext_trig6 | Reserved | sai1_fs_b_in | |
| Iptim_ext_trig7 | Reserved | Reserved | Reserved |

Table 147. LPTIM1/2/3 input 1 connection

| Iptim_in1_mux | LPTIM1/2/3 input 1 connected to |
|----------------|---------------------------------|
| Iptim_in1_mux0 | GPIO |
| Iptim_in1_mux1 | Reserved |
| Iptim_in1_mux2 | Reserved |
| Iptim_in1_mux3 | Reserved |

Table 148. LPTIM1/2/3 input 2 connection

| Iptim_in2_mux | LPTIM1/2/3 input 2 connected to |
|----------------|---------------------------------|
| Iptim_in2_mux0 | GPIO |
| Iptim_in2_mux1 | Reserved |
| Iptim_in2_mux2 | Reserved |
| Iptim_in2_mux3 | Reserved |

Table 149. LPTIM1/2/3 input capture 1 connection

| Iptim_ic1_mux | LPTIM1 input capture 1 connected to | LPTIM2 input capture 1 connected to | LPTIM3 input capture 1 connected to |
|----------------|-------------------------------------|-------------------------------------|-------------------------------------|
| Iptim_ic1_mux0 | GPIO | GPIO | GPIO |
| Iptim_ic1_mux1 | Reserved | I3C1 IBI request (_IBIAck) | I3C2 IBI request (_IBIAck) |
| Iptim_ic1_mux2 | Reserved | Reserved | Reserved |
| Iptim_ic1_mux3 | Reserved | Reserved | Reserved |

Table 150. LPTIM1 input capture 2 connection

| Iptim_ic2_mux | LPTIM1 input capture 2 connected to |
|----------------|-------------------------------------|
| Iptim_ic2_mux0 | I/O |
| Iptim_ic2_mux1 | LSI |
| Iptim_ic2_mux2 | LSE |
| Iptim_ic2_mux3 | Reserved |

Table 151. LPTIM2 input capture 2 connection

| Iptim_ic2_mux | LPTIM2 input capture 2 connected to |
|----------------|-------------------------------------|
| Iptim_ic2_mux0 | I/O |
| Iptim_ic2_mux1 | MCG256/1024 |
| Iptim_ic2_mux2 | MCG3072/128 |
| Iptim_ic2_mux3 | I3C2 IBI request (_IBIAck) |

Table 152. LPTIM3 input capture 2 connection

| Iptim_ic2_mux | LPTIM3 input capture 2 connected to |
|----------------|-------------------------------------|
| Iptim_ic2_mux0 | I/O |
| Iptim_ic2_mux1 | Reserved |
| Iptim_ic2_mux2 | Reserved |
| Iptim_ic2_mux3 | I3C1 IBI request (_IBIAck) |

27.4.4 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 uses 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 must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

27.4.5 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 must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

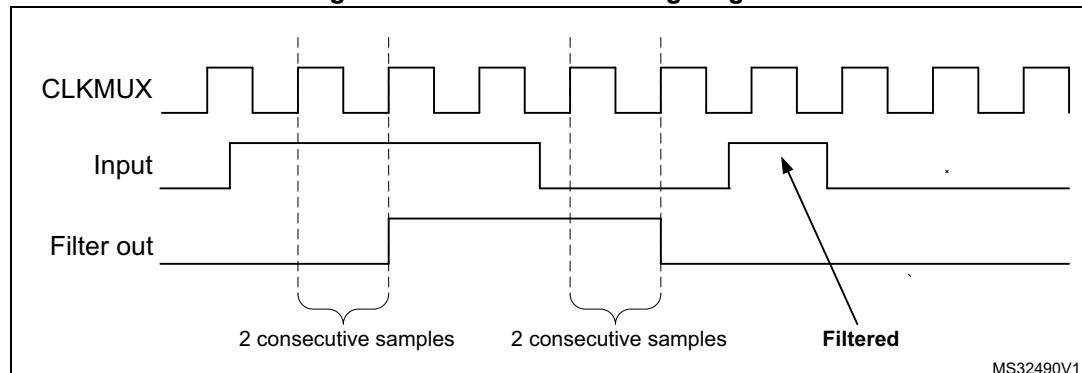
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF 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 is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. [Figure 277](#) shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

Figure 277. Glitch filter timing diagram



Note: *In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to '0'. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.*

27.4.6 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 153. Prescaler division ratios

| programming | dividing factor |
|-------------|-----------------|
| 000 | /1 |
| 001 | /2 |

Table 153. Prescaler division ratios (continued)

| programming | dividing factor |
|-------------|-----------------|
| 010 | /4 |
| 011 | /8 |
| 100 | /16 |
| 101 | /32 |
| 110 | /64 |
| 111 | /128 |

27.4.7 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 is 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 is 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.*

27.4.8 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 an LPTIM update event is generated.

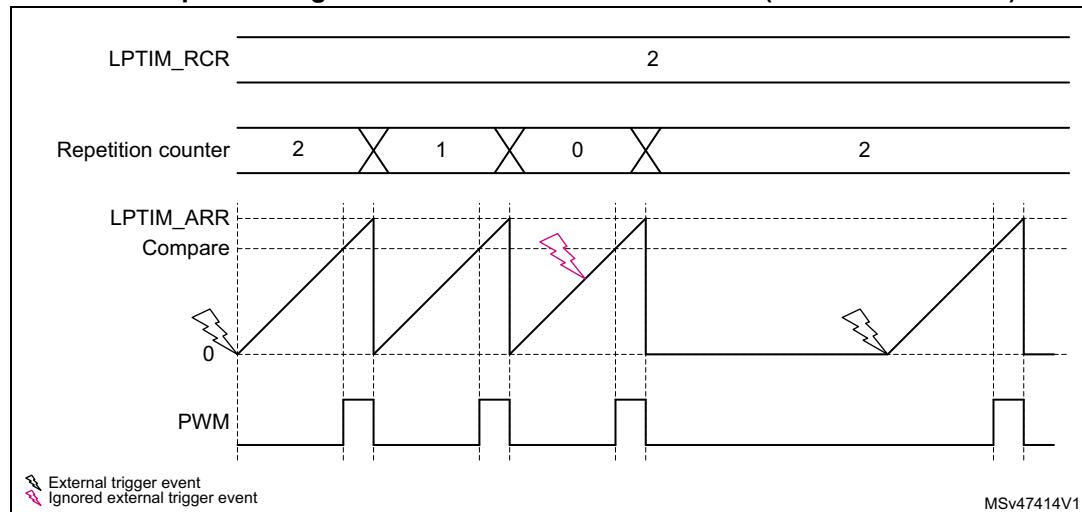
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in [Figure 278](#).

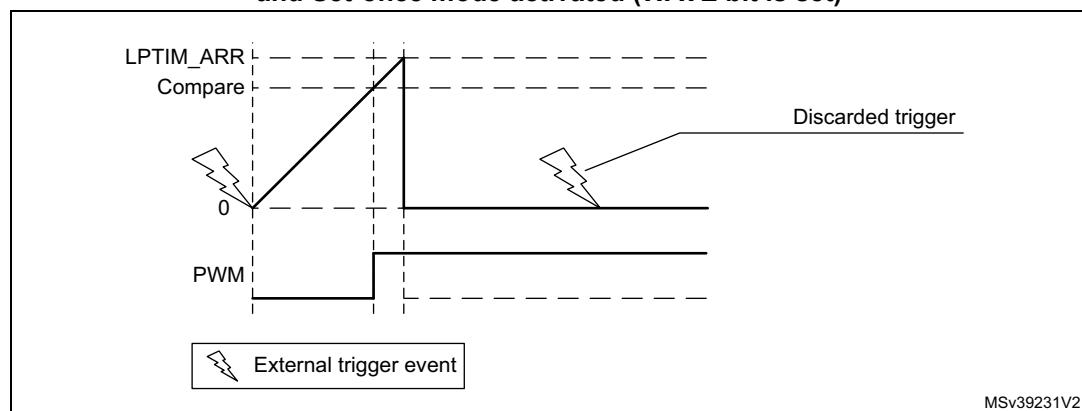
Figure 278. LPTIM output waveform, single counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note that when the WAVE bitfield 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 279](#).

Figure 279. 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 starts the counter for one-shot counting.

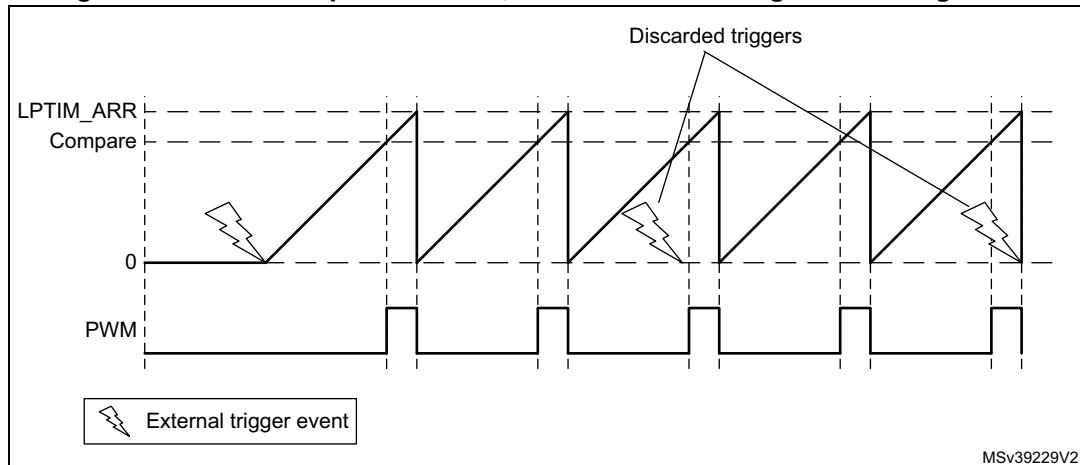
Continuous mode

To enable the continuous counting, the CNTSTART bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTART is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 280*.

In case of software start (TRIGEN[1:0] = '00'), setting CNTSTART starts the counter for continuous counting.

Figure 280. LPTIM output waveform, Continuous counting mode configuration



MSv39229V2

SNGSTART and CNTSTART 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 SNGSTART switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTART switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

27.4.9 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 starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

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.

27.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/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_CCRx. The LPTIM output is reset as soon as a

match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see [Section 27.4.19: PWM mode](#).

- 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_CCRx register value.

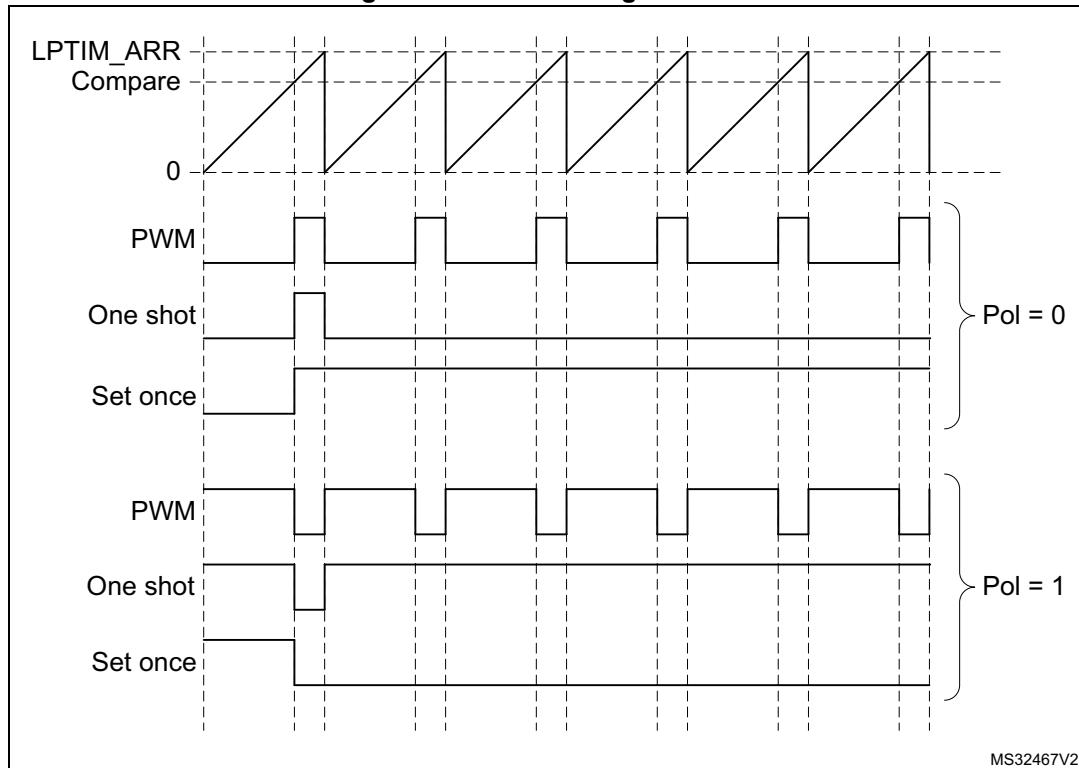
The LPTIM output waveform can be configured through the WAVE bit as follow:

- 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 CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes 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 281](#) 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 CCxP bit.

Figure 281. Waveform generation



27.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to '0', the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to '1', the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, 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, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx 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, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

27.4.12 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 is 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 must 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.

27.4.13 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 register must be modified only when the LPTIM is disabled.

27.4.14 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 bitfield 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 elapse before the reset is taken into account. This makes 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 resets its content to zero. Asynchronous reset must 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 must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.

Note 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 must make sure that these two mechanisms are used exclusively.

27.4.15 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.

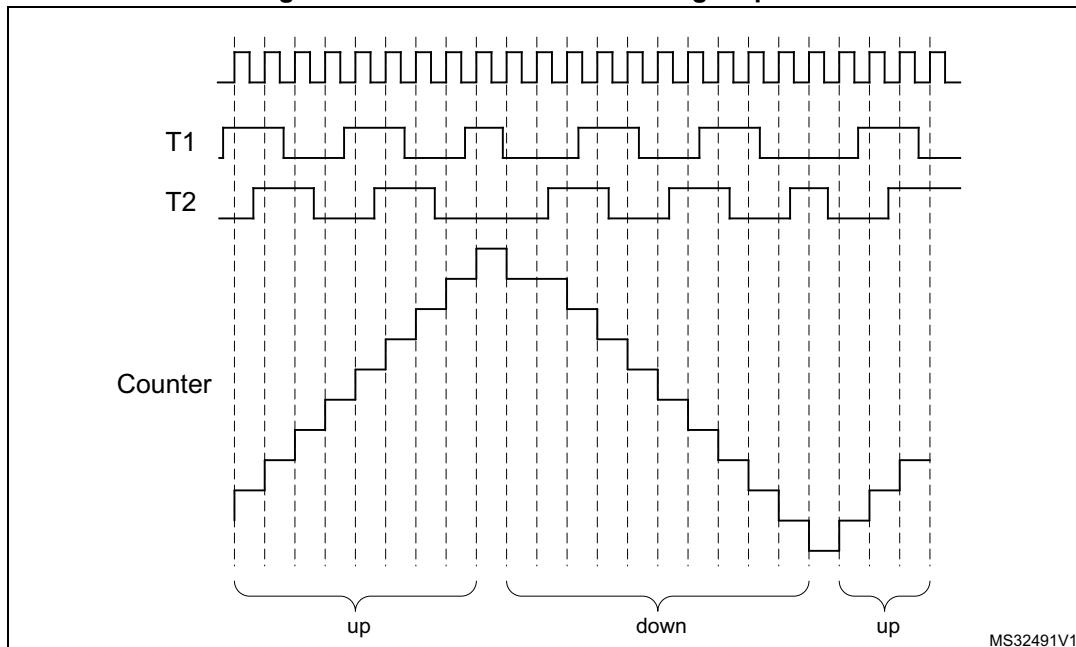
Table 154. Encoder counting scenarios

| 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 282. Encoder mode counting sequence



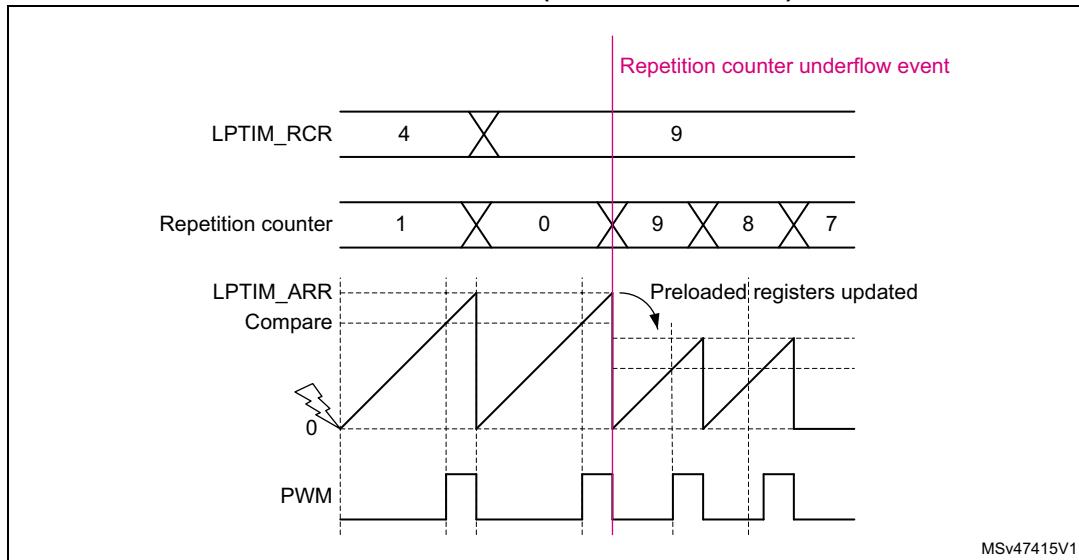
27.4.16 Repetition Counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in [Figure 283](#).

Figure 283. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



MSv47415V1

A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to section "Register update" for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, that new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain so that the new written value is loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the REPOK flag is set, leads to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

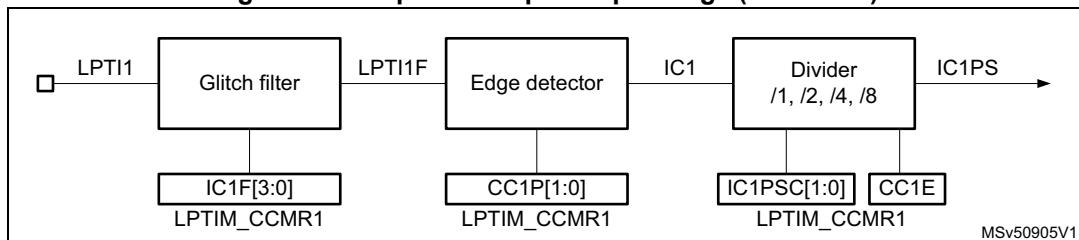
27.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIx F. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

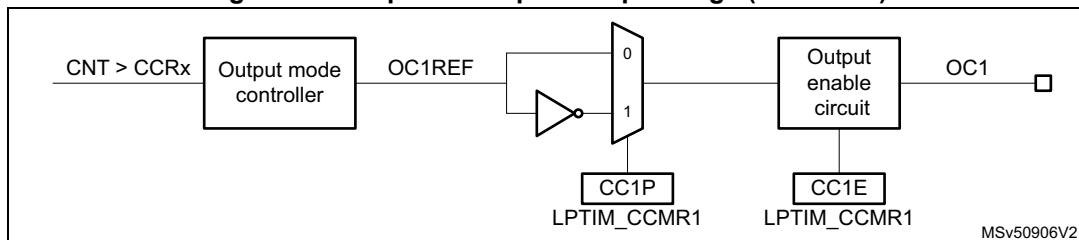
Figure 284. Capture/compare input stage (channel 1)



Output 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 285. Capture/compare output stage (channel 1)



27.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR 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 (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note:

In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the IC DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value

(PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see [Table 155](#)) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

$$\text{Real capture value} = \text{captured(LPTIM_CCR}_x\text{)} - \text{offset}$$

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the [Table 155](#), the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to $9 - 5 = 4$.

Table 155. Input capture Glitch filter latency (in counter step unit)

| Prescaler PRESC[2:0] | ICxF[1:0] | Offset |
|----------------------|-----------|--------|
| 0 | 0 | 2 |
| | 1 | 7 |
| | 2 | 9 |
| | 3 | 13 |
| 1 | 0 | 3 |
| | 1 | 5 |
| | 2 | 6 |
| | 3 | 8 |
| 2 | 0 | 2 |
| | 1 | 3 |
| | 2 | 4 |
| | 3 | 5 |
| 3 | 0 | 2 |
| | 1 | 2 |
| | 2 | 3 |
| | 3 | 3 |
| 4 | 0 | 2 |
| | 1 | 2 |
| | 2 | 2 |
| | 3 | 2 |

Table 155. Input capture Glitch filter latency (in counter step unit) (continued)

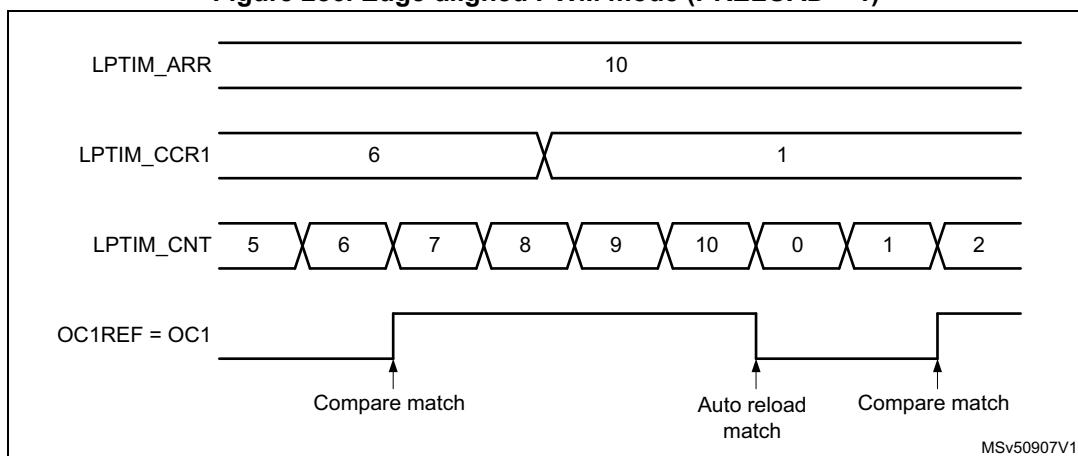
| Prescaler PRESC[2:0] | ICxF[1:0] | Offset |
|----------------------|-----------|--------|
| 5 | 0 | 2 |
| | 1 | 2 |
| | 2 | 2 |
| | 3 | 2 |
| 6 | 0 | 2 |
| | 1 | 2 |
| | 2 | 2 |
| | 3 | 2 |
| 7 | 0 | 2 |
| | 1 | 2 |
| | 2 | 2 |
| | 3 | 2 |

27.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

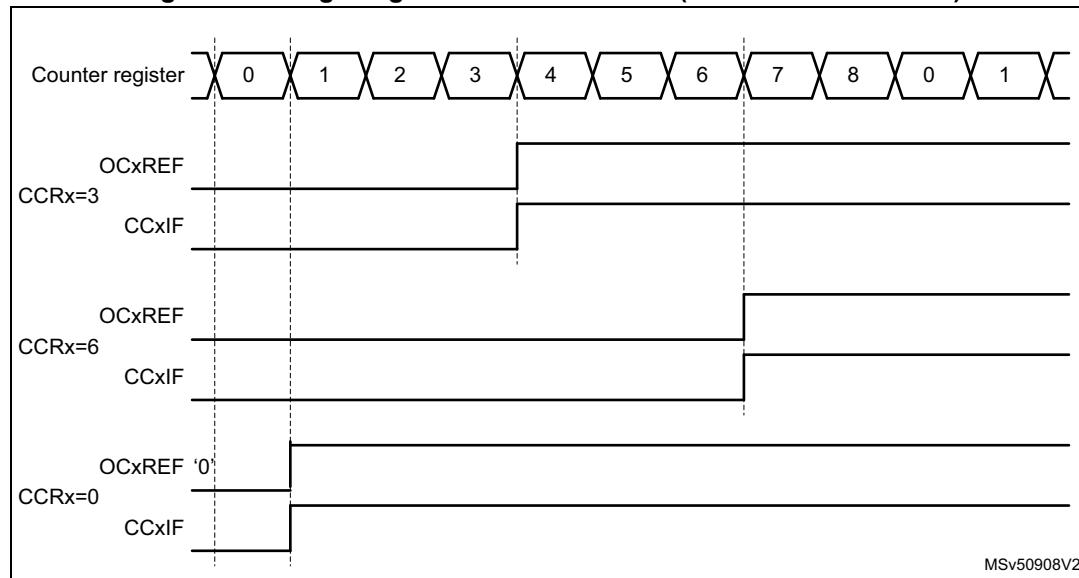
Figure 286 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

Figure 286. Edge-aligned PWM mode (PRELOAD = 1)

In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT \leq LPTIM_CCRx else it becomes high.

[Figure 287](#) shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 287. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



MSv50908V2

PWM mode with immediate update PRELOAD = 0

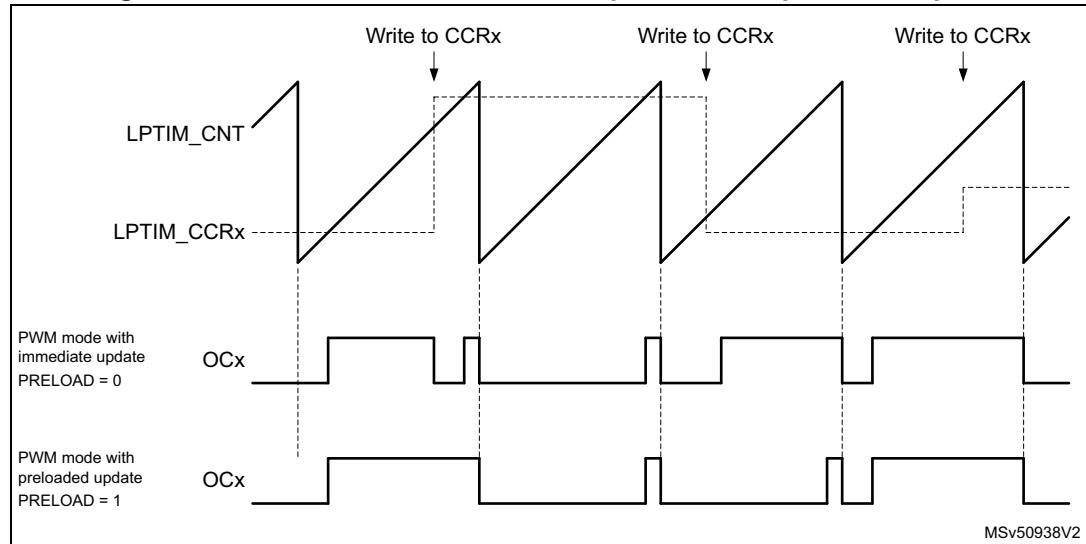
The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. [Figure 288](#) illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 288. PWM mode with immediate update versus preloaded update

Note: For both PWM modes, the compare match, auto-reload match and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. [Figure 286](#) illustrates this behavior.

27.4.20 DMA requests

The LPTIM has the capability to generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests are used to re-program part of the LPTIMER, multiple times, at regular intervals, without software overhead.

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the CCRx register. The captured values in CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)
- or if the channel x is disabled (clear CCxE bit)
- or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note:

The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

27.4.21 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

27.5 LPTIM low-power modes

Table 156. 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. |
| Standby | The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode. |

Note:

All DMA requests must be disabled (reset UEDE and CCxDE bits) before entering Sleep, Stop and Standby modes.

27.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER 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).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.

Table 157. Interrupt events

| Interrupt vector | Interrupt event | Event flag | Enable control bit | Interrupt clear method | Exit from Sleep mode | Exit from Stop mode ⁽¹⁾ |
|------------------|---|------------|--------------------|------------------------|----------------------|------------------------------------|
| LPTIMx | Compare match | CCxIF | CCxIE | Write 1 to CCxCF | Yes | Yes |
| | Input capture | CCxIF | CCxIE | Write 1 to CCxCF | Yes | Yes |
| | Over-capture | CCxOF | CCxOIE | Write 1 to CCxOCF | Yes | Yes |
| | Auto-reload match | ARRM | ARRMIE | Write 1 to ARRMCF | Yes | Yes |
| | External trigger event | EXTTRIG | EXTTRIGIE | Write 1 to EXTTRIGCF | Yes | Yes |
| | Auto-reload register update OK | ARROK | ARROKIE | Write 1 to ARROKCF | Yes | Yes |
| | Capture/compare register update OK | CMPxOK | CMPxOKIE | Write 1 to CMPxOKCF | Yes | Yes |
| | Direction change to up ⁽²⁾ | UP | UPIE | Write 1 to UPCF | Yes | Yes |
| | Direction change to down ⁽²⁾ | DOWN | DOWNIE | Write 1 to DOWNCF | Yes | Yes |
| | Update Event | UE | UEIE | Write 1 to UECF | Yes | Yes |
| | Repetition register update OK | REPOK | REPOKIE | Write 1 to REPOKCF | Yes | Yes |

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to [Section 27.3: LPTIM implementation](#).
2. If LPTIM does not support encoder mode feature, this event does not exist. Refer to [Section 27.3: LPTIM implementation](#).

27.7 LPTIM registers

Refer to [Section 1.2: List of abbreviations for registers on page 51](#) for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

27.7.1 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture mode.

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. | DIER OK | Res. | Res. | CMP4 OK | CMP3 OK | CMP2 OK | 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. | CC4IF | CC3IF | CC2IF | REP OK | UE | DOWN | UP | ARR OK | CMP1 OK | EXT TRIG | ARRM | CC1IF |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

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

Bit 24 **DIEROK**: Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

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

Bit 21 **CMP4OK**: Compare register 4 update OK

CMP4OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR4 register has been successfully completed. CMP4OK flag can be cleared by writing 1 to the CMP4OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 20 **CMP3OK**: Compare register 3 update OK

CMP3OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR3 register has been successfully completed. CMP3OK flag can be cleared by writing 1 to the CMP3OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 **CC4IF**: Compare 4 interrupt flag

If channel CC4 is configured as output:

The CC4IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC4IF flag can be cleared by writing 1 to the CC4CF bit in the LPTIM_ICR register.

0:No match

1:The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR4 register's value

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 10 **CC3IF:** Compare 3 interrupt flag

If channel CC3 is configured as output:

The CC3IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC3IF flag can be cleared by writing 1 to the CC3CF bit in the LPTIM_ICR register.

0:No match

1:The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR3 register's value.

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to [Section 27.3](#).

Bit 9 **CC2IF:** Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to [Section 27.3](#).

Bit 8 **REPOK:** Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE:** LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

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. Refer to [Section 27.3](#).

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. Refer to [Section 27.3](#).

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 **CMP1OK:** Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF 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 **CC1IF**: Compare 1 interrupt flag**If channel CC1 is configured as output:**

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

27.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) ($x = 1$ to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

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. | DIER OK | 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 |
| CC4 OF | CC3 OF | CC2 OF | CC1 OF | CC4IF | CC3IF | CC2IF | REP OK | UE | DOWN | UP | ARR OK | Res. | EXT TRIG | ARRM | CC1IF |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

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

Bit 24 **DIEROK**: Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **CC4OF**: Capture 4 over-capture 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 1 to the CC4OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR4 register while CC4IF flag was already set

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 14 **CC3OF**: Capture 3 over-capture 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 1 to the CC3OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR3 register while CC3IF flag was already set

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 13 **CC2OF**: Capture 2 over-capture 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 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bit 12 **CC1OF**: Capture 1 over-capture 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 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 27.3.

Bit 11 **CC4IF**: Capture 4 interrupt flag**If channel CC4 is configured as input:**

CC4IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR4 register. The corresponding interrupt or DMA request is generated if enabled. The CC4OF flag is set if the CC4IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR4 register. (An edge has been detected on IC4 which matches the selected polarity). The CC4IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC4IF flag can be cleared by writing 1 to the CC4CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 10 **CC3IF**: Capture 3 interrupt flag**If channel CC3 is configured as input:**

CC3IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR3 register. The corresponding interrupt or DMA request is generated if enabled. The CC3OF flag is set if the CC3IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR3 register. (An edge has been detected on IC3 which matches the selected polarity). The CC3IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC3IF flag can be cleared by writing 1 to the CC3CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to [Section 27.3](#).

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

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. Refer to [Section 27.3](#).

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. Refer to [Section 27.3](#).

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 Reserved, must be kept at reset value.

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 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0:No input capture occurred

1:The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

27.7.3 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

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. | DIER OKCF | Res. | Res. | CMP4 OKCF | CMP3 OKCF | CMP2 OKCF | Res. | Res. | Res. |
| | | | | | | | w | | | w | w | w | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | CC4CF | CC3CF | CC2CF | REPOK CF | UECF | DOWN CF | UPCF | ARR OKCF | CMP1 OKCF | EXT TRIG CF | ARRM CF | CC1CF |
| | | | | w | w | w | w | w | w | w | w | w | w | w | w |

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

Bit 24 **DIEROKCF**: Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

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

Bit 21 **CMP4OKCF**: Compare register 4 update OK clear flag

Writing 1 to this bit clears the CMP4OK flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 20 **CMP3OKCF**: Compare register 3 update OK clear flag

Writing 1 to this bit clears the CMP3OK flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag

Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 **CC4CF**: Capture/compare 4 clear flag

Writing 1 to this bit clears the CC4IF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 10 **CC3CF**: Capture/compare 3 clear flag

Writing 1 to this bit clears the CC3IF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 9 **CC2CF**: Capture/compare 2 clear flag

Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

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. Refer to Section 27.3.

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. Refer to Section 27.3.

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 **CMP1OKCF**: Compare register 1 update OK clear flag

Writing 1 to this bit clears the CMP1OK 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 **CC1CF**: Capture/compare 1 clear flag

Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

27.7.4 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

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. | DIER OKCF | 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 |
| CC4 OCF | CC3 OCF | CC2 OCF | CC1 OCF | CC4CF | CC3CF | CC2CF | REPOK CF | UECF | DOWN CF | UPCF | ARRO KCF | Res. | EXTTR IGCF | ARRM CF | CC1CF |
| w | w | w | w | w | w | w | w | w | w | w | w | | w | w | w |

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

Bit 24 **DIEROKCF**: Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **CC4OCF**: Capture/compare 4 over-capture clear flag

Writing 1 to this bit clears the CC4OF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 14 **CC3OCF**: Capture/compare 3 over-capture clear flag

Writing 1 to this bit clears the CC3OF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag

Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag

Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 27.3.

Bit 11 **CC4CF**: Capture/compare 4 clear flag

Writing 1 to this bit clears the CC4IF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 10 **CC3CF**: Capture/compare 3 clear flag

Writing 1 to this bit clears the CC3IF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 9 **CC2CF**: Capture/compare 2 clear flag

Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

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. Refer to [Section 27.3](#).

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. Refer to [Section 27.3](#).

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 Reserved, must be kept at reset value.

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 **CC1CF**: Capture/compare 1 clear flag

Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

27.7.5 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) ($x = 1$ to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

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. | UEDE | Res. | CMP4 OKIE | CMP3 OKIE | CMP2 OKIE | Res. | Res. | Res. |
| | | | | | | | | rw | | rw | rw | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | CC4IE | CC3IE | CC2IE | REPOK IE | UEIE | DOWNI E | UPIE | ARRO KIE | CMP1 OKIE | EXT TRIGIE | ARRM IE | CC1IE |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bit 23 **UEDE**: Update event DMA request enable

0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to [Section 27.3](#).

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CMP4OKIE**: Compare register 4 update OK interrupt enable

0: CMPOK register 4 interrupt disabled
1: CMPOK register 4 interrupt enabled

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to [Section 27.3](#).

Bit 20 **CMP3OKIE**: Compare register 3 update OK interrupt enable

- 0: CMPOK register 3 interrupt disabled
- 1: CMPOK register 3 interrupt enabled

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable

- 0: CMPOK register 2 interrupt disabled
- 1: CMPOK register 2 interrupt enabled

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 **CC4IE**: Capture/compare 4 interrupt enable

- 0: Capture/compare 4 interrupt disabled
- 1: Capture/compare 4 interrupt enabled

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 10 **CC3IE**: Capture/compare 3 interrupt enable

- 0: Capture/compare 3 interrupt disabled
- 1: Capture/compare 3 interrupt enabled

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable

- 0: Capture/compare 2 interrupt disabled
- 1: Capture/compare 2 interrupt enabled

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable

- 0: Repetition register update OK interrupt disabled
- 1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable

- 0: Update event interrupt disabled
- 1: Update event interrupt enabled

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. Refer to Section 27.3.

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. Refer to Section 27.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable

- 0: ARROK interrupt disabled
- 1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable

- 0: CMPOK register 1 interrupt disabled
- 1: CMPOK register 1 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 **CC1IE**: Capture/compare 1 interrupt enable

- 0: Capture/compare 1 interrupt disabled
- 1: Capture/compare 1 interrupt enabled

27.7.6 LPTIM_x interrupt enable register [alternate] (LPTIM_x_DIER) ($x = 1$ to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

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. | CC4DE | CC3DE | CC2DE | Res. | UEDE | Res. | Res. | Res. | Res. | Res. | Res. | CC1DE |
| | | | | rw | rw | rw | | rw | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CC4OI E | CC3OI E | CC2OI E | CC1OI E | CC4IE | CC3IE | CC2IE | REPOK IE | UEIE | DOWNI E | UPIE | ARRO KIE | Res. | EXT TRIGIE | ARRM IE | CC1IE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw |

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

Bit 27 **CC4DE**: Capture/compare 4 DMA request enable

- 0: CC4 DMA request disabled. Writing '0' to the CC4DE bit resets the associated ic4_dma_req signal.
- 1: CC4 DMA request enabled

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 26 **CC3DE**: Capture/compare 3 DMA request enable

- 0: CC3 DMA request disabled. Writing '0' to the CC3DE bit resets the associated ic3_dma_req signal.
- 1: CC3 DMA request enabled

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable

- 0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
- 1: CC2 DMA request enabled

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable

- 0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
- 1: UE DMA request enabled

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 27.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable

- 0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
- 1: CC1 DMA request enabled

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 27.3.

Bit 15 **CC4OIE**: Capture/compare 4 over-capture interrupt enable

- 0: CC4 over-capture interrupt disabled
- 1: CC4 over-capture interrupt enabled

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 14 **CC3OIE**: Capture/compare 3 over-capture interrupt enable

- 0: CC3 over-capture interrupt disabled
- 1: CC3 over-capture interrupt enabled

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable

- 0: CC2 over-capture interrupt disabled
- 1: CC2 over-capture interrupt enabled

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable

- 0: CC1 over-capture interrupt disabled
- 1: CC1 over-capture interrupt enabled

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 27.3.

Bit 11 **CC4IE**: Capture/compare 4 interrupt enable

- 0: Capture/compare 4 interrupt disabled
- 1: Capture/compare 4 interrupt enabled

Note: If LPTIM does not implement at least 4 channels this bit is reserved. Refer to Section 27.3.

Bit 10 **CC3IE**: Capture/compare 3 interrupt enable

- 0: Capture/compare 3 interrupt disabled
- 1: Capture/compare 3 interrupt enabled

Note: If LPTIM does not implement at least 3 channels this bit is reserved. Refer to Section 27.3.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable

- 0: Capture/compare 2 interrupt disabled
- 1: Capture/compare 2 interrupt enabled

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 27.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable

- 0: Repetition register update OK interrupt disabled
- 1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable

- 0: Update event interrupt disabled
- 1: Update event interrupt enabled

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. Refer to [Section 27.3](#).

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. Refer to [Section 27.3](#).

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable

- 0: ARROK interrupt disabled
- 1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

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 **CC1IE**: Capture/compare 1 interrupt enable

- 0: Capture/compare 1 interrupt disabled
- 1: Capture/compare 1 interrupt enabled

27.7.7 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 | PRE LOAD | Res. | WAVE | TIMOUT | TRIGEN[1:0] | Res. | |
| | | | | | | | 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. Refer to [Section 27.3](#).

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, LPTIM_RCR and the LPTIM_CCRx 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 Reserved, must be kept at reset value.

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 is ignored
- 1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition 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 serves 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 27.4.3: LPTIM input and 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 are 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 are 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

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 27.4.15: Encoder mode](#) for more details about Encoder mode sub-modes.

Bit 0 CKSEL: Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

- 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_CFG register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0').

27.7.8 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 asynchronously resets 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 triggers 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 must 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 does 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 is 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 stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 ENABLE: LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

27.7.9 LPTIM compare register 1 (LPTIM_CCR1)

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. |
| | | | | | | | | | | | | | | | |
| CCR1[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 **CCR1[15:0]:** Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PRELOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

27.7.10 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. |
| | | | | | | | | | | | | | | | |
| 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 CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

27.7.11 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. |
| | | | | | | | | | | | | | | | |
| 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, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform consecutive reads until two returned values are identical.

27.7.12 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

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. | IC2SEL[1:0] | Res. | Res. | IC1SEL[1:0] | | |
| | | | | | | | | | | rw | rw | | | rw | rw |
| IC2SEL[1:0] | | | | | | | | | | | | | | | |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | rw | rw | | | rw | rw |

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection

The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.

- 00: lptim_ic2_mux0
- 01: lptim_ic2_mux1
- 10: lptim_ic2_mux2
- 11: lptim_ic2_mux3

For connection details refer to [Section 27.4.3: LPTIM input and trigger mapping](#).

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection

The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.

- 00: lptim_ic1_mux0
- 01: lptim_ic1_mux1
- 10: lptim_ic1_mux2
- 11: lptim_ic1_mux3

For connection details refer to [Section 27.4.3: LPTIM input and trigger mapping](#).

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection

The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.

- 00: lptim_in2_mux0
- 01: lptim_in2_mux1
- 10: lptim_in2_mux2
- 11: lptim_in2_mux3

For connection details refer to [Section 27.4.3: LPTIM input and trigger mapping](#).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection

The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.

- 00: lptim_in1_mux0
- 01: lptim_in1_mux1
- 10: lptim_in1_mux2
- 11: lptim_in1_mux3

For connection details refer to [Section 27.4.3: LPTIM input and trigger mapping](#).

27.7.13 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

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

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

Bits 7:0 REP[7:0]: Repetition register value

REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to '1'). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

27.7.14 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the CCxSEL bits.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|-----------|----|------|------|-------------|----|------|------|------|------|-----------|----|------|---------|
| Res. | Res. | IC2F[1:0] | | Res. | Res. | IC2PSC[1:0] | | Res. | Res. | Res. | Res. | CC2P[1:0] | | CC2E | CC2 SEL |
| | | 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. | IC1F[1:0] | | Res. | Res. | IC1PSC[1:0] | | Res. | Res. | Res. | Res. | CC1P[1:0] | | CC1E | CC1 SEL |
| | | rw | rw | | | rw | rw | | | | | rw | rw | rw | rw |

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

Bits 29:28 IC2F[1:0]: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

- 00: any external input capture signal level change is considered as a valid transition
- 01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
- 10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
- 11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 IC2PSC[1:0]: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

- 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 23:20 Reserved, must be kept at reset value.

Bits 19:18 CC2P[1:0]: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

- 0: OC2 active high
- 1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

- 00: rising edge, circuit is sensitive to IC2 rising edge
- 01: falling edge, circuit is sensitive to IC2 falling edge
- 10: reserved, do not use this configuration.
- 11: both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 CC2E: Capture/compare 2 output enable.

Condition: CC2 as output

- 0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.
- 1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

- 0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.
- 1: Capture enabled.

Bit 16 CC2SEL: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

- 0: CC2 channel is configured in output PWM mode
- 1: CC2 channel is configured in input capture mode

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

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

- 00: any external input capture signal level change is considered as a valid transition
- 01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
- 10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
- 11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

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

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

- 0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers
- 1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

- 00: rising edge, circuit is sensitive to IC1 rising edge
- 01: falling edge, circuit is sensitive to IC1 falling edge
- 10: reserved, do not use this configuration.
- 11: both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

- 0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.
- 1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

- 0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.
- 1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

- 0: CC1 channel is configured in output PWM mode
- 1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3)

kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

27.7.15 LPTIM capture/compare mode register 2 (LPTIM_CCMR2)

Address offset: 0x030

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|-----------|----|------|------|-------------|----|------|------|------|------|-----------|----|------|---------|
| Res. | Res. | IC4F[1:0] | | Res. | Res. | IC4PSC[1:0] | | Res. | Res. | Res. | Res. | CC4P[1:0] | | CC4E | CC4 SEL |
| | | 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. | IC3F[1:0] | | Res. | Res. | IC3PSC[1:0] | | Res. | Res. | Res. | Res. | CC3P[1:0] | | CC3E | CC3 SEL |
| | | rw | rw | | | rw | rw | | | | | rw | rw | rw | rw |

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

Bits 29:28 **IC4F[1:0]: Input capture 4 filter**

This bitfield defines the number of consecutive equal samples that should be detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: any external input capture signal level change is considered as a valid transition

01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC4PSC[1:0]: Input capture 4 prescaler**

This bitfield defines the ratio of the prescaler acting on the CC4 input (IC4).

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 23:20 Reserved, must be kept at reset value.

Bits 19:18 **CC4P[1:0]**: Capture/compare 4 output polarity.

Condition: CC4 as output:

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC4 active high

1: OC4 active low

Condition: CC4 as input:

This field is used to select the IC4 polarity for capture operations.

00: rising edge, circuit is sensitive to IC4 rising edge

01: falling edge, circuit is sensitive to IC4 falling edge

10: reserved, do not use this configuration.

11: both edges, circuit is sensitive to both IC4 rising and falling edges.

Bit 17 **CC4E**: Capture/compare 4 output enable.

Condition: CC4 as output:

0: Off - OC4 is not active. Writing '0' to the CC4E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC4 signal is output on the corresponding output pin

Condition: CC4 as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 4 (LPTIM_CCR4) or not.

0: Capture disabled. Writing '0' to the CC4E bit resets the associated ic4_dma_req signal.

1: Capture enabled.

Bit 16 **CC4SEL**: Capture/compare 4 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC4 channel is configured in output PWM mode

1: CC4 channel is configured in input capture mode

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

Bits 13:12 **IC3F[1:0]**: Input capture 3 filter

This bitfield defines the number of consecutive equal samples that should be detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: any external input capture signal level change is considered as a valid transition

01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC3PSC[1:0]**: Input capture 3 prescaler

This bitfield defines the ratio of the prescaler acting on the CC3 input (IC3).

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

Bits 3:2 **CC3P[1:0]**: Capture/compare 3 output polarity.

Condition: CC3 as output:

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC3 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC3 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC3 as input:

This field is used to select the IC3 polarity for capture operations.

00: rising edge, circuit is sensitive to IC3 rising edge

01: falling edge, circuit is sensitive to IC3 falling edge

10: reserved, do not use this configuration.

11: both edges, circuit is sensitive to both IC3 rising and falling edges.

Bit 1 **CC3E**: Capture/compare 3 output enable.

Condition: CC3 as output:

0: Off - OC3 is not active. Writing '0' to the CC3E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC3 signal is output on the corresponding output pin

Condition: CC3 as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 3 (LPTIM_CCR3) or not.

0: Capture disabled. Writing '0' to the CC3E bit resets the associated ic3_dma_req signal.

1: Capture enabled.

Bit 0 **CC3SEL**: Capture/compare 3 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC3 channel is configured in output PWM mode

1: CC3 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3) kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

27.7.16 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034

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 |
| CCR2[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 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PRELOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to [Section 27.3: LPTIM implementation](#).

27.7.17 LPTIM compare register 3 (LPTIM_CCR3)

Address offset: 0x038

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 |
| CCR3[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 **CCR3[15:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the capture/compare 3 register.

Depending on the PRELOAD option, the CCR3 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PRELOAD bit is reset.

The capture/compare register 3 contains the value to be compared to the counter LPTIM_CNT and signaled on OC3 output.

If channel CC3 is configured as input:

CCR3 becomes read-only, it contains the counter value transferred by the last input capture 3 event. The LPTIM_CCR3 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR3 register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

Note: If the LPTIM implements less than 3 channels this register is reserved. Refer to [Section 27.3: LPTIM implementation](#).

27.7.18 LPTIM compare register 4 (LPTIM_CCR4)

Address offset: 0x03C

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. |
| | | | | | | | | | | | | | | | |
| CCR4[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 CCR4[15:0]: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the capture/compare 4 register.

Depending on the PRELOAD option, the CCR4 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PRELOAD bit is reset.

The capture/compare register 4 contains the value to be compared to the counter LPTIM_CNT and signaled on OC4 output.

If channel CC4 is configured as input:

CCR4 becomes read-only, it contains the counter value transferred by the last input capture 4 event. The LPTIM_CCR4 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR4 register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

Note: If the LPTIM implements less than 4 channels this register is reserved. Refer to [Section 27.3: LPTIM implementation](#).

27.7.19 LPTIM register map

The following table summarizes the LPTIM registers.

Table 158. 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 | LPTIMx_ISR (x = 1 to 3) Output compare mode | Res. | DIEROK | Res. | Res. | CMP4OK ² | 20 | CMP3OK ² | 19 | CMP2OK ³ | 18 | CC4IF | 17 | CC3IF | 16 | CC2IF ³ | 15 | CC1IF | 14 | CC3OF | 13 | CC2OF ³ | 12 | CC4OF | 11 | CC3OF | 10 | CC2OF ³ | 9 |
| | 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 | | | |
| | LPTIMx_ISR (x = 1 to 3) Input capture mode | Res. | DIEROK | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 158. LPTIM register map and reset values (continued)

Table 158. LPTIM 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 | | |
|--------|---------------------------|------|------|------|-----------|------|------|------|------|------|------|------|------|------|-----------|------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------------|------|------|
| 0x02C | LPTIM_CCMR1 | Res. | Res. | | IC2F[1:0] | | | | Res. | Res. | Res. | Res. | Res. | Res. | CC2P[1:0] | 0 | IC2PSC[1:0] | 0 | Res. | Res. | Res. |
| | Reset value | 0 | 0 | | IC4F[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 | | |
| 0x030 | LPTIM_CCMR2 | Res. | Res. | | IC4F[1:0] | | | | Res. | 0 | CC4P[1:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | |
| 0x034 | LPTIM_CCR2 ⁽⁵⁾ | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0x03C | LPTIM_CCR4 ⁽⁶⁾ | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

1. If LPTIM does not implement at least 4 channels this bit is reserved. Refer to [Section 27.3: LPTIM implementation](#).
2. If LPTIM does not implement at least 3 channels this bit is reserved. Refer to [Section 27.3: LPTIM implementation](#).
3. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to [Section 27.3: LPTIM implementation](#).
4. If LPTIM does not support encoder mode feature, this bit is reserved. Refer to [Section 27.3: LPTIM implementation](#).
5. If the LPTIM implements less than 2 channels this register is reserved. Refer to [Section 27.3: LPTIM implementation](#).
6. If the LPTIM implements less than 4 channels this register is reserved. Refer to [Section 27.3: LPTIM implementation](#).

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

28 Independent watchdog (IWDG)

28.1 Introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to [Section 28.3](#) to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

28.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
 - In case of timeout
 - In case of refresh outside the expected window

28.3 IWDG implementation

Table 159. IWDG features ⁽¹⁾

| IWDG modes/features | IWDG |
|--|------|
| LSI used as IWDG kernel clock (iwdg_ker_ck) | X |
| Window function | X |
| Early wake-up interrupt generation | X |
| System reset generation ⁽²⁾ | X |
| Capability to work in system Stop | X |
| Capability to work in system Standby | X |
| Capability to generate an interrupt in system Stop ⁽³⁾ | X |
| Capability to generate an interrupt in system Standby | - |
| Capability to be frozen when the microcontroller enters in Debug mode ⁽⁴⁾ | X |
| Option bytes to control the activity in Stop mode ⁽⁵⁾ | X |
| Option bytes to control the activity in Standby mode ⁽⁶⁾ | X |
| Option bytes to control the Hardware mode ⁽⁷⁾ | X |

1. 'X' = supported, '-' = not supported.

2. Refer to the RCC section for additional information.

3. Wake-up from Stop with interrupt is supported only in Stop 0, Stop 1, and Stop 2 modes.

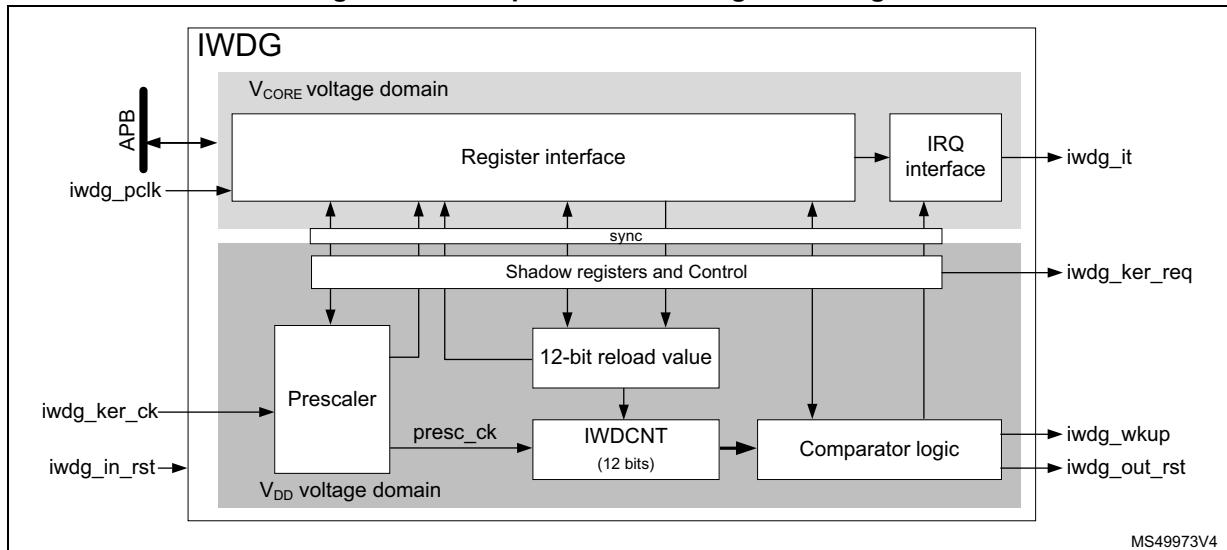
4. Controlled via DBG_IWDG_STOP in DBG section.
5. Controlled via the option byte IWDG_STOP in FLASH section.
6. Controlled via the option byte IWDG_STDBY in FLASH section.
7. Controlled via the option byte IWDG_SW in FLASH section.

28.4 IWDG functional description

28.4.1 IWDG block diagram

Figure 289 shows the functional blocks of the independent watchdog module.

Figure 289. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See [Section 28.3](#) for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

28.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in [Table 160](#).

Table 160. IWDG internal input/output signals

| Signal name | Signal type | Description |
|--------------|-------------|------------------------------|
| iwdg_ker_ck | Input | IWDG kernel clock |
| iwdg_ker_req | Input | IWDG kernel clock request |
| iwdg_pclk | Input | IWDG APB clock |
| iwdg_out_RST | Output | IWDG reset output |
| iwdg_in_RST | Input | IWDG reset input |
| iwdg_wkup | Output | IWDG wake-up event |
| iwdg_it | Output | IWDG early wake-up interrupt |

28.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The Software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the [IWDG key register \(IWDG_KR\)](#), and the IWDCNT starts counting down from the reset value (0xFFFF).

In the hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_RST). The IWDCNT down-counter starts counting down from the reset value 0xFFFF. The hardware watchdog mode feature is enabled through the device option bits, see [Section 28.3](#) for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_RST asserted).

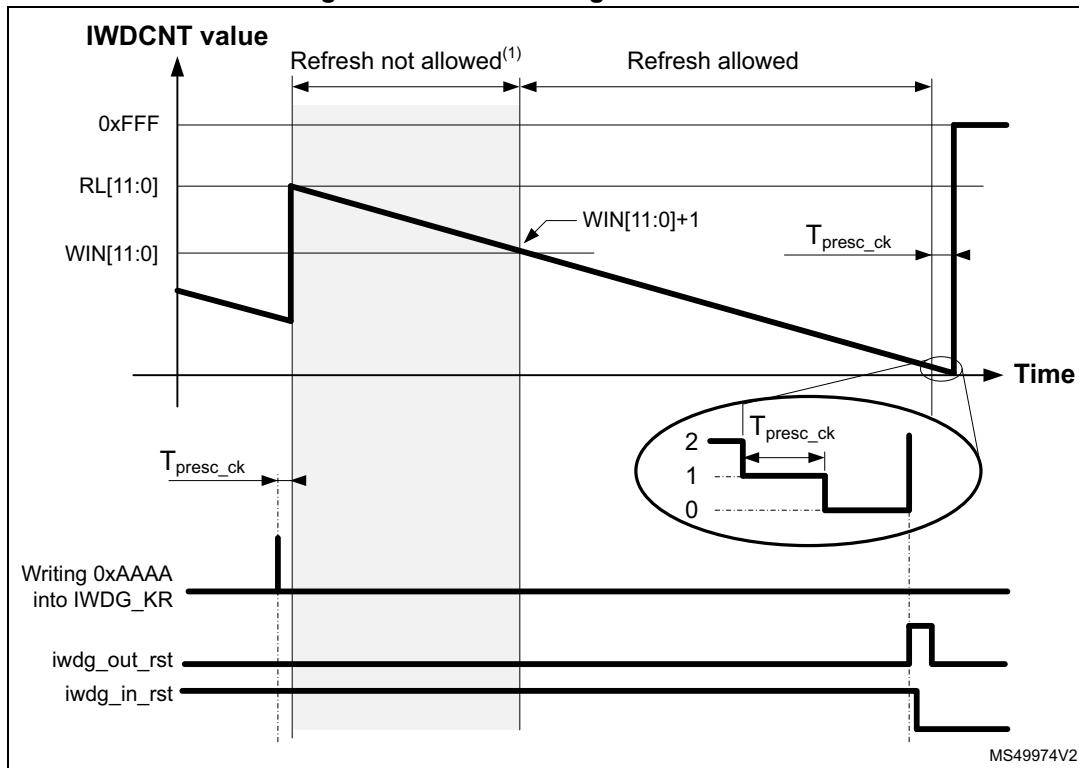
Whenever the key value 0x0000 AAAA is written in the [IWDG key register \(IWDG_KR\)](#), the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_RST asserted).

As shown in [Figure 290](#), when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 290. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCT reaches 1, the IWDG generates a reset (iwdg_out_RST is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_RST) to clear the reset source.

28.4.4 Window option

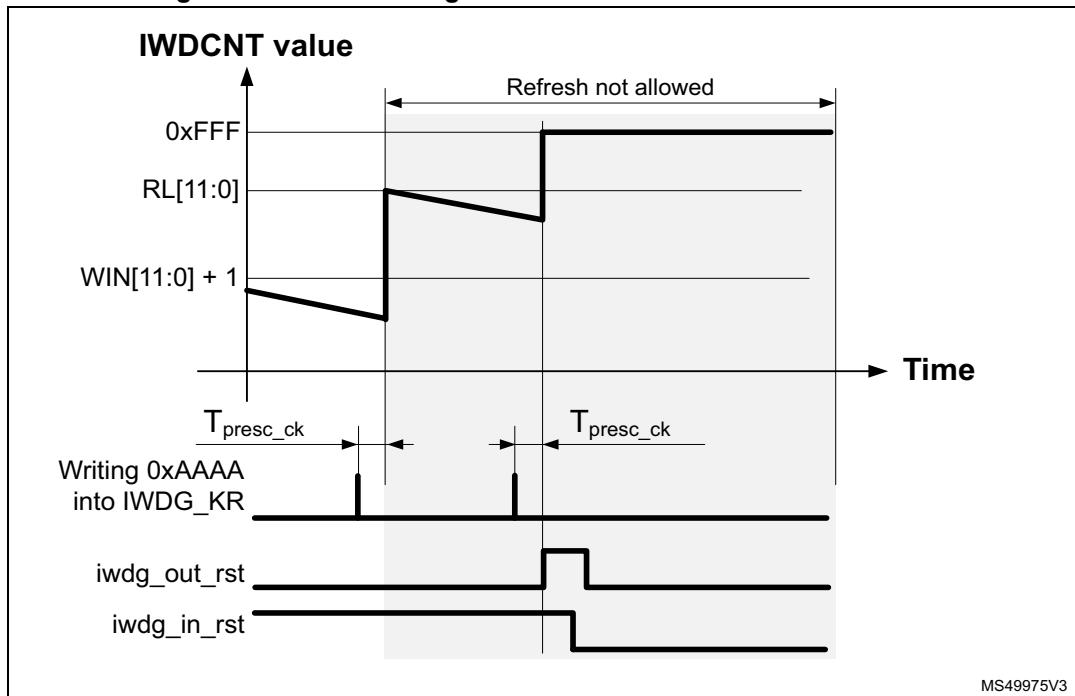
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than $WIN[11:0] + 1$, a reset is generated. $WIN[11:0]$ is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 291*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 291. Reset timing due to refresh in the not allowed area



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)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

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)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

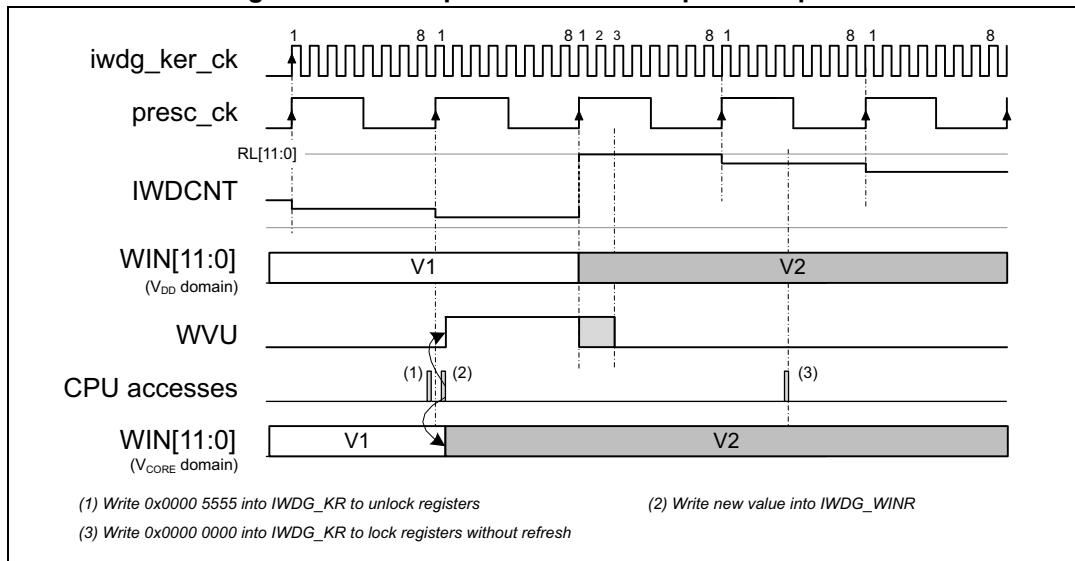
Updating the window comparator

It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step 3 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 292 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high. The new window value and the reload of IWDCNT with RL[11:0] are effective on the next rising edge of presc_ck. The WVU flag goes back to 0, in the worst case, two kernel clock periods later. So WVU remains high at most one period of presc_ck, plus two periods of the kernel clock.

Figure 292. Example of window comparator update

28.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to [Section 28.3](#) for details on the capabilities of this product.

28.4.6 Register access protection

Write accesses to [*IWDG prescaler register \(IWDG_PR\)*](#), [*IWDG reload register \(IWDG_RLR\)*](#), [*IWDG early wake-up interrupt register \(IWDG_EWCR\)*](#) and [*IWDG window register \(IWDG_WINR\)*](#) are protected. To modify them, first write 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 update of the prescaler or the down-counter reload value or the window value is ongoing.

28.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to [Section 28.3](#) for details.

Table 161. Effect of low power modes on IWDG

| Mode | Description |
|---------|--|
| Sleep | No effect. IWDG interrupts cause the device to exit from the mode. |
| Stop | The IWDG remains active or not, depending on option bytes configuration. Refer to Section 28.3 for details. IWDG interrupts cause the device exit the Stop mode. |
| Standby | The IWDG remains active or not, depending on option bytes configuration. Refer to Section 28.3 for details. IWDG interrupts do not make the device to exit from Standby mode. |

28.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the [*IWDG early wake-up interrupt register \(IWDG_EWCR\)*](#) to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

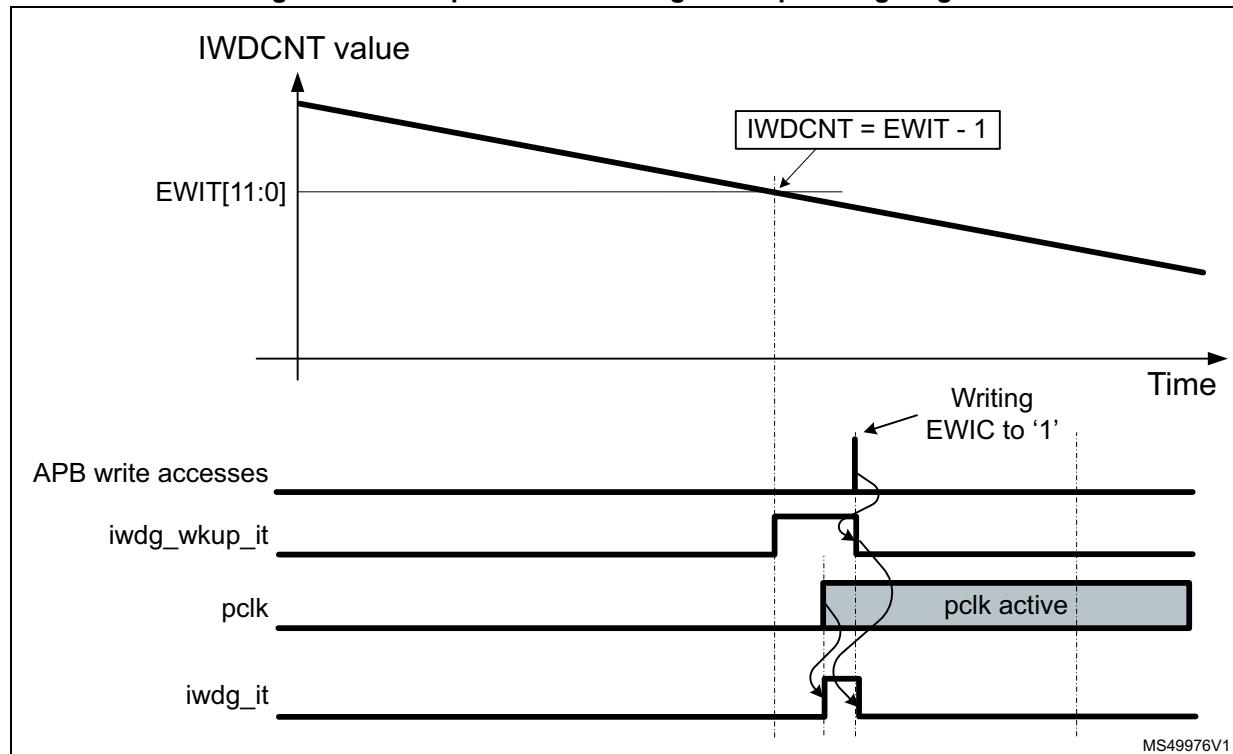
When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the [*IWDG status register \(IWDG_SR\)*](#) is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the [*IWDG early wake-up interrupt register \(IWDG_EWCR\)*](#).

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 293. Independent watchdog interrupt timing diagram

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the [IWDG key register \(IWDG_KR\)](#).
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the [IWDG early wake-up interrupt register \(IWDG_EWCR\)](#).
3. Wait for EWU = 0, EWU is located into the [IWDG status register \(IWDG_SR\)](#).
4. Write-protect registers by writing 0x0000 0000 to [IWDG key register \(IWDG_KR\)](#).

Step 3 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

[Figure 293](#) shows this sequence. As soon as the IWDG_EWCR register is written, the EWU flag goes high. The new comparator value and the reload of IWDCNT with RL[11:0] are effective on the next rising edge of presc_ck. The EWU flag goes back to 0, in the worst case, two kernel clock periods later. So, EWU remains high at most one period of presc_ck, plus two periods of the kernel clock.

Figure 294. Example of early wake-up comparator update

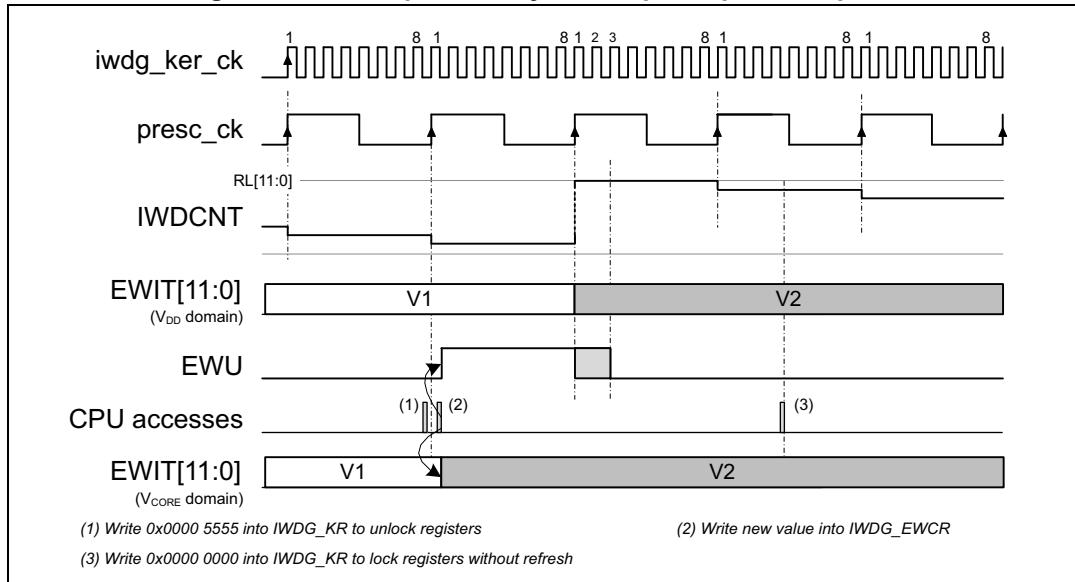


Table 162 summarizes the IWDG interrupt request.

Table 162. IWDG interrupt request

| Interrupt event | Event flag | Interrupt clear method | Interrupt enable control bit | Activated interrupt | |
|---------------------------|------------|------------------------|------------------------------|---------------------|------------------|
| | | | | iwdg_it | iwdg_wkup_it |
| IWDCNT reaches EWIT value | EWIF | Writing EWIC to 1 | EWIE | Y ⁽¹⁾ | Y ⁽²⁾ |

1. Generated when a clock is present on iwdg_pclk input.

2. Generated when a clock is present on iwdg_ker_ck input.

28.7 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).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_RST is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

28.7.1 IWDG key register (IWDG_KR)

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. |
| | | | | | | | | | | | | | | | |
| 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 can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCTNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

28.7.2 IWDG prescaler register (IWDG_PR)

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. |
| | | | | | | | | | | | | | | | |
| PR[3:0] | | | | | | | | | | | | | | | |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | rw | rw |

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

Bits 3:0 **PR[3:0]**: Prescaler divider

These bits are write access protected, see [Section 28.4.6](#). 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 to be able to change the prescaler divider.

0000: divider / 4
 0001: divider / 8
 0010: divider / 16
 0011: divider / 32
 0100: divider / 64
 0101: divider / 128
 0110: divider / 256
 0111: divider / 512
 Others: divider / 1024

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.

28.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|----------|------|------|------|------|------|------|------|------|------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 [Section 28.4.6](#). 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 prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

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, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

28.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in

software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 1.

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | EWIF | Res. | Res. | Res. | Res. | Res. | ONF | Res. | Res. | Res. | Res. | EWU | WVU | RVU | PVU |
| | r | | | | | | r | | | | | r | r | r | r |

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

Bit 14 **EWIF**: Watchdog early interrupt flag

This bit is set to '1' by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_EWCR register to '1'.

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit

Set to '1' by hardware as soon as the IWDG is started. In software mode, it remains to '1' until the IWDG is reset. In hardware mode, this bit is always set to '1'.

0: The IWDG is not activated

1: The IWDG is activated and needs to be refreshed regularly by the application

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

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain (takes up to one period of presc_ck and two periods of the IWDG kernel clock iwdg_ker_ck).

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

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 one period of presc_ck and two periods of the IWDG kernel clock iwdg_ker_ck).

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic "window" = 1.

Bit 1 **RVU**: Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to six periods of the IWDG kernel clock iwdg_ker_ck).

The 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 six periods of the IWDG kernel clock iwdg_ker_ck).

The prescaler value can be updated only when PVU bit is reset.

Note:

If several reload, prescaler, early interrupt position 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, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the

early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

28.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|-----------|------|------|------|------|------|------|------|------|------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 28.4.6](#). They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU 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 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.

28.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

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

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

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the [IWDG status register \(IWDG_SR\)](#) must be reset to be able to change the value of this bit.

Bit 14 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has no effect, reading this flag returns a 0.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]**: Watchdog counter window value

These bits are write access protected (see [Section 28.4.6](#)). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU 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 Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the [IWDG status register \(IWDG_SR\)](#) is reset.

28.7.7 IWDG register map

Table 163. 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x04 | IWDG_PR | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| | Reset value | | | | | | | | | | | | | | | | x | ONF | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0x08 | IWDG_RLR | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0C | IWDG_SR | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | 0 | EWIF | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | |
| | Reset value | | | | | | | | | | | | | | | | | 0 | EWIF | Res |
| 0x10 | IWDG_WINR | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x14 | IWDG_EWCR | Res | EWIE | Res | EWIC | Res | ONF | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | Reset value | 0 | 0 | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | Res | 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 55](#) for the register boundary addresses.

29 System window watchdog (WWDG)

29.1 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 a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A 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.

29.2 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 296](#))
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

29.3 WWDG implementation

Table 164. WWDG features⁽¹⁾

| WWDG mode / feature | WWDG |
|--|------|
| Window function | X |
| Early wake-up interrupt generation | X |
| System reset generation ⁽²⁾ | X |
| Capability to work in system Stop | - |
| Capability to work in system Standby | - |
| Capability to be frozen when the microcontroller enters in Debug mode ⁽³⁾ | X |
| Option bytes to control the Hardware mode ⁽⁴⁾ | X |

1. “X” = supported, “-” = not supported.

2. Refer to the RCC section for additional information.

3. Controlled via DBG_WWDG_STOP in DBG block.

4. Controlled via the option byte WWDG_SW. When WWDG_SW is set in HW mode the WWDG is running as soon as the CPU is in Run or Sleep modes.

29.4 WWDG functional description

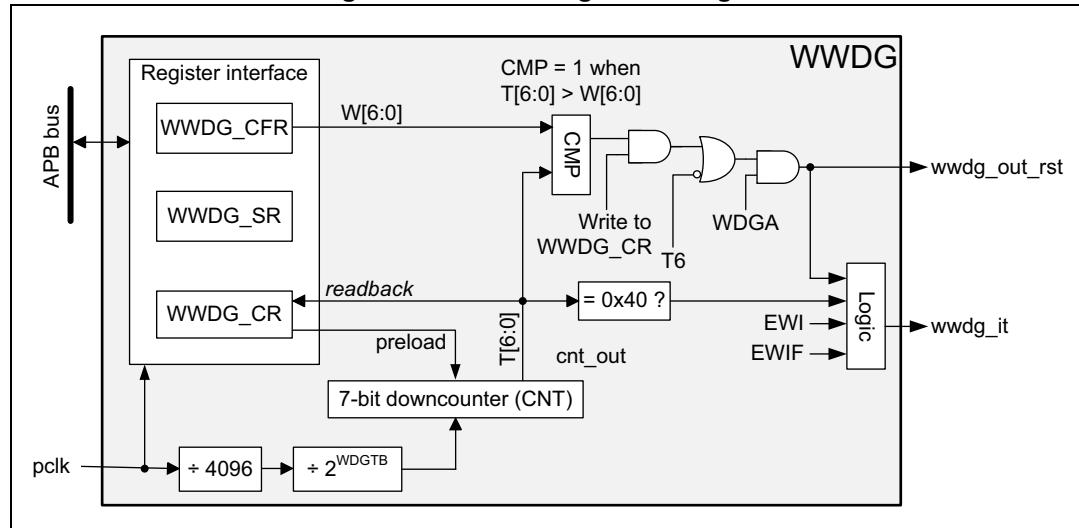
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a 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 295](#) for the WWDG block diagram.

29.4.1 WWDG block diagram

Figure 295. Watchdog block diagram



29.4.2 WWDG internal signals

[Table 165](#) gives the list of WWDG internal signals.

Table 165. WWDG internal input/output signals

| Signal name | Signal type | Description |
|--------------|----------------|-----------------------------|
| pclk | Digital input | APB bus clock |
| wwdg_out_rst | Digital output | WWDG reset signal output |
| wwdg_it | Digital output | WWDG early interrupt output |

29.4.3 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.

29.4.4 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 296](#)). 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 296](#) describes the window watchdog process.

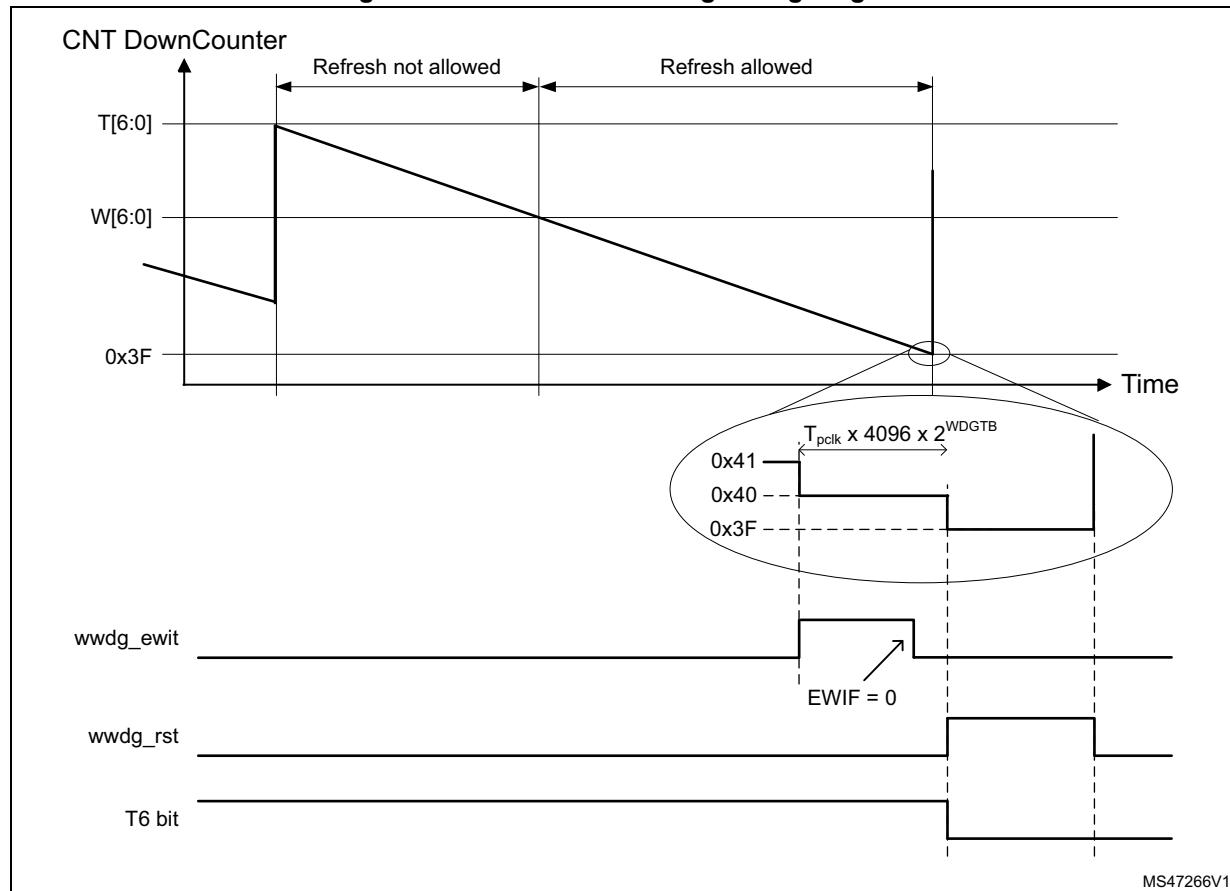
Note: *The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).*

29.4.5 How to program the watchdog timeout

Use the formula in [Figure 296](#) 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 296. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{\text{WWDG}} = t_{\text{PCLK}} \times 4096 \times 2^{\text{WDGTB}[2:0]} \times (T[5:0] + 1) \quad (\text{ms})$$

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:

$$t_{\text{WWDG}} = (1 / 48000) \times 4096 \times 2^3 \times (63 + 1) = 43.69\text{ms}$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG} .

29.4.6 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 37: Debug support \(DBG\)](#)

29.5 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.

29.6 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).

29.6.1 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 $(4096 \times 2^{\text{WDGTB}[2:0]})$ PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

29.6.2 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.

29.6.3 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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.

29.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 166. 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. | T[6:0] | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 1 1 1 1 1 1 1 | | | |
| 0x004 | WWDG_CFR | Res. | WDGTB [2:0] | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | W[6:0] | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | 0 0 0 | 0 | 0 | 1 1 1 1 1 1 1 | | | | | | | | | | |
| 0x008 | WWDG_SR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | EWIF | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

30 Real-time clock (RTC)

30.1 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.

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).

The RTC is functional in V_{BAT} mode.

30.2 RTC main features

The RTC supports the following features (see [Figure 297: RTC block diagram](#)):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the V_{BAT} pin.

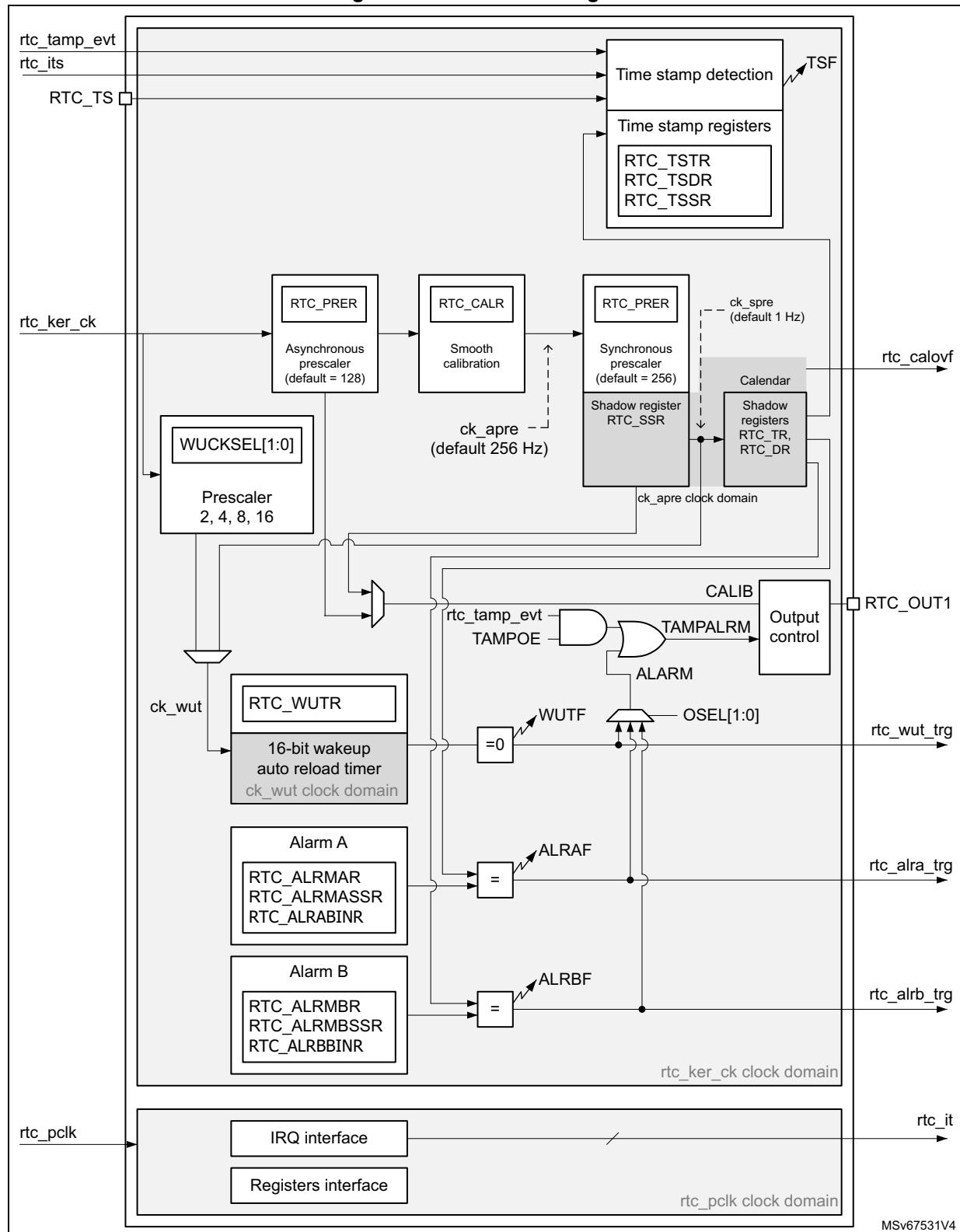
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

30.3 RTC functional description

30.3.1 RTC block diagram

Figure 297. RTC block diagram



MSv67531V4

30.3.2 RTC pins and internal signals

Table 167. RTC input/output pins

| Pin name | Signal type | Description |
|-----------|-------------|---------------------------------------|
| RTC_TS | Input | RTC timestamp input |
| RTC_REFIN | Input | RTC 50 or 60 Hz reference clock input |
| RTC_OUT1 | Output | RTC output 1 |
| RTC_OUT2 | Output | RTC output 2 |

RTC_OUT1 and RTC_OUT2 which select one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 168. RTC internal input/output signals

| Internal signal name | Signal type | Description |
|----------------------|-------------|--|
| rtc_ker_ck | Input | RTC kernel clock, also named RTCCLK in this document |
| rtc_pclk | Input | RTC APB clock |
| rtc_its | Input | RTC internal timestamp event |
| rtc_tamp_evt | Input | Tamper event (internal or external) detected in TAMP peripheral |
| rtc_it | Output | RTC interrupts (refer to Section 30.5: RTC interrupts for details) |
| rtc_alra_trg | Output | RTC alarm A event detection trigger |
| rtc_alrb_trg | Output | RTC alarm B event detection trigger |
| rtc_wut_trg | Output | RTC wake-up timer event detection trigger |
| rtc_calovf | Output | RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31 st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow. |

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or V_{BAT} when the selected clock is not LSE. Refer to [Section 30.4: RTC low-power modes](#) for more details.

Table 169. RTC interconnection

| Signal name | Source/destination |
|-------------|--|
| rtc_its | From power controller (PWR): main power loss/switch to V _{BAT} detection output |

The triggers outputs can be used as triggers for other peripherals.

30.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT1, RTC_TS and TAMP_IN1 are mapped on the same pin. The RTC and TAMP functions mapped on this pin are available in all power modes. This pin output mechanism follows the priority order shown in [Table 170](#).

Table 170. RTC pin configuration⁽¹⁾

| Pin function | | OSEL[1:0] (ALARM output enable) | TAMPOE (TAMPER output enable) | COE (CALIB output enable) | TAMPALRM_TYPE | TAMPALRM_PU | TAMP1E (TAMP_IN1 input enable) | TSE (RTC_TS input enable) |
|--|---------------------|------------------------------------|----------------------------------|------------------------------|---------------|-------------|-----------------------------------|------------------------------|
| TAMPALRM output Push-Pull | | 01 or 10 or 11 | 0 | Don't care | 0 | 0 | Don't care | Don't care |
| | | 00 | 1 | | | | | |
| | | 01 or 10 or 11 | 1 | | | | | |
| TAMPALRM output Open-Drain ⁽²⁾ | No pull | 01 or 10 or 11 | 0 | Don't care | 1 | 0 | Don't care | Don't care |
| | | 00 | 1 | | | | | |
| | | 01 or 10 or 11 | 1 | | | | | |
| | Internal pull-up | 01 or 10 or 11 | 0 | Don't care | 1 | 1 | Don't care | Don't care |
| | | 00 | 1 | | | | | |
| | | 01 or 10 or 11 | 1 | | | | | |

Table 170. RTC pin configuration⁽¹⁾ (continued)

| Pin function | OSEL[1:0] (ALARM output enable) | TAMPOE (TAMPER output enable) | COE (CALIB output enable) | TAMPALRM_TYPE | TAMPALRM_PU | TAMP1E (TAMP_IN1 input enable) | TSE (RTC_TS input enable) |
|------------------------------------|------------------------------------|----------------------------------|------------------------------|---------------|-------------|-----------------------------------|------------------------------|
| CALIB output PP | 00 | 0 | 1 | Don't care | Don't care | Don't care | Don't care |
| TAMP_IN1 input floating | 00 | 0 | 0 | Don't care | Don't care | 1 | 0 |
| | 00 | 0 | 1 | | | | |
| | Don't care | Don't care | 0 | | | | |
| RTC_TS and TAMP_IN1 input floating | 00 | 0 | 0 | Don't care | Don't care | 1 | 1 |
| | 00 | 0 | 1 | | | | |
| | Don't care | Don't care | 0 | | | | |
| RTC_TS input floating | 00 | 0 | 0 | Don't care | Don't care | 0 | 1 |
| | 00 | 0 | 1 | | | | |
| | Don't care | Don't care | 0 | | | | |
| Wakeup pin or Standard GPIO | 00 | 0 | 0 | Don't care | Don't care | 0 | 0 |
| | 00 | 0 | 1 | | | | |
| | Don't care | Don't care | 0 | | | | |

1. OD: open drain; PP: push-pull.

2. In this configuration the GPIO must be configured in input.

In addition, it is possible to output RTC_OUT2 on PB2 pin thanks to OUT2EN bit. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in [Table 171: RTC_OUT mapping](#).

Table 171. RTC_OUT mapping

| RTC_OUT mapping | | | | |
|---|----------------------------------|---------------|---------------------|--------------------|
| OSEL[1:0] bits ALARM output enable) | COE bit (CALIB output enable) | OUT2EN bit | RTC_OUT1 on PC13 | RTC_OUT2 on PB2 |
| 00 | 0 | 0 | - | - |
| 00 | 1 | | CALIB | - |
| 01 or 10 or 11 | Don't care | | TAMPALRM | - |

Table 171. RTC_OUT mapping (continued)

| RTC_OUT mapping | | | | |
|---|----------------------------------|---------------|---------------------|--------------------|
| OSEL[1:0] bits ALARM output enable) | COE bit (CALIB output enable) | OUT2EN bit | RTC_OUT1 on PC13 | RTC_OUT2 on PB2 |
| 00 | 0 | 1 | - | - |
| 00 | 1 | | - | CALIB |
| 01 or 10 or 11 | 0 | | - | TAMPALRM |
| 01 or 10 or 11 | 1 | | TAMPALRM | CALIB |

30.3.4 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) $\geq 2 \times$ frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

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 297: 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:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{\text{PREDIV_A} + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(\text{PREDIV_S} + 1) \times (\text{PREDIV_A} + 1)}$$

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 30.3.8: Periodic auto-wake-up](#) for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

30.3.5 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 periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see [Section 30.6.10: RTC shift control register \(RTC_SHIFTR\)](#)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

30.3.6 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to [Section : RTC ultra-low-power mode](#)).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

30.3.7 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.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

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 TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

30.3.8 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 ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.
When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 μ s to 32 s, with a resolution down to 61 μ s.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.
When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in 2 parts:
 - from 1 s to 18 hours when WUCKSEL [2:1] = 10
 - and from around 18 h to 36 h 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 910](#)), 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_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value ($0x0000 < \text{WUTOCLR} \leq \text{WUT}$).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section [Section 30.3.1: RTC block diagram](#)).

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 TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM 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.

30.3.9 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bits INIT, BIN and BCDU in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

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_ICSR 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_ICSR register. The initialization phase mode is entered when INITF is set to 1.
 - If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
 - If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR 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.
 - If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
 - If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of $1 / (\text{PREDIV_S} + 1)$ 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 μ s 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 dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: *After a system reset, the application can read the INITS flag in the RTC_ICSR 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).*

Note: *To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR 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 or RTC_ALRABINR).
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_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode.
 - If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
 - If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[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.
 - If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
 - If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

30.3.10 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_ICSR 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 periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (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 909](#)): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to [Section 30.3.12: 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.

30.3.11 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) 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 wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about 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.

30.3.12 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond 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.

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 with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

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:** In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.
- Caution:** Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.
- Caution:** This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

30.3.13 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

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 asynchronous 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
- PREDIV_S = 0x00FF

Note: *RTC_REFIN clock detection is not available in Standby mode.*

30.3.14 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 ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTC ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about $2^{20} \times$ PREDIV_A \times RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL=0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration 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 ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration 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 permits 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 ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration 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 (FCAL) given the input frequency (FRTCCLK) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

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.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. 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 calibration 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

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

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 8 s). 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_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/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.

30.3.15 Timestamp function

Timestamp 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 timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDDR) when an internal or external tamper event is detected. Refer to [RTC control register \(RTC_CR\)](#) and refer to [Section : Timestamp on tamper event.](#)

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the V_{BAT} supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDDR) maintain the results of the previous event.

Note: *TSF is set up to 2 ck_apre cycles after the timestamp event from RTC_TS pin or from rtc_its internal signal occurs due to synchronization process. TSF is set up to 3 ck_apre cycles after tamper flags.*

TSOVF is set up to only 1 ck_apre cycle after the event occurs. This means that if two timestamp 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 timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

30.3.16 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK}/64$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The 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 CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = 0x7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: *When COSEL is cleared, the CALIB output is the output of the 6th stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.*

When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

30.3.17 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note:

Once the TAMPALRM output is enabled, it has priority over CALIB on RTC_OUT1.

In case the TAMPALRM is configured open-drain in the RTC, the RTC_OUT1 GPIO must be configured as input.

30.4 RTC low-power modes

Table 172. Effect of low-power modes on RTC

| Mode | Description |
|----------|---|
| Sleep | No effect RTC interrupts cause the device to exit the Sleep mode. |
| Stop | The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode. |
| Standby | The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode. |
| Shutdown | The RTC remains active when the RTC clock source is LSE. RTC interrupts cause the device to exit the Shutdown mode. |

The table below summarizes the RTC pins and functions capability in all modes.

Table 173. RTC pins functionality over modes

| Functions | Functional in all low-power modes except Stop 2, Standby and Shutdown modes | Functional in Stop 2, Standby and Shutdown modes | Functional in V _{BAT} mode |
|-----------|---|--|-------------------------------------|
| RTC_TS | Yes | Yes | Yes |
| RTC_REFIN | Yes | No | No |
| RTC_OUT1 | Yes | Yes | PC13: Yes PA1: No |

Table 173. RTC pins functionality over modes (continued)

| Functions | Functional in all low-power modes except Stop 2, Standby and Shutdown modes | Functional in Stop 2, Standby and Shutdown modes | Functional in V _{BAT} mode |
|-----------|---|--|-------------------------------------|
| RTC_OUT2 | Yes | No | No |

30.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

Table 174. Interrupt requests

| Interrupt acronym | Interrupt event | Event flag ⁽¹⁾ | Enable control bit ⁽²⁾ | Interrupt clear method | Exit from low-power modes |
|-------------------|------------------------|---------------------------|-----------------------------------|------------------------|---------------------------|
| RTC | Alarm A | ALRAF | ALRAIE | write 1 in CALRAF | Yes ⁽³⁾ |
| | Alarm B | ALRBF | ALRBIE | write 1 in CALRBF | Yes ⁽³⁾ |
| | Timestamp | TSF | TSIE | write 1 in CTSF | Yes ⁽³⁾ |
| | Wake-up timer | WUTF | WUTIE | write 1 in CWUTF | Yes ⁽³⁾ |
| | SSR underflow (reload) | SSRUF | SSRUIE | write 1 in CSSRUF | Yes ⁽³⁾ |

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

30.6 RTC registers

Refer to [Section 1.2](#) of the reference manual for a list of abbreviations used in register descriptions.

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

30.6.1 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 909](#) and [Reading the calendar on page 911](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 909](#).

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 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

30.6.2 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 909](#) and [Reading the calendar on page 911](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 909](#).

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 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

Note: *The calendar is frozen when reaching the maximum value, and can't roll over.*

30.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 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 |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| SS[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 |
| SS[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values

When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):

SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[31:16] are forced by hardware to 0x0000.

SS[15:0]: Subsecond value/synchronous binary counter LSB values

When Binary mode is selected (BIN = 01 or 10 or 11):

SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

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)

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.

30.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 909](#).

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. | Res. | RECAL PF |
| | | | | | | | | | | | | | | | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | BCDU[2:0] | | | BIN[1:0] | | INIT | INITF | RSF | INITS | SHPF | WUTWF | Res. | Res. |
| | | | rw | rw | rw | rw | rw | r | rc_w0 | r | r | r | | | |

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

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](#).

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

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

- 0x0: 1s calendar increment is generated each time SS[7:0] = 0
- 0x1: 1s calendar increment is generated each time SS[8:0] = 0
- 0x2: 1s calendar increment is generated each time SS[9:0] = 0
- 0x3: 1s calendar increment is generated each time SS[10:0] = 0
- 0x4: 1s calendar increment is generated each time SS[11:0] = 0
- 0x5: 1s calendar increment is generated each time SS[12:0] = 0
- 0x6: 1s calendar increment is generated each time SS[13:0] = 0
- 0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

- 00: Free running BCD calendar mode (Binary mode disabled).
- 01: Free running Binary mode (BCD mode disabled)
- 10: Free running BCD calendar and Binary modes
- 11: Free running BCD calendar and Binary modes

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), plus BIN and BCDU fields. 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 (BYPSSHAD = 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

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFT register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Wake-up timer configuration update not allowed except in initialization mode

1: Wake-up timer configuration update allowed

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

30.6.5 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 909](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 909](#).

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:

$$\text{ck_apre frequency} = \text{RTCCLK frequency}/(\text{PREDIV_A}+1)$$

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:

$$\text{ck_spre frequency} = \text{ck_apre frequency}/(\text{PREDIV_S}+1)$$

30.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

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 |
|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| WUTOCLR[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 |
| WUT[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16 **WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

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 between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

30.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 909](#).

Address offset: 0x18

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 |
|------------|-----------------------|---------------------|-------------|--------------|------------|------------|-------|------------|-----------|-------------|-------------|------------|--------------|-------|-------|
| OUT2 EN | TAMP ALRM_ TYPE | TAMP ALRM_ PU | ALRB CLR | ALRAF CLR | TAMP OE | TAMP TS | ITSE | COE | OSEL[1:0] | | POL | COSEL | BKP | SUB1H | ADD1H |
| rw | rw | rw | rw | rw | rw | rw | 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 | ALRB IE | ALRA IE | TSE | WUTE | ALRBE | ALRAE | SSR UIE | FMT | BYP SHAD | REFCK ON | TS EDGE | WUCKSEL[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:

OUT2EN = 0: RTC output 2 disable

If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1

If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1

OUT2EN = 1: RTC output 2 enable

If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2

If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2

If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output

1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output

1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.

1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.

1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM

0: The tamper flag is not routed on TAMPALRM

1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a RTC timestamp to be saved

1: Save RTC timestamp on tamper detection event

TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to 3 ck_apre cycles after the tamper flags.

Note: TAMPTS must be cleared before entering RTC initialization mode.

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 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 TAMPALRM 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 TAMPALRM output.

- 0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
- 1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection

When COE = 1, this bit selects which signal is output on CALIB.

- 0: Calibration output is 512 Hz
- 1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to [Section 30.3.16: 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**: Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

- 0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

- 0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable

- 0: Timestamp interrupt disable
- 1: Timestamp 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 **SSRUIE**: SSR underflow interrupt enable

- 0: SSR underflow interrupt disabled
- 1: SSR underflow interrupt enabled

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: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge

- 0: RTC_TS input rising edge generates a timestamp event
- 1: RTC_TS input falling edge generates a timestamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut 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 in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.

11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/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_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

30.6.8 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

30.6.9 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 909](#).

Address offset: 0x28

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 | CALW16 | LPCAL | Res. | CALM[8:0] |
| 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:
 $(512 \times \text{CALP}) - \text{CALM}$.

Refer to [Section 30.3.14: 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 30.3.14: 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 30.3.14: RTC smooth digital calibration](#).

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 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 30.3.14: RTC smooth digital calibration on page 913](#).

30.6.10 RTC shift control register (RTC_SHIFTTR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 909](#).

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. | 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 | 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_ICSR).

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_ICSR).

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))).

In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

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.

30.6.11 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. 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 | 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.

30.6.12 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. 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

30.6.13 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the 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 |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| SS[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 |
| SS[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values

SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

30.6.14 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

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 |
|------|----------|---------|----|----------|----|----|----|------|---------|---------|---------|---------|----|----|----|
| 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.

30.6.15 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

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 |
|---------------|------|-------------|----|----|----|----|----|----|------|------|------|------|------|------|------|
| SSCLR | Res. | MASKSS[5:0] | | | | | | | Res. |
| rw | | rw | rw | rw | rw | rw | rw | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. SS[14:0] | | | | | | | | | | | | | | | |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | w | rw | rw |

- Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
 0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
 1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

- Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit
 0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
 1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.
 2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.
 ...
 31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.
 From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits 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]**: Subseconds 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.
 This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

30.6.16 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

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 |
|------|----------|---------|----|----------|----|----|----|------|---------|---------|----|---------|----|----|----|
| MSK4 | WD SEL | 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

30.6.17 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

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 |
|-------|------|-------------|----|-------------|----|------|------|----------|------|------|------|------|------|------|------|
| SSCLR | Res. | MASKSS[5:4] | | MASKSS[3:0] | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| rw | | rw | rw | rw | rw | rw | rw | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | | | | | | | | SS[14:0] | | | | | | | |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | w | rw | rw |

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00) 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]**: Subseconds 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.

This field is the mirror of SS[14:0] in the RTC_ALRBINR, and so can also be read or written through RTC_ALRBINR.

30.6.18 RTC status register (RTC_SR)

Address offset: 0x50

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SSR UF | ITSF | TSOVF | TSF | WUTF | ALRBF | ALRAF |
| | | | | | | | | | r | r | r | r | r | r | r |

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

Bit 6 **SSRUF**: SSR underflow flag

This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.

Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload.

Bit 5 **ITSF**: Internal timestamp flag

This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

If ITSF flag is set, TSF must be cleared together with ITSF.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event. Refer to [Timestamp on tamper event](#) for more details.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.

If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.

If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **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).

Bit 0 **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).

Note: *The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.*

30.6.19 RTC masked interrupt status register (RTC_MISR)

Address offset: 0x54

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SSR UMF | ITS MF | TSOV MF | TS MF | WUT MF | ALRB MF | ALRA MF |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bit 6 **SSRUMF**: SSR underflow masked flag

This flag is set by hardware when the SSR underflow interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp masked flag

This flag is set by hardware when a timestamp on the internal event occurs and timestampinterrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow masked flag

This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp masked flag

This flag is set by hardware when a timestamp interrupt occurs.

If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer masked flag

This flag is set by hardware when the wake-up timer interrupt occurs.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B masked flag

This flag is set by hardware when the alarm B interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag

This flag is set by hardware when the alarm A interrupt occurs.

Note: *The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.*

30.6.20 RTC status clear register (RTC_SCR)

Address offset: 0x5C

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | CSSR UF | CITS F | CTSOV F | CTS F | CWUT F | CALRB F | CALRA F |
| | | | | | | | | | w | w | w | w | w | w | w |

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

Bit 6 **CSSRUF**: Clear SSR underflow flag

Writing '1' in this bit clears the SSRUF in the RTC_SR register.

Bit 5 **CITSF**: Clear internal timestamp flag

Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag

Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag

Writing 1 in this bit clears the TSF bit in the RTC_SR register.

If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wake-up timer flag

Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag

Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag

Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

30.6.21 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

Address offset: 0x70

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 |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| SS[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 |
| SS[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSRR, and so can also be read or written through RTC_ALRMASSR.

30.6.22 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

Address offset: 0x74

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 |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| SS[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 |
| SS[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSRR, and so can also be read or written through RTC_ALRMBSSR.

30.6.23 RTC register map

Table 175. RTC register map and reset values

Table 175. RTC register map and reset values (continued)

Refer to [Section 2.2](#) for the register boundary addresses.

31 Tamper and backup registers (TAMP)

31.1 Introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 9 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 5 tamper pins and 4 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering.

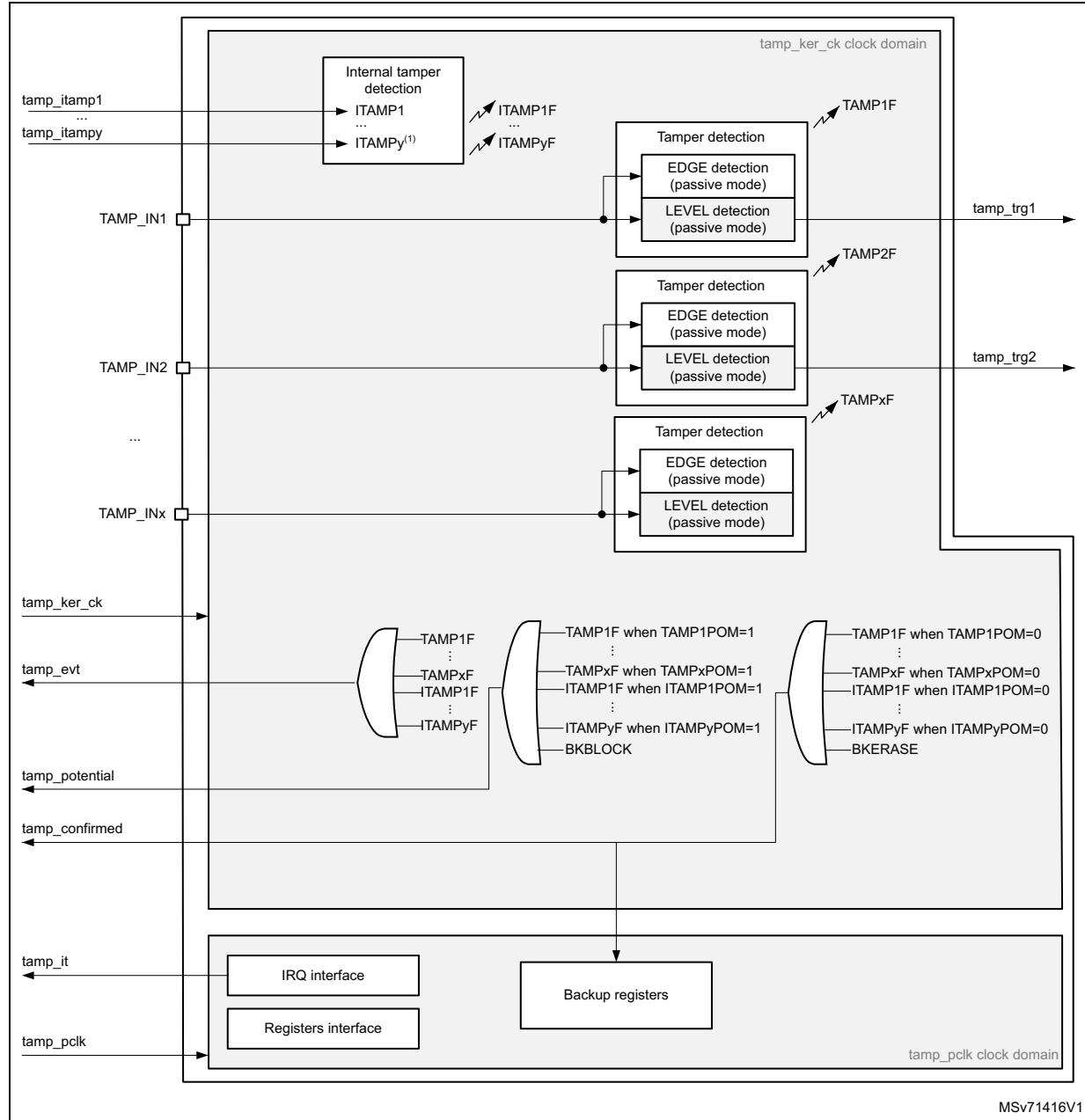
31.2 TAMP main features

- A tamper detection can optionally erase the backup registers and backup SRAM. The device resources protected by tamper are named “device secrets”.
- 9 32-bit backup registers:
 - The backup registers (TAMP_BKPxR) are implemented in the backup domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- 5 tamper pin for 5 external tamper detection event:
 - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
 - Configurable digital filter.
- 4 internal tamper events to protect against transient attacks
- Each tamper can be configured in two modes:
 - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
 - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.

31.3 TAMP functional description

31.3.1 TAMP block diagram

Figure 298. TAMP block diagram



1. The number of external and internal tampers depends on products.

31.3.2 TAMP pins and internal signals

Table 176. TAMP input/output pins

| Pin name | Signal type | Description |
|--------------------------|-------------|------------------|
| TAMP_INx (x = pin index) | Input | Tamper input pin |

Table 177. TAMP internal input/output signals

| Internal signal name | Signal type | Description |
|----------------------------------|-------------|---|
| tamp_ker_ck | Input | TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document |
| tamp_pclk | Input | TAMP APB clock, connected to rtc_pclk |
| tamp_itamp[y] (y = signal index) | Inputs | Internal tamper event sources |
| tamp_evt | Output | Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration. |
| tamp_potential | Output | Potential tamper detection signal, used for device secrets ⁽¹⁾ protection. This signal is active when: <ul style="list-style-type: none">– a tamper event detection flag (internal or external tamper), is generated in potential mode.– or a software request is done by writing BKBLOCK to 1 |
| tamp_confirmed | Output | Confirmed tamper detection signal, used for device secrets ⁽¹⁾ protection. This signal is active when: <ul style="list-style-type: none">– a tamper event detection flag (internal or external tamper), is generated in confirmed mode.– or a software request is done by writing BKERASE to 1 |
| tamp_it | Output | TAMP interrupt (refer to Section 31.5: TAMP interrupts for details) |
| tamp_trg[x] (x = signal index) | Output | Tamper detection trigger |

1. Refer to [Table 178: TAMP interconnection](#).

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or V_{BAT} depending on the selected clock (refer to [Section 31.4: TAMP low-power modes](#) for more details).

Table 178. TAMP interconnection

| Signal name | Source/Destination |
|----------------|--|
| tamp_evt | rtc_tamp_evt used to generate a timestamp event |
| tamp_potential | The tamp_potential signal is used to block the read and write accesses to the device secrets listed hereafter: – backup registers – BKPRAM |
| tamp_confirmed | The tamp_confirmed signal is used to erase the device secrets listed hereafter: – backup registers – BKPRAM The device secrets access is blocked when erase is ongoing. |
| tamp_itamp3 | LSE monitoring (LSECSS)(⁽¹⁾) |
| tamp_itamp4 | HSE monitoring (CSS)(⁽²⁾) |
| tamp_itamp5 | RTC calendar overflow (rtc_calovf) |
| tamp_itamp6 | ST manufacturer readout |

1. This monitoring must be enabled by setting LSECSSON in [RTC domain control register \(RCC_BDCR\)](#).
2. This monitoring must be enabled by setting HSECSSON in RCC control register (RCC_CR).

31.3.3 GPIOs controlled by the RTC and TAMP

Refer to [Section 30.3.3: GPIOs controlled by the RTC and TAMP](#).

31.3.4 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

31.3.5 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- erase the backup registers and other device secrets stored in SRAMs or peripherals listed in [Table 178: TAMP interconnection](#)
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in [Table 178: TAMP interconnection](#)
- generate an interrupt, capable to wake-up from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or level.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets

erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

31.3.6 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 in the TAMP_CR2 register).

Note: *The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.*

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with `tamp_confirmed` signal in the [Table 178: TAMP interconnection](#).

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with `tamp_potential` signal in the [Table 178: TAMP interconnection](#).

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to [Table 178: TAMP interconnection](#)). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

31.3.7 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in level detection with filtering mode (TAMPFLT ≠ 00). Refer to [Section : Level detection with filtering on tamper inputs \(passive mode\)](#).

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs.

Note: *TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).*

Note: *If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.*

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx 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 TAMP_INx inputs are deactivated when edge detection is selected.

Caution: 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 TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPXR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx 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 TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

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 TAMP_INx 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 TAMP_INx 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 microcontroller datasheet for the electrical characteristics of the pull-up resistors.

31.4 TAMP low-power modes

Table 179. Effect of low-power modes on TAMP

| Mode | Description |
|----------|---|
| Sleep | No effect. TAMP interrupts cause the device to exit the Sleep mode. |
| Stop | No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Stop mode. |
| Standby | No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Standby mode. |
| Shutdown | No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE. TAMP interrupts cause the device to exit the Shutdown mode. |

Table 180. TAMP pins functionality over modes

| Pin name | Functional in all low-power modes | Functional in V_{BAT} mode |
|--------------|-----------------------------------|------------------------------|
| TAMP_IN[5:0] | Yes | Yes |

31.5 TAMP interrupts

Table 181. Interrupt requests

| Interrupt acronym | Interrupt event | Event flag ⁽¹⁾ | Enable control bit | Interrupt clear method | Exit from low-power modes |
|-------------------|----------------------------------|---------------------------|--------------------|------------------------|---------------------------|
| TAMP | Tamper x ⁽²⁾ | TAMPxF | TAMPxIE | Write 1 in CTAMPxF | Yes ⁽³⁾ |
| | Internal tamper y ⁽²⁾ | ITAMPyF | ITAMPyIE | Write 1 in CITAMPyF | Yes ⁽³⁾ |

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to [Table 179: Effect of low-power modes on TAMP](#) for more details about available features in the low-power modes.

31.6 TAMP registers

Refer to [Section 1.2](#) of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

31.6.1 TAMP control register 1 (TAMP_CR1)

Address offset: 0x000

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. | ITAMP6 E | ITAMP5 E | ITAMP4 E | ITAMP3 E | Res. | Res. |
| | | | | | | | | | | rw | rw | rw | rw | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TAMP5 E | TAMP4 E | TAMP3 E | TAMP2 E | TAMP1 E | |
| | | | | | | | | | | rw | rw | rw | rw | rw | |

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

- Bit 22 Reserved, must be kept at reset value.
- Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.
- Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.
- Bit 19 **ITAMP4E**: Internal tamper 4 enable
0: Internal tamper 4 disabled.
1: Internal tamper 4 enabled.
- Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.
- Bit 17 Reserved, must be kept at reset value.
- Bit 16 Reserved, must be kept at reset value.
- Bits 15:8 Reserved, must be kept at reset value.
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 Reserved, must be kept at reset value.
- Bit 4 **TAMP5E**: Tamper detection on TAMP_IN5 enable⁽¹⁾
0: Tamper detection on TAMP_IN5 is disabled.
1: Tamper detection on TAMP_IN5 is enabled.
- Bit 3 **TAMP4E**: Tamper detection on TAMP_IN4 enable⁽¹⁾
0: Tamper detection on TAMP_IN4 is disabled.
1: Tamper detection on TAMP_IN4 is enabled.
- Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable⁽¹⁾
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.
- Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable⁽¹⁾
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.
- Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable⁽¹⁾
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR register and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

31.6.2 TAMP control register 2 (TAMP_CR2)

Address offset: 0x04

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. | TAMP5 TRG | TAMP4 TRG | TAMP3 TRG | TAMP2 TRG | TAMP1 TRG | BK ERASE | BK BLOCK | Res. | Res. | Res. | TAMP3 MSK | TAMP2 MSK | TAMP1 MSK |
| | | | rw | rw | rw | rw | rw | w | 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. | TAMP5 POM | TAMP4 POM | TAMP3 POM | TAMP2 POM | TAMP1 POM |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **TAMP5TRG**: Active level for tamper 5 input (active mode disabled)

- 0: If TAMPFLT ≠ 00 tamper 5 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 5 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 5 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 5 input falling edge triggers a tamper detection event.

Bit 27 **TAMP4TRG**: Active level for tamper 4 input (active mode disabled)

- 0: If TAMPFLT ≠ 00 tamper 4 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 4 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 4 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 4 input falling edge triggers a tamper detection event.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input

- 0: If TAMPFLT ≠ 00 tamper 3 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 3 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input falling edge triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input

- 0: If TAMPFLT ≠ 00 tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input falling edge triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input

- 0: If TAMPFLT ≠ 00 tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input falling edge triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets⁽¹⁾ erase

Writing ‘1’ to this bit reset the backup registers and device secrets⁽¹⁾. Writing 0 has no effect.
This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets⁽¹⁾ access blocked
 0: backup registers and device secrets⁽¹⁾ can be accessed if no tamper flag is set
 1: backup registers and device secrets⁽¹⁾ cannot be accessed

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask
 0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.
 1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and device secrets⁽¹⁾ are not erased.
The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask
 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 and device secrets⁽¹⁾ are not erased.
The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask
 0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.
 1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets⁽¹⁾ are not erased.
The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

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

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TAMP5POM**: Tamper 5 potential mode
 0: Tamper 5 event detection is in confirmed mode⁽¹⁾.
 1: Tamper 5 event detection is in potential mode⁽²⁾.

Bit 3 **TAMP4POM**: Tamper 4 potential mode
 0: Tamper 4 event detection is in confirmed mode⁽¹⁾.
 1: Tamper 4 event detection is in potential mode⁽²⁾.

Bit 2 **TAMP3POM**: Tamper 3 potential mode
 0: Tamper 3 event detection is in confirmed mode⁽¹⁾.
 1: Tamper 3 event detection is in potential mode⁽²⁾.

Bit 1 **TAMP2POM**: Tamper 2 potential mode
 0: Tamper 2 event detection is in confirmed mode⁽¹⁾.
 1: Tamper 2 event detection is in potential mode⁽²⁾.

Bit 0 **TAMP1POM**: Tamper 1 potential mode
 0: Tamper 1 event detection is in confirmed mode⁽¹⁾.
 1: Tamper 1 event detection is in potential mode⁽²⁾.

1. The effects of tamper detection in confirmed mode is described with `tamp_confirmed` signal in [Table 178: TAMP interconnection](#).
2. The effects of tamper detection in potential mode is described with `tamp_potential` signal in [Table 178: TAMP interconnection](#).

31.6.3 TAMP control register 3 (TAMP_CR3)

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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | ITAMP6 POM | ITAMP5 POM | ITAMP4 POM | ITAMP3 POM | Res. | Res. |
| | | | | | | | | | | rw | rw | rw | rw | | |

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

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode

0: Internal tamper 6 event detection is in confirmed mode⁽¹⁾.

1: Internal tamper 6 event detection is in potential mode⁽²⁾.

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode

0: Internal tamper 5 event detection is in confirmed mode⁽¹⁾.

1: Internal tamper 5 event detection is in potential mode⁽²⁾.

Bit 3 **ITAMP4POM**: Internal tamper 4 potential mode

0: Internal tamper 4 event detection is in confirmed mode⁽¹⁾.

1: Internal tamper 4 event detection is in potential mode⁽²⁾.

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode

0: Internal tamper 3 event detection is in confirmed mode⁽¹⁾.

1: Internal tamper 3 event detection is in potential mode⁽²⁾.

Bit 1 Reserved, must be kept at reset value.

Bit 0 Reserved, must be kept at reset value.

1. The effects of internal tamper detection in confirmed mode is described with `tamp_confirmed` signal in [Table 178: TAMP interconnection](#)

2. The effects of internal tamper detection in potential mode is described with `tamp_potential` signal in [Table 178: TAMP interconnection](#).

31.6.4 TAMP filter control register (TAMP_FLTCR)

Address offset: 0x0C

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TAMP PUDIS | TAMPPRCH [1:0] | TAMPFLT [1:0] | TAMPFREQ [2:0] | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bit 7 TAMPPUDIS: TAMP_INx pull-up disable

This bit determines if each of the TAMPx pins are precharged before each sample.

0: Precharge TAMP_INx pins before sampling (enable internal pull-up)

1: Disable precharge of TAMP_INx pins.

Bits 6:5 TAMPPRCH[1:0]: TAMP_INx 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 TAMP_INx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 TAMPFLT[1:0]: TAMP_INx 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 TAMP_INx inputs.

0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx 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 2:0 TAMPFREQ[2:0]: Tamper sampling frequency

Determines the frequency at which each of the TAMP_INx 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)

Note: This register concerns only the tamper inputs in passive mode.

31.6.5 TAMP interrupt enable register (TAMP_IER)

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 |
| Res. | ITAMP6 IE | ITAMP5 IE | ITAMP4 IE | ITAMP3 IE | Res. | Res. |
| | | | | | | | | | | rw | rw | rw | rw | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TAMP 5IE | TAMP 4IE | TAMP 3IE | TAMP 2IE | TAMP 1IE | |
| | | | | | | | | | | rw | rw | rw | rw | rw | |

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable

0: Internal tamper 6 interrupt disabled.

1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable

0: Internal tamper 5 interrupt disabled.

1: Internal tamper 5 interrupt enabled.

Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable

0: Internal tamper 4 interrupt disabled.

1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable

0: Internal tamper 3 interrupt disabled.

1: Internal tamper 3 interrupt enabled.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

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

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TAMP5IE**: Tamper 5 interrupt enable

- 0: Tamper 5 interrupt disabled.
- 1: Tamper 5 interrupt enabled.

Bit 3 **TAMP4IE**: Tamper 4 interrupt enable

- 0: Tamper 4 interrupt disabled.
- 1: Tamper 4 interrupt enabled.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable

- 0: Tamper 3 interrupt disabled.
- 1: Tamper 3 interrupt enabled..

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable

- 0: Tamper 2 interrupt disabled.
- 1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable

- 0: Tamper 1 interrupt disabled.
- 1: Tamper 1 interrupt enabled.

31.6.6 TAMP status register (TAMP_SR)

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. | ITAMP6 F | ITAMP5 F | ITAMP4 F | ITAMP3 F | Res. | Res. |
| | | | | | | | | | | r | r | r | r | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TAMP 5F | TAMP 4F | TAMP 3F | TAMP 2F | TAMP 1F | |
| | | | | | | | | | | r | r | r | r | r | |

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6F**: Internal tamper 6 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: Internal tamper 4 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: Internal tamper 3 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

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

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TAMP5F**: TAMP5 detection flag

This flag is set by hardware when a tamper detection event is detected on the TAMP5 input.

Bit 3 **TAMP4F**: TAMP4 detection flag

This flag is set by hardware when a tamper detection event is detected on the TAMP4 input.

Bit 2 **TAMP3F**: TAMP3 detection flag

This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag

This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag

This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

31.6.7 TAMP masked interrupt status register (TAMP_MISR)

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. | ITAMP6 MF | ITAMP5 MF | ITAMP4 MF | ITAMP3 MF | Res. | Res. |
| | | | | | | | | | | r | r | r | r | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TAMP 5MF | TAMP 4MF | TAMP 3MF | TAMP 2MF | TAMP 1MF | |
| | | | | | | | | | | r | r | r | r | r | |

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

- Bit 28 Reserved, must be kept at reset value.
- Bit 27 Reserved, must be kept at reset value.
- Bit 26 Reserved, must be kept at reset value.
- Bit 25 Reserved, must be kept at reset value.
- Bit 24 Reserved, must be kept at reset value.
- Bit 23 Reserved, must be kept at reset value.
- Bit 22 Reserved, must be kept at reset value.
- Bit 21 **ITAMP6MF**: Internal tamper 6 interrupt masked flag
This flag is set by hardware when the internal tamper 6 interrupt is raised.
- Bit 20 **ITAMP5MF**: Internal tamper 5 interrupt masked flag
This flag is set by hardware when the internal tamper 5 interrupt is raised.
- Bit 19 **ITAMP4MF**: Internal tamper 4 interrupt masked flag
This flag is set by hardware when the internal tamper 4 interrupt is raised.
- Bit 18 **ITAMP3MF**: Internal tamper 3 interrupt masked flag
This flag is set by hardware when the internal tamper 3 interrupt is raised.
- Bit 17 Reserved, must be kept at reset value.
- Bit 16 Reserved, must be kept at reset value.
- Bits 15:8 Reserved, must be kept at reset value.
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 Reserved, must be kept at reset value.
- Bit 4 **TAMP5MF**: TAMP5 interrupt masked flag
This flag is set by hardware when the tamper 5 interrupt is raised.
- Bit 3 **TAMP4MF**: TAMP4 interrupt masked flag
This flag is set by hardware when the tamper 4 interrupt is raised.
- Bit 2 **TAMP3MF**: TAMP3 interrupt masked flag
This flag is set by hardware when the tamper 3 interrupt is raised.
- Bit 1 **TAMP2MF**: TAMP2 interrupt masked flag
This flag is set by hardware when the tamper 2 interrupt is raised.
- Bit 0 **TAMP1MF**: TAMP1 interrupt masked flag
This flag is set by hardware when the tamper 1 interrupt is raised.

31.6.8 TAMP status clear register (TAMP_SCR)

Address offset: 0x3C

System reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|------|------|------------------|------------------|------------------|------------------|-------------|------|
| Res. | C ITAMP 6F | C ITAMP 5F | C ITAMP 4F | C ITAMP 3F | Res. | Res. |
| | | | | | | | | | | w | w | w | w | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | CTAMP 5F | CTAMP 4F | CTAMP 3F | CTAMP 2F | CTAMP 1F | |
| | | | | | | | | | | w | w | w | w | w | |

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CITAMP6F:** Clear ITAMP6 detection flag

Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F:** Clear ITAMP5 detection flag

Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 **CITAMP4F:** Clear ITAMP4 detection flag

Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F:** Clear ITAMP3 detection flag

Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

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

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTAMP5F:** Clear TAMP5 detection flag

Writing 1 in this bit clears the TAMP5F bit in the TAMP_SR register.

Bit 3 **CTAMP4F:** Clear TAMP4 detection flag

Writing 1 in this bit clears the TAMP4F bit in the TAMP_SR register.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag

Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag

Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag

Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

31.6.9 TAMP backup x register (TAMP_BKPxR)

Address offset: $0x100 + 0x04 * x$, ($x = 0$ to 8)

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.

In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

31.6.10 TAMP register map

Table 182. TAMP register map and reset values

Refer to [Section 2.2](#) for the register boundary addresses.



32 Inter-integrated circuit interface (I2C)

32.1 Introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multimaster capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

I2C peripheral can use DMA to reduce the CPU load.

32.2 I2C main features

- I²C-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
 - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- Independent clock
- Wake-up from Stop mode on address match

For information on I2C instantiation, refer to [Section 32.3: I2C implementation](#).

32.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 183. I2C implementation

| I2C features ⁽¹⁾ | I2C1 | I2C2 | I2C3 | I2C4 ⁽²⁾ |
|----------------------------------|------|------|------|---------------------|
| 7-bit addressing mode | X | X | X | X |
| 10-bit addressing mode | X | X | X | X |
| Standard-mode (up to 100 kbit/s) | X | X | X | X |
| Fast-mode (up to 400 kbit/s) | X | X | X | X |

Table 183. I2C implementation (continued)

| I2C features ⁽¹⁾ | I2C1 | I2C2 | I2C3 | I2C4 ⁽²⁾ |
|--|------|------|------|---------------------|
| Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s) | X | X | X | X |
| Independent clock | X | - | X | - |
| Wake-up from Stop 1 mode | X | - | X | - |
| Wake-up from Stop 2 mode | - | - | X | - |

1. X = supported.

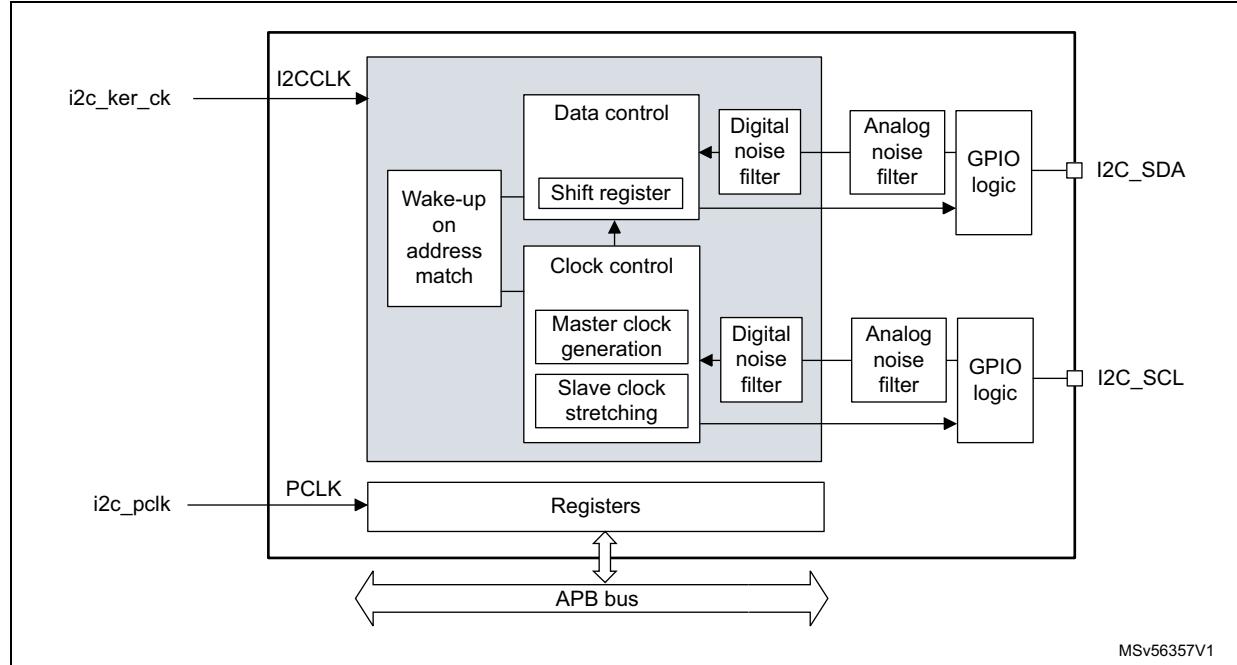
2. I2C4 is only available on STM32U073xx and STM32U083xx devices.

32.4 I2C functional description

In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The independent clock function allows the I²C communication speed to be independent of the PCLK frequency.

32.4.1 I2C block diagram

Figure 299. Block diagram

MSv56357V1

32.4.2 I2C pins and internal signals

Table 184. I2C input/output pins

| Pin name | Signal type | Description |
|----------|---------------|----------------------------|
| I2C_SDA | Bidirectional | I ² C-bus data |
| I2C_SCL | Bidirectional | I ² C-bus clock |

Table 185. 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 194 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) |

32.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period t_{I2CCLK} must respect the following conditions:

$$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$$

$$t_{I2CCLK} < t_{HIGH}$$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is $DNF[3:0] \times t_{I2CCLK}$.

The PCLK clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3 t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by PCLK, this clock must respect the conditions for t_{I2CCLK} .

32.4.4 I2C mode selection

The peripheral can operate as:

- slave transmitter
- slave receiver
- master transmitter
- master receiver

By default, the peripheral operates in slave mode. It automatically switches from slave to master mode upon generating START condition, and from master to slave mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multimaster I²C-bus environment.

Communication flow

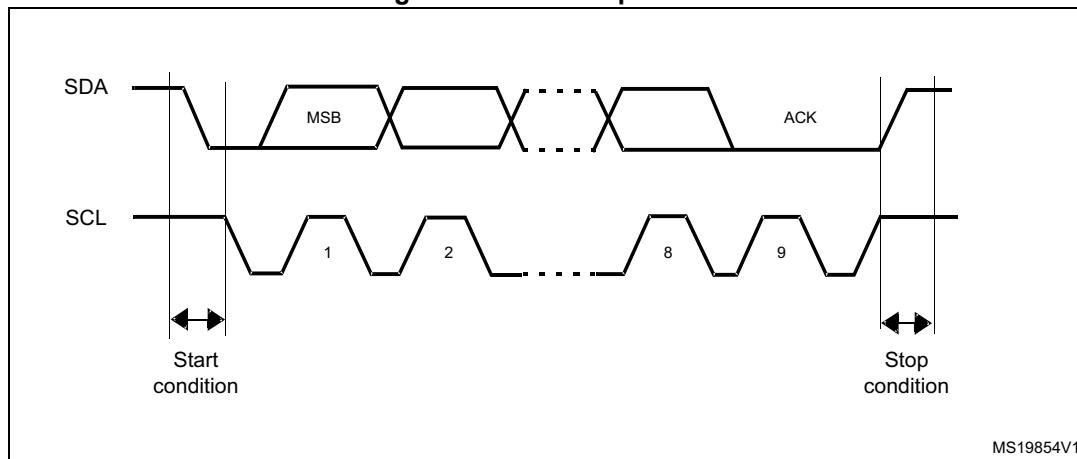
In master mode, the I²C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In slave mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first one (7-bit addressing) or two (10-bit addressing) bytes following the START condition contain the address. The address is always transmitted in master mode.

The following figure shows the transmission of a single byte. The master generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 300. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I²C peripheral can be selected by software.

32.4.5 I²C initialization

Enabling and disabling the peripheral

Before enabling the I²C peripheral, configure and enable its clock through the clock controller, and initialize its control registers.

The I²C peripheral can then be enabled by setting the PE bit of the I²C_CR1 register.

Disabling the I²C peripheral by clearing the PE bit resets the I²C peripheral. Refer to [Section 32.4.6](#) for more details.

Noise filters

Before enabling the I²C peripheral by setting the PE bit of the I²C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of I2CCLK. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen I2CCLK periods.

The following table compares the two filters.

Table 186. Comparison of analog and digital filters

| Item | Analog filter | Digital filter |
|-----------------------------------|---|---|
| Filtering capacity ⁽¹⁾ | ≥ 50 ns | One to fifteen I2CCLK periods |
| Benefits | Available in Stop mode | <ul style="list-style-type: none"> – Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity |
| Drawbacks | Filtering capacity variation with temperature, voltage, and silicon process | Wake-up from Stop mode on address match not supported when the digital filter is enabled |

1. Maximum duration of spikes that the filter can suppress

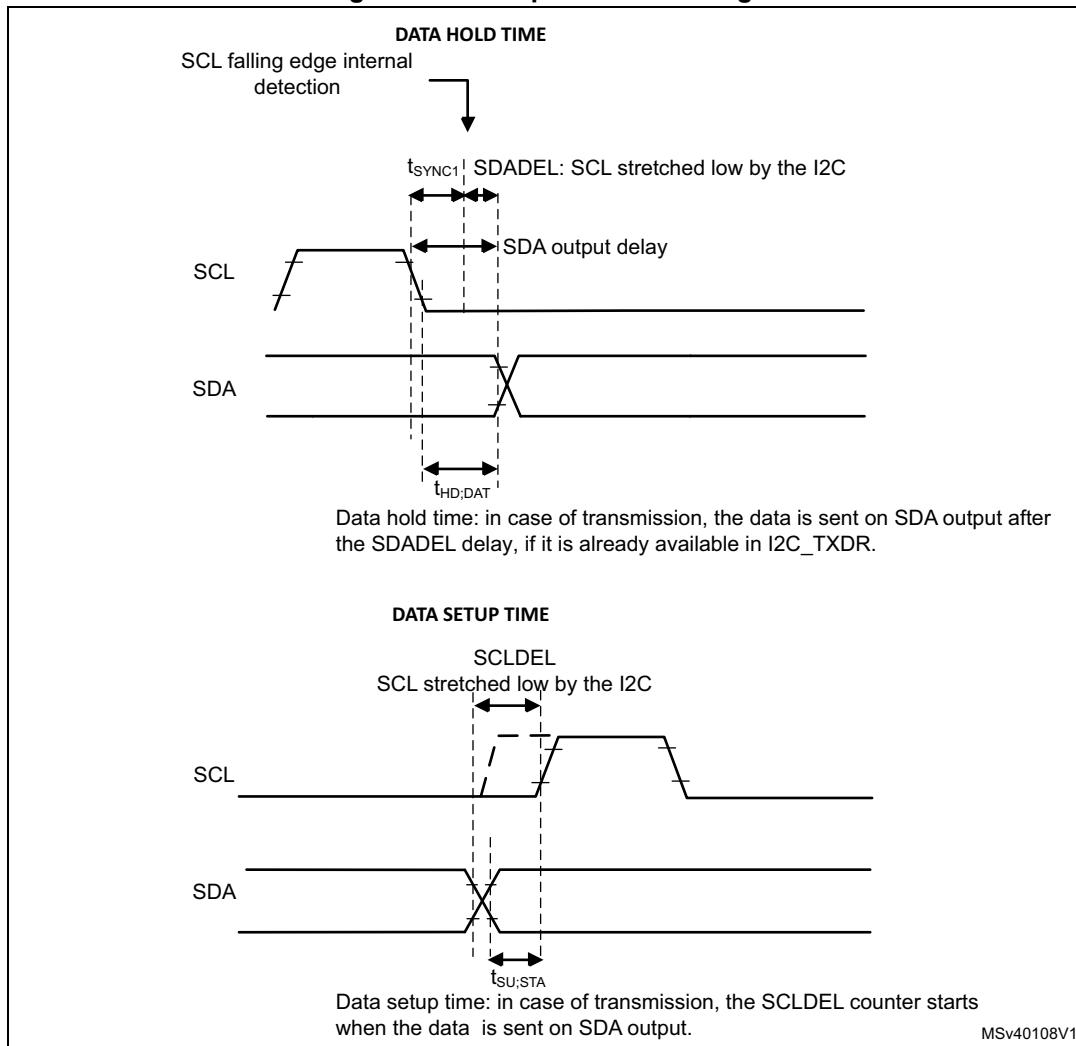
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 301. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

$$t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK}, \text{ where } t_{PRESC} = (\text{PRESC} + 1) \times t_{I2CCLK}.$$

The total SDA output delay is:

$$t_{SYNC1} + \{[SDADEL \times (\text{PRESC} + 1) + 1] \times t_{I2CCLK}\}$$

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay $t_{DNF} = DNF \times t_{I2CCLK}$ introduced by the digital filter (if enabled)
- 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 set SDADEL[3:0] so as to fulfill the following condition:

$$\{t_{f(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(\text{PRESC} + 1) \times t_{I2CCLK}\} \leq SDADEL$$

$$SDADEL \leq \{t_{HD;DAT(max)} - t_{AF(max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(\text{PRESC} + 1) \times t_{I2CCLK}\}$$

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 μs for Standard-mode, 0.9 μs for Fast-mode, and 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. When it stretches 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. The previous condition then 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 $SCLDEL[3:0]$ value.

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, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. 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 $SCLDEL[3:0]$ so as to fulfill the following condition:

$$\{[t_r(\max) + t_{SU;DAT}(\min)] / [(PRESC + 1) \times t_{I2CCLK}] - 1 \leq SCLDEL$$

Refer to the following table for t_f , t_r , $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed $SDADEL[3:0]$ and $SCLDEL[3:0]$ values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as master or slave stretches SCL low during at least $[(SDADEL + SCLDEL + 1) \times (PRESC + 1) + 1] \times t_{I2CCLK}$, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps 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.

When the NOSTRETCH bit is set in slave mode, the SCL is not stretched. The $SDADEL[3:0]$ must then be programmed so that it ensures a sufficient setup time.

Table 187. I²C-bus specification data setup and hold times

| Symbol | Parameter | Standard-mode (Sm) | | Fast-mode (Fm) | | Fast-mode Plus (Fm+) | | Unit |
|--------------|---------------------------------------|--------------------|------|----------------|-----|----------------------|------|---------|
| | | Min | Max | Min | Max | Min | Max | |
| $t_{HD;DAT}$ | Data hold time | 0 | - | 0 | - | 0 | - | μs |
| $t_{VD;DAT}$ | Data valid time | - | 3.45 | - | 0.9 | - | 0.45 | |
| $t_{SU;DAT}$ | Data setup time | 250 | - | 100 | - | 50 | - | ns |
| t_r | Rise time of both SDA and SCL signals | - | 1000 | - | 300 | - | 120 | |
| t_f | Fall time of both SDA and SCL signals | - | 300 | - | 300 | - | 120 | |

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] bitfields of the I2C_TIMINGR register.

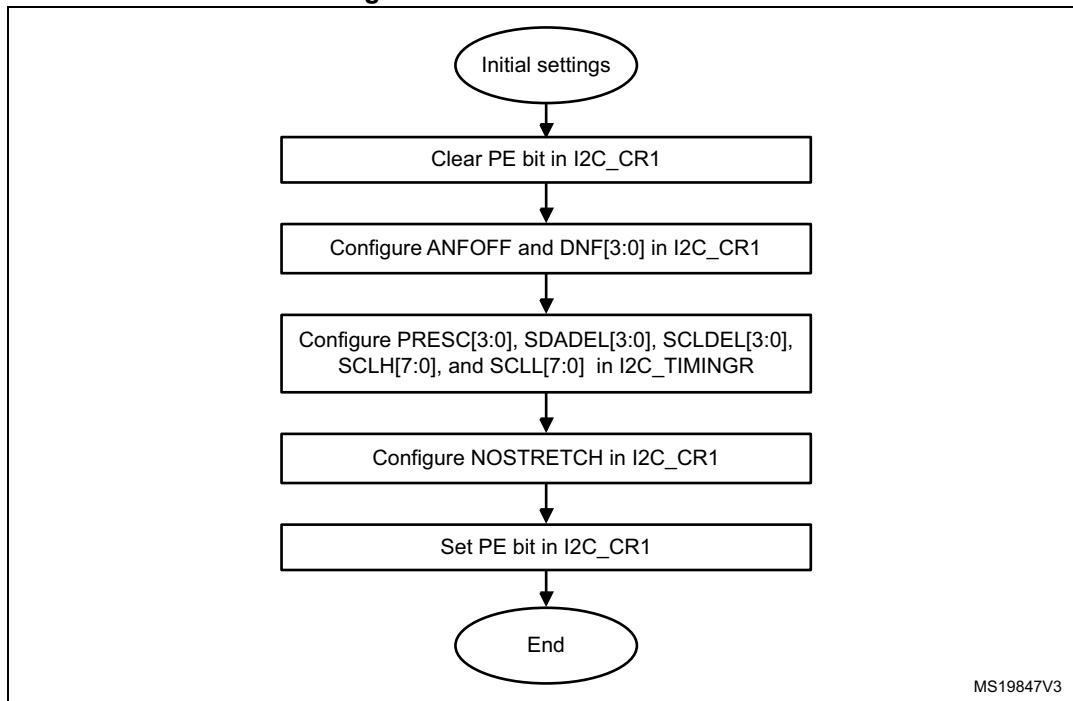
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay $t_{SCLL} = (SCLL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. The t_{SCLL} delay impacts the SCL low time t_{LOW} .

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay $t_{SCLH} = (SCLH + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. The t_{SCLH} impacts the SCL high time t_{HIGH} .

Refer to [I2C master initialization](#) for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the slave NOSTRETCH settings must also be done before enabling the peripheral. Refer to [I2C slave initialization](#) for more details.

Figure 302. I2C initialization flow



32.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

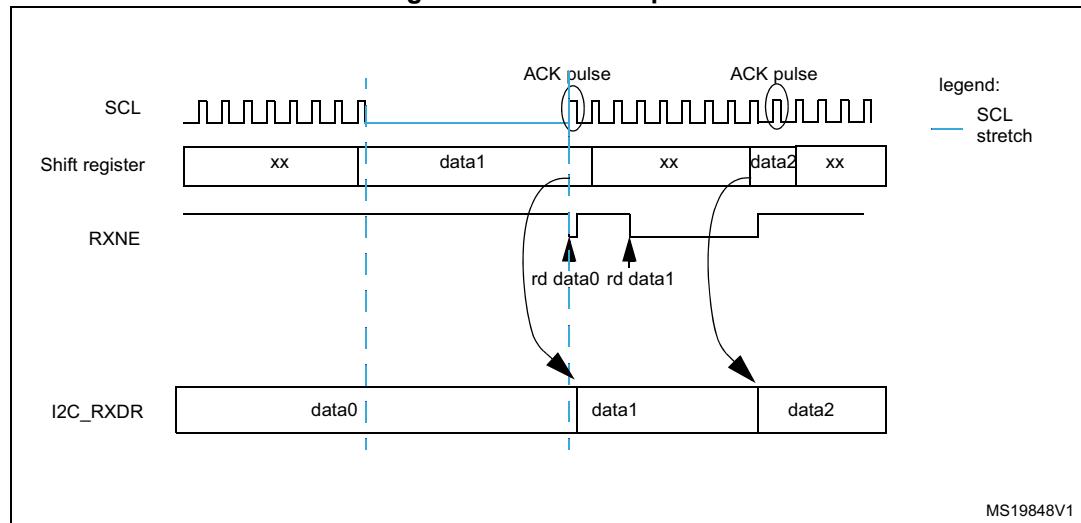
32.4.7 I2C data transfer

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 the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

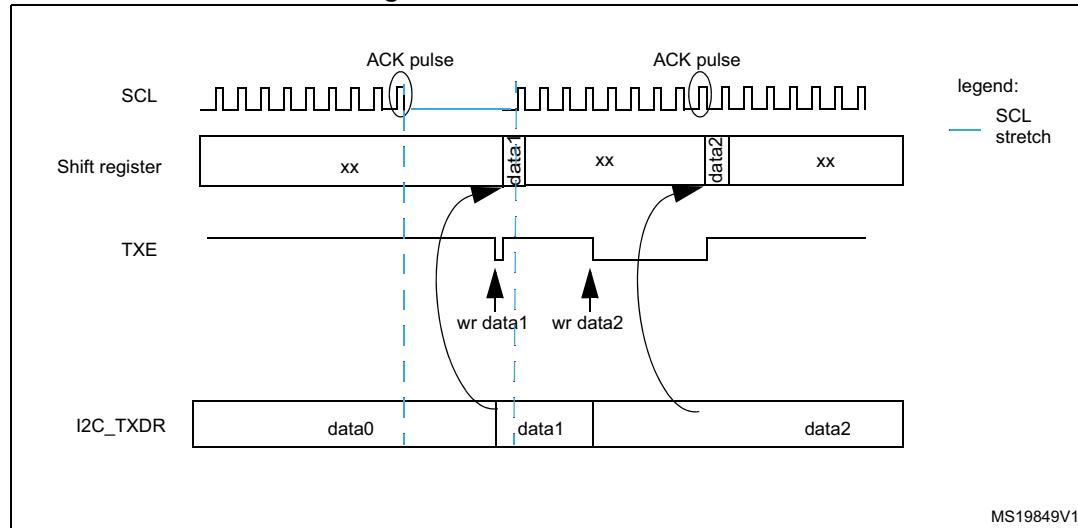
Figure 303. Data reception



Transmission

If the I2C_TXDR register is not empty ($\text{TXE} = 0$), 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 the SDA line. If $\text{TXE} = 1$, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 304. 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

In master mode, the byte counter is always used. By default, it is disabled in slave mode. It can be enabled by software, by setting the SBC (slave byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number 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 of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the 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] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield 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 of the I2C_CR2 register is set. 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 188. 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 |

32.4.8 I2C slave mode

I2C slave initialization

To work in slave mode, the user must enable at least one slave address. The I2C_OAR1 and I2C_OAR2 registers are available to program the slave own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional slave addresses are required, the second slave address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] 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. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral 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 required, to perform software actions. If the master does not

support clock stretching, 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] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Slave with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus slave stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled slave addresses.
The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
The stretch is released when I2C_RXDR is read.
- In slave byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Slave without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus 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 of the I2C_CR1 register is set. 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 of the I2C_CR1 register is set.

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 of the I2C_CR1 register.

The reload mode must be selected to allow byte ACK control in slave reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] 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 the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of 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.

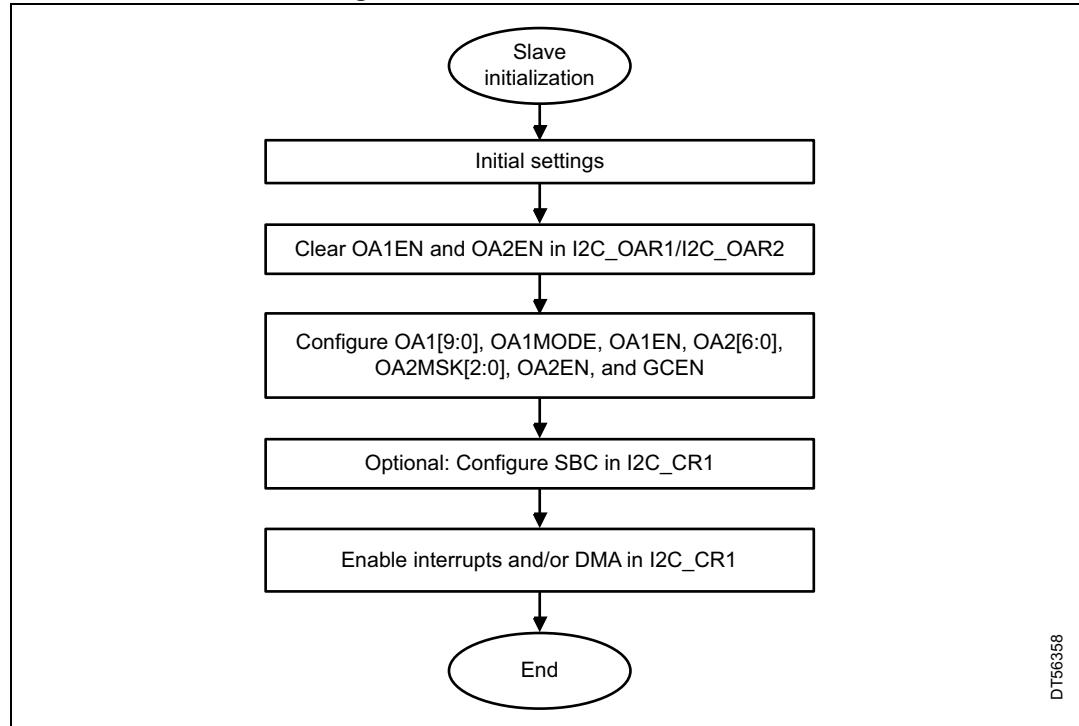
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: *The SBC bit must be configured when the I2C peripheral is disabled, 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 305. Slave initialization flow



Slave transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. 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 STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set 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.

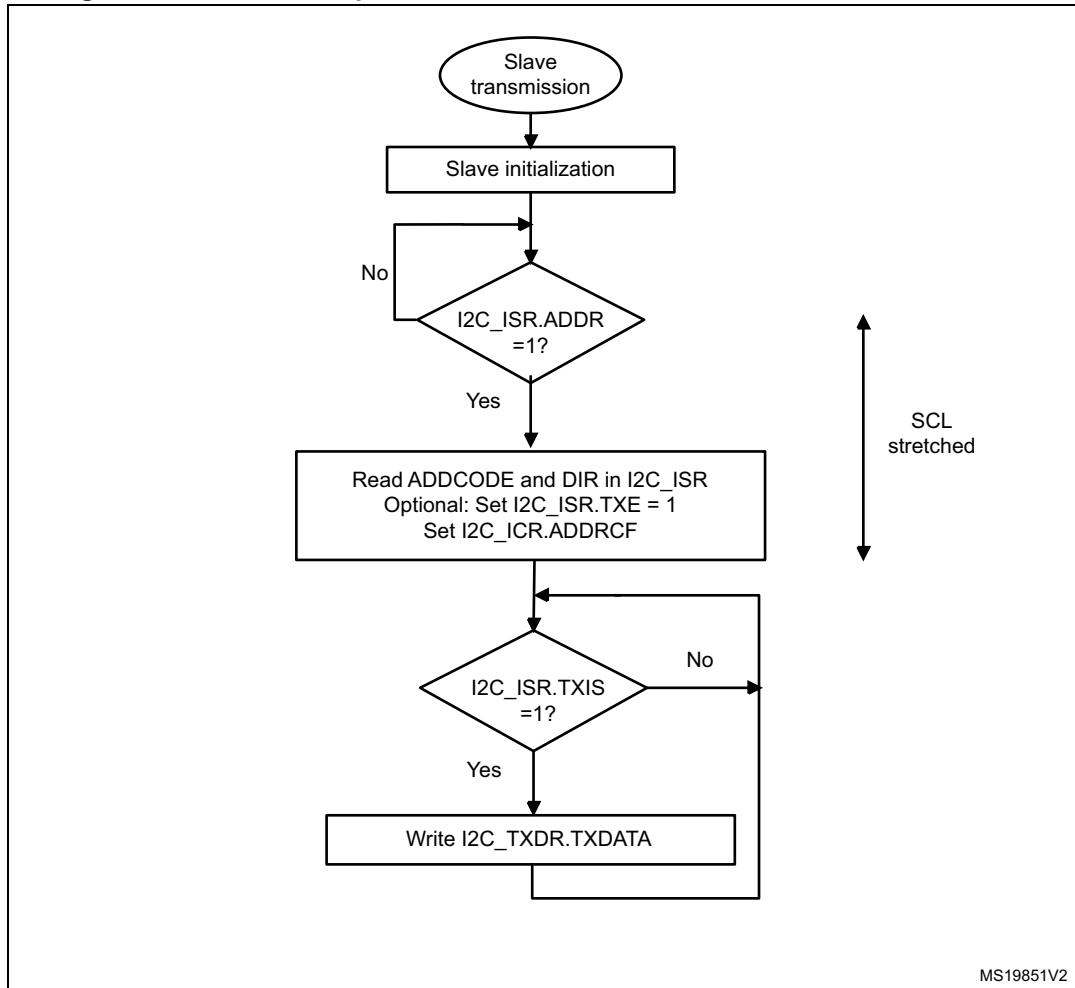
In slave byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

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 send 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 send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees 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 required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 306. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0

MS19851V2

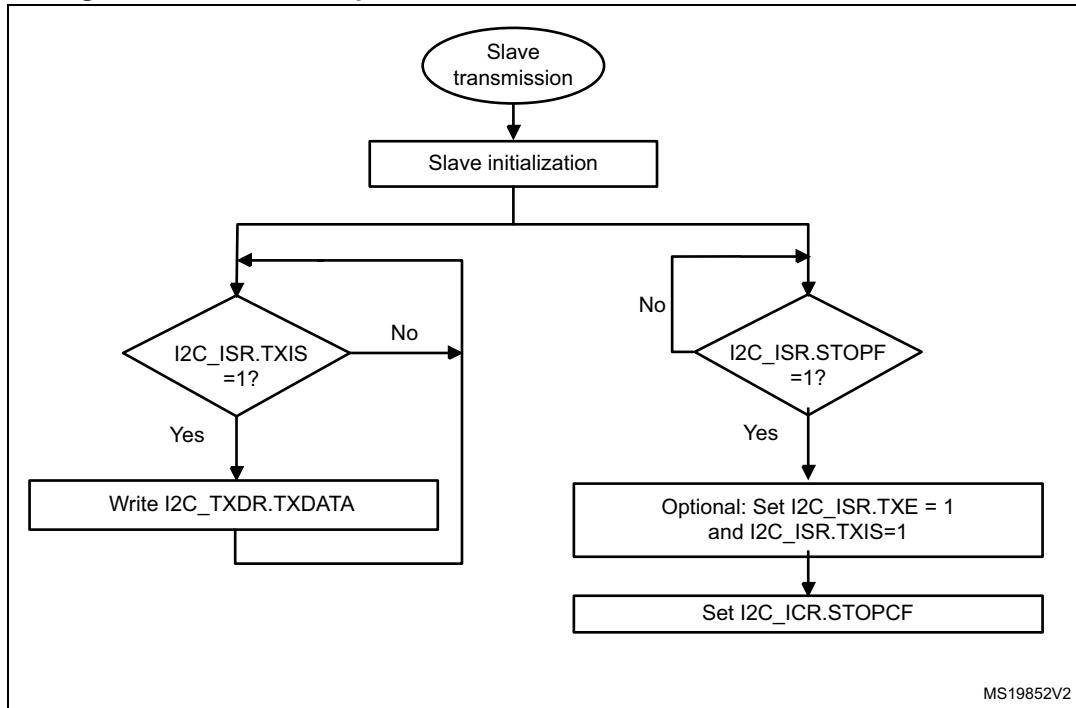
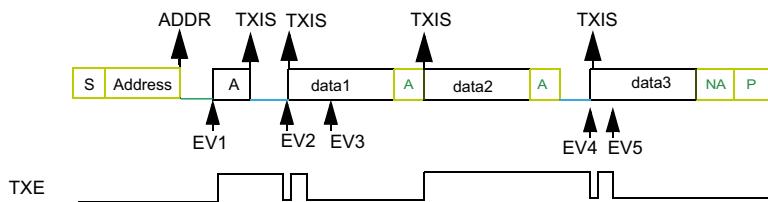
Figure 307. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1

Figure 308. Transfer bus diagrams for I2C slave transmitter (mandatory events only)

Example I2C slave transmitter 3 bytes with 1st data flushed,
NOSTRETCH=0:



legend:

- transmission
- reception
- SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF

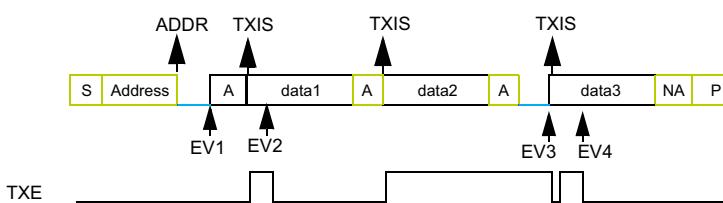
EV2: TXIS ISR: wr data1

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:



legend :

- transmission
- reception
- SCL stretch

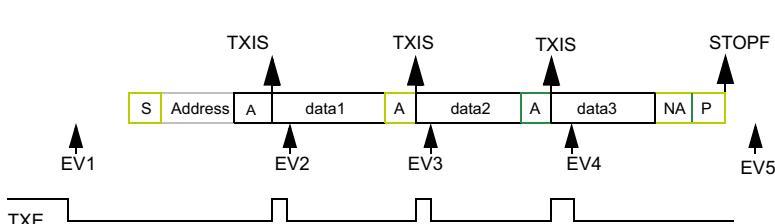
EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF

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:



legend:

- transmission
- reception
- SCL stretch

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

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 309. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 0

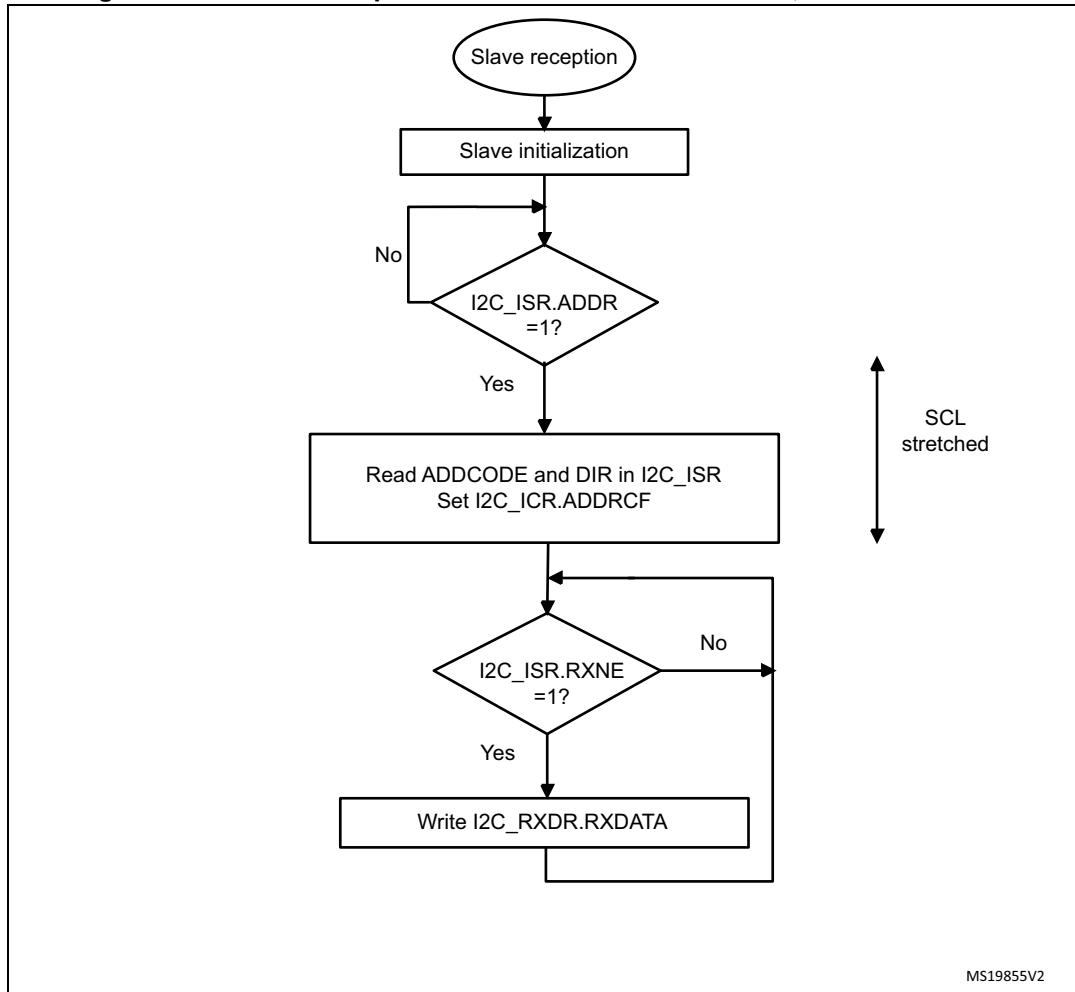
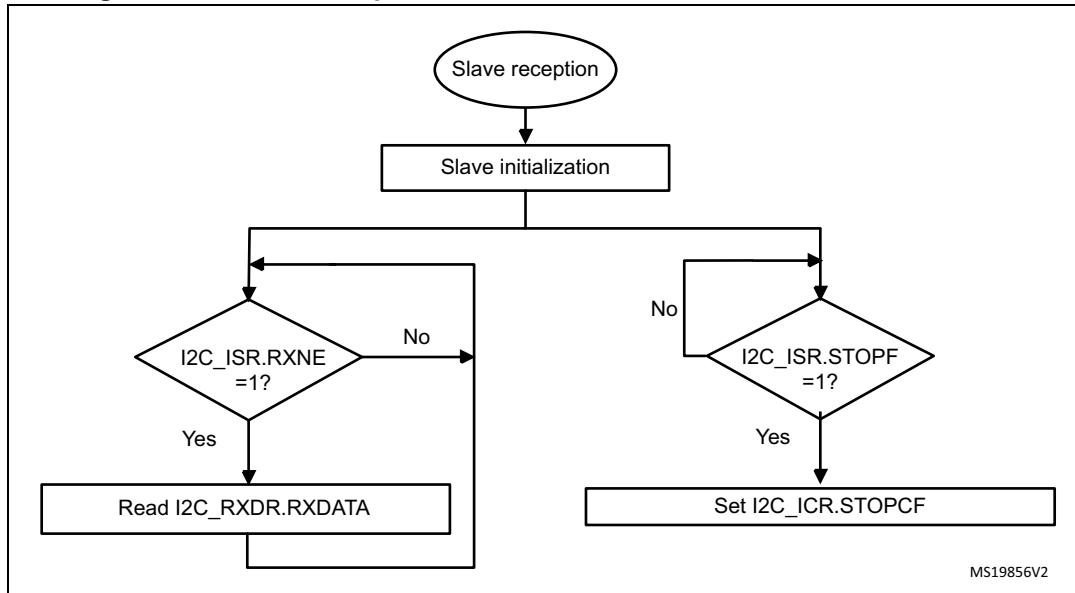
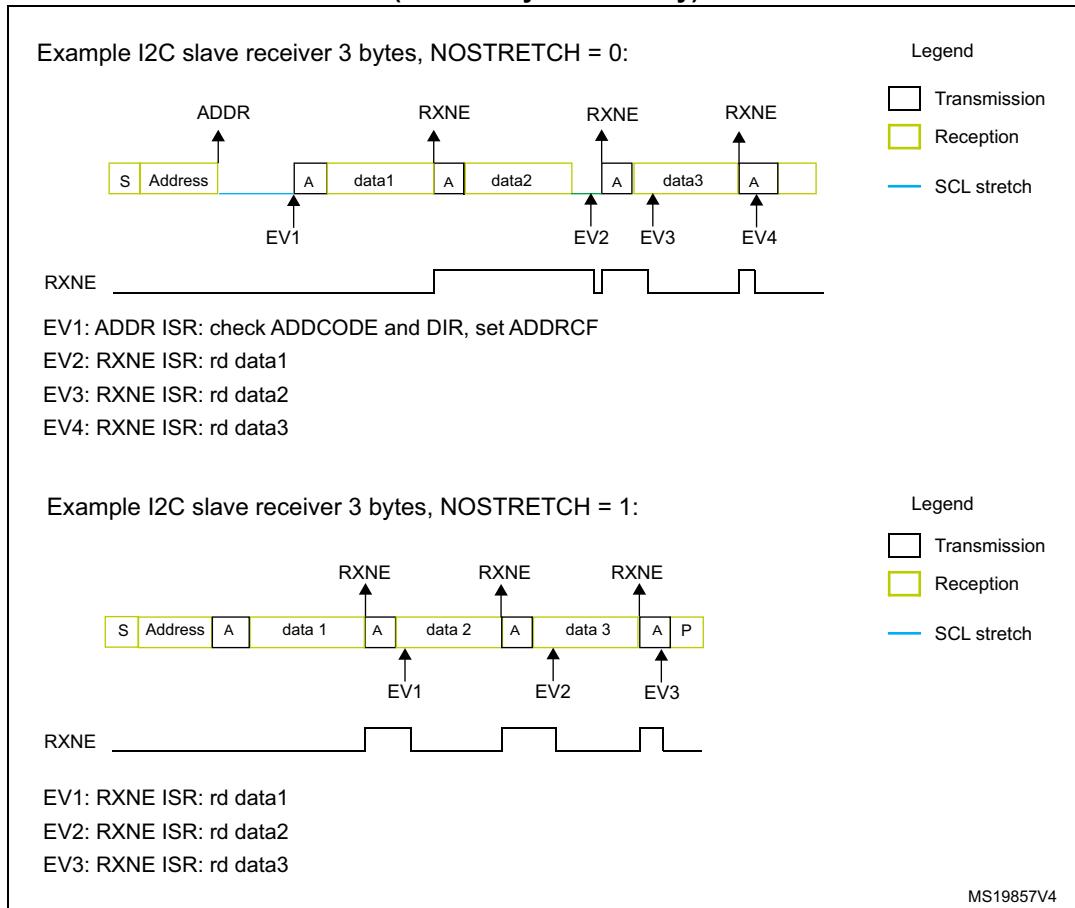


Figure 310. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 1**Figure 311. Transfer bus diagrams for I2C slave receiver (mandatory events only)**

32.4.9 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.

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. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the master clock period is:

$$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$$

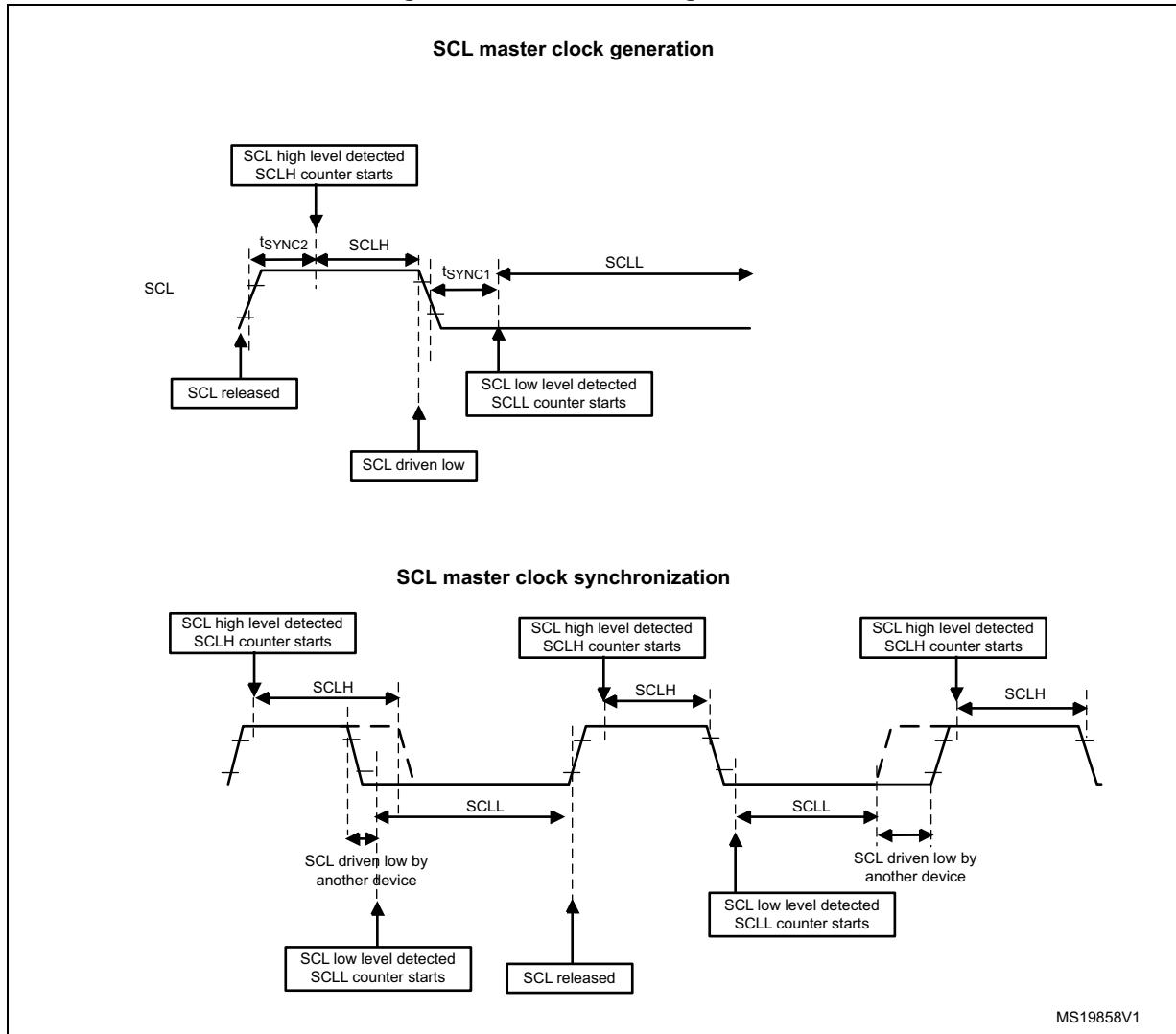
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] $\times t_{I2CCLK}$
- delay due to SCL synchronization with the I2CCLK clock (two to three I2CCLK periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] $\times t_{I2CCLK}$
- delay due to SCL synchronization with the I2CCLK clock (two to three I2CCLK periods)

Figure 312. Master clock generation



Caution: For compliance with the I²C-bus specification, the master clock must respect the timings in the following table.

Table 189. I²C-bus specification clock timings

| Symbol | Parameter | Standard-mode (Sm) | | Fast-mode (Fm) | | Fast-mode Plus (Fm+) | | Unit |
|---------------------|--|--------------------|------|----------------|-----|----------------------|------|---------|
| | | Min | Max | Min | Max | Min | Max | |
| f _{SCL} | SCL clock frequency | - | 100 | - | 400 | - | 1000 | kHz |
| t _{HD:STA} | Hold time (repeated) START condition | 4.0 | - | 0.6 | - | 0.26 | - | μ s |
| t _{SU:STA} | Set-up time for a repeated START condition | 4.7 | - | 0.6 | - | 0.26 | - | |
| t _{SU:STO} | Set-up time for STOP condition | 4.0 | - | 0.6 | - | 0.26 | - | |
| t _{BUF} | Bus free time between a STOP and START condition | 4.7 | - | 1.3 | - | 0.5 | - | |
| t _{LOW} | Low period of the SCL clock | 4.7 | - | 1.3 | - | 0.5 | - | |
| t _{HIGH} | High period of the SCL clock | 4.0 | - | 0.6 | - | 0.26 | - | |
| t _r | Rise time of both SDA and SCL signals | - | 1000 | - | 300 | - | 120 | ns |
| t _f | Fall time of both SDA and SCL signals | - | 300 | - | 300 | - | 120 | |

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and t_{SU:STA} timings and SCLH[7:0] the t_{HD:STA} and t_{SU:STO} timings.

Refer to [Section 32.4.10](#) for examples of I²C_TIMINGR settings versus the I²CCLK frequency.

Master communication initialization (address phase)

To initiate the communication with a slave to address, set the following bitfields of the I²C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: slave address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I²C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I²C peripheral in case of a deadlock, by toggling the PE bit of the I²C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The master then automatically sends a START condition followed by the slave address as soon as it detects that the bus is free ($BUSY = 0$) and after the t_{BUF} delay from a previous STOP condition expires.

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 upon arbitration loss.*

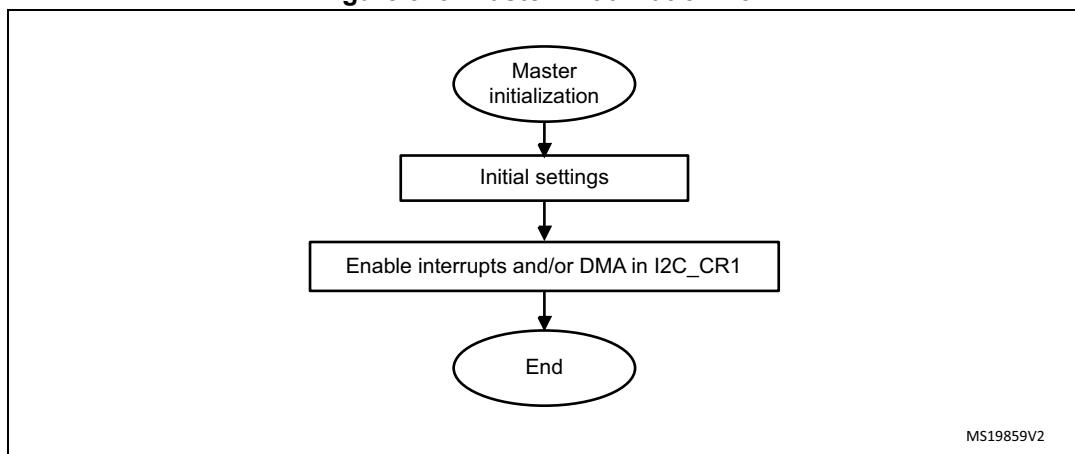
In 10-bit addressing mode, the master automatically keeps resending the slave address in a loop until the first address byte (first seven address bits) is acknowledged by the slave.

Setting the ADDRCF bit makes I2C quit that loop.

If the I2C peripheral is addressed as a slave (ADDR = 1) while the START bit is set, the I2C peripheral switches to slave mode and the START bit is cleared.

Note: *The same procedure is applied for a repeated START condition. In this case, BUSY = 1.*

Figure 313. Master initialization flow

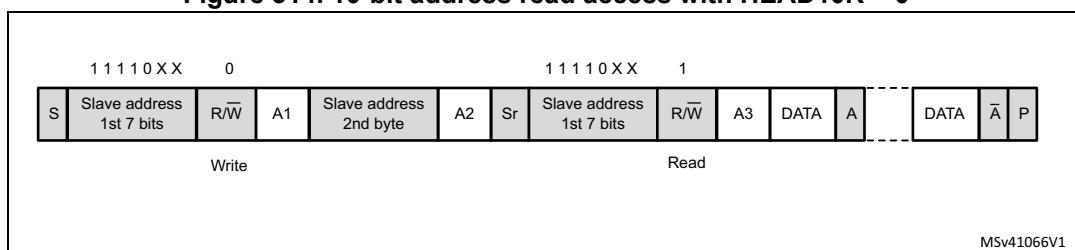


Initialization of a master receiver addressing a 10-bit address slave

If the slave address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of 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.

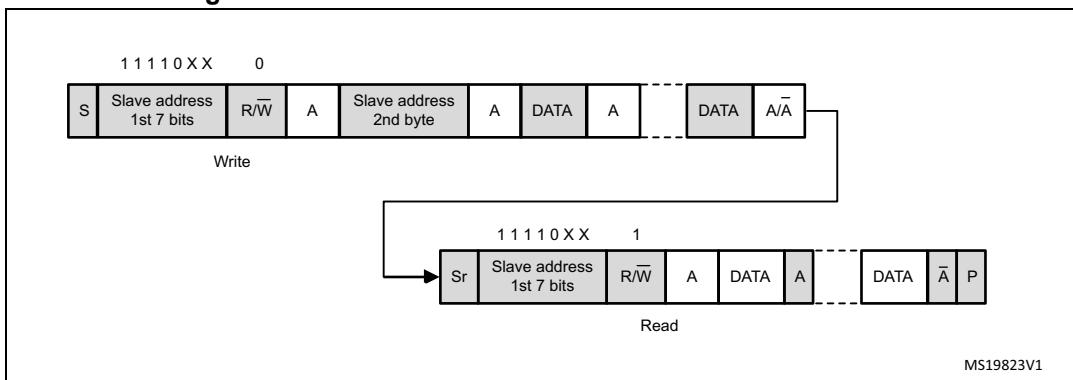
Figure 314. 10-bit address read access with HEAD10R = 0



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 315. 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 of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

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 transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition 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 of the I2C_CR2 register with the proper slave address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
 - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. the NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

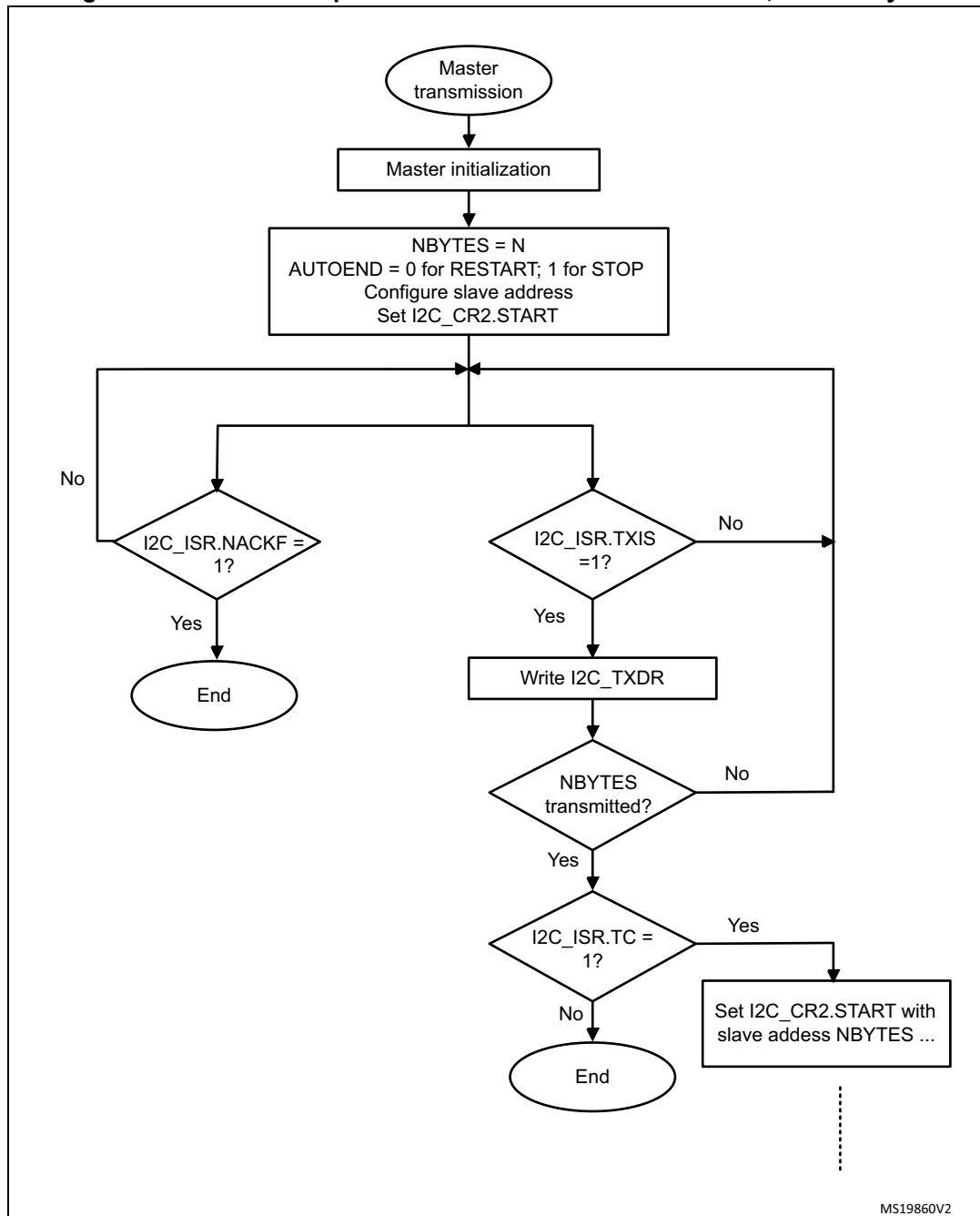
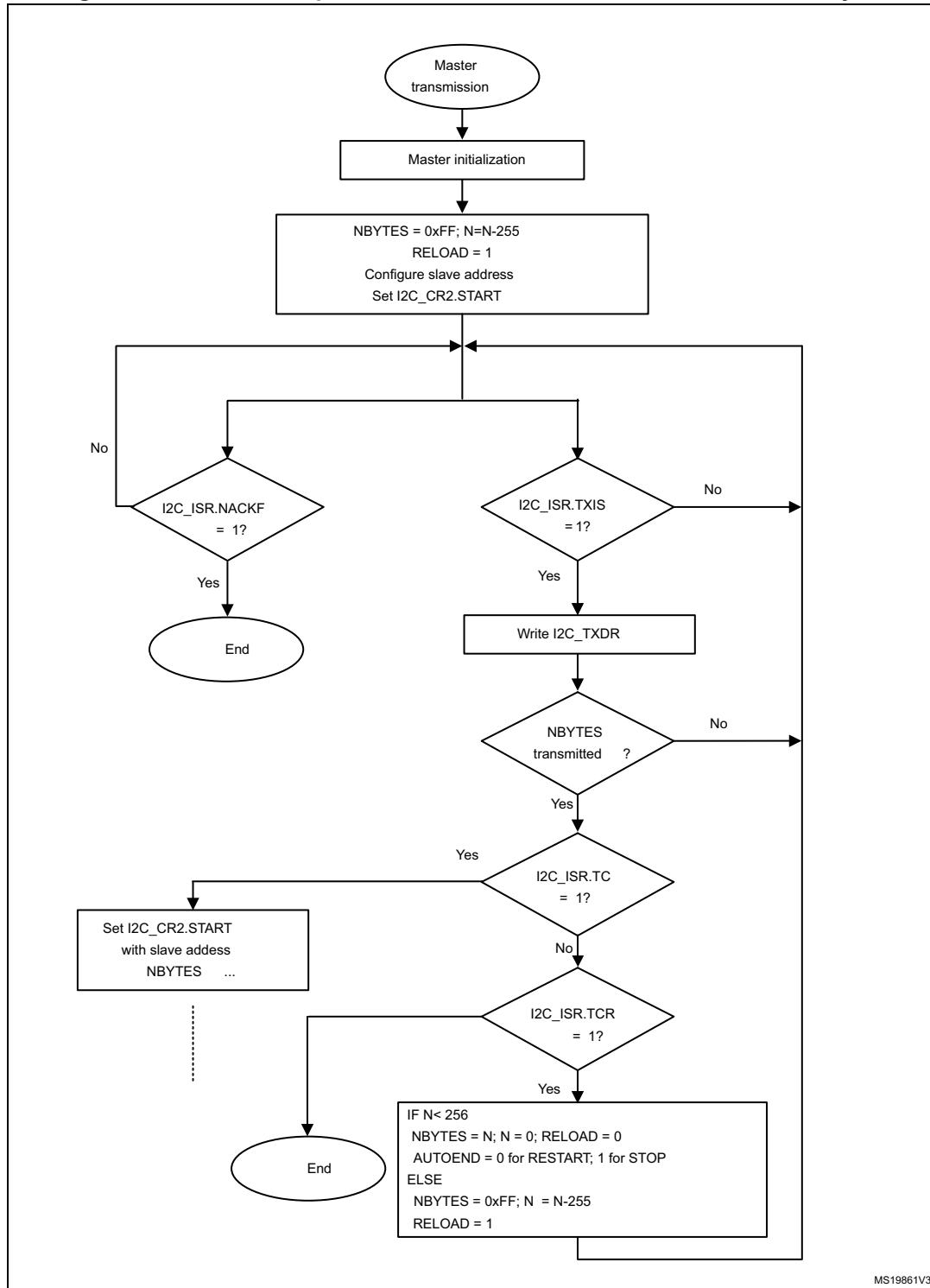
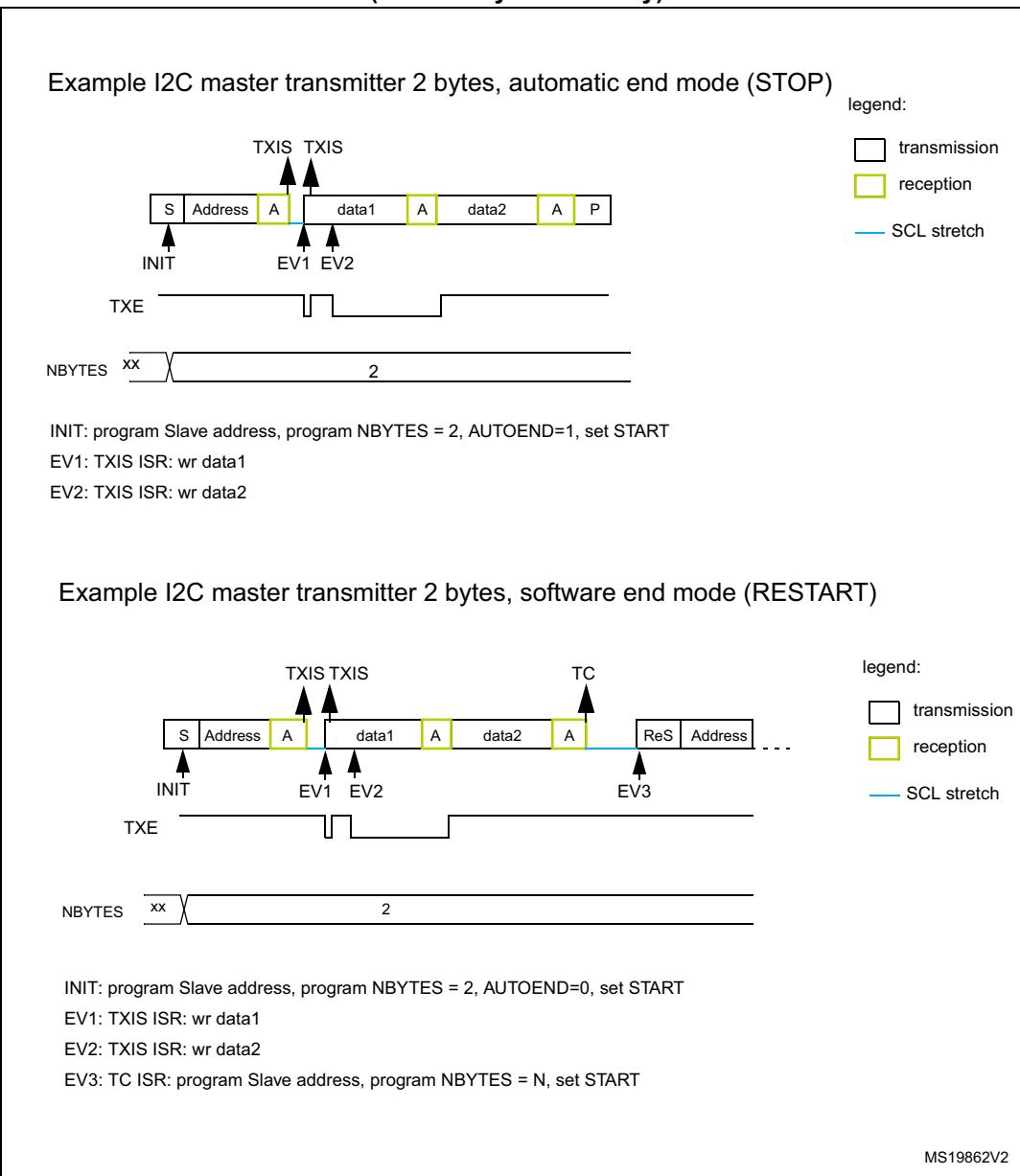
Figure 316. Transfer sequence flow for I2C master transmitter, N ≤ 255 bytes

Figure 317. Transfer sequence flow for I2C master transmitter, N > 255 bytes



MS19861V3

**Figure 318. Transfer bus diagrams for I2C master transmitter
(mandatory events only)**



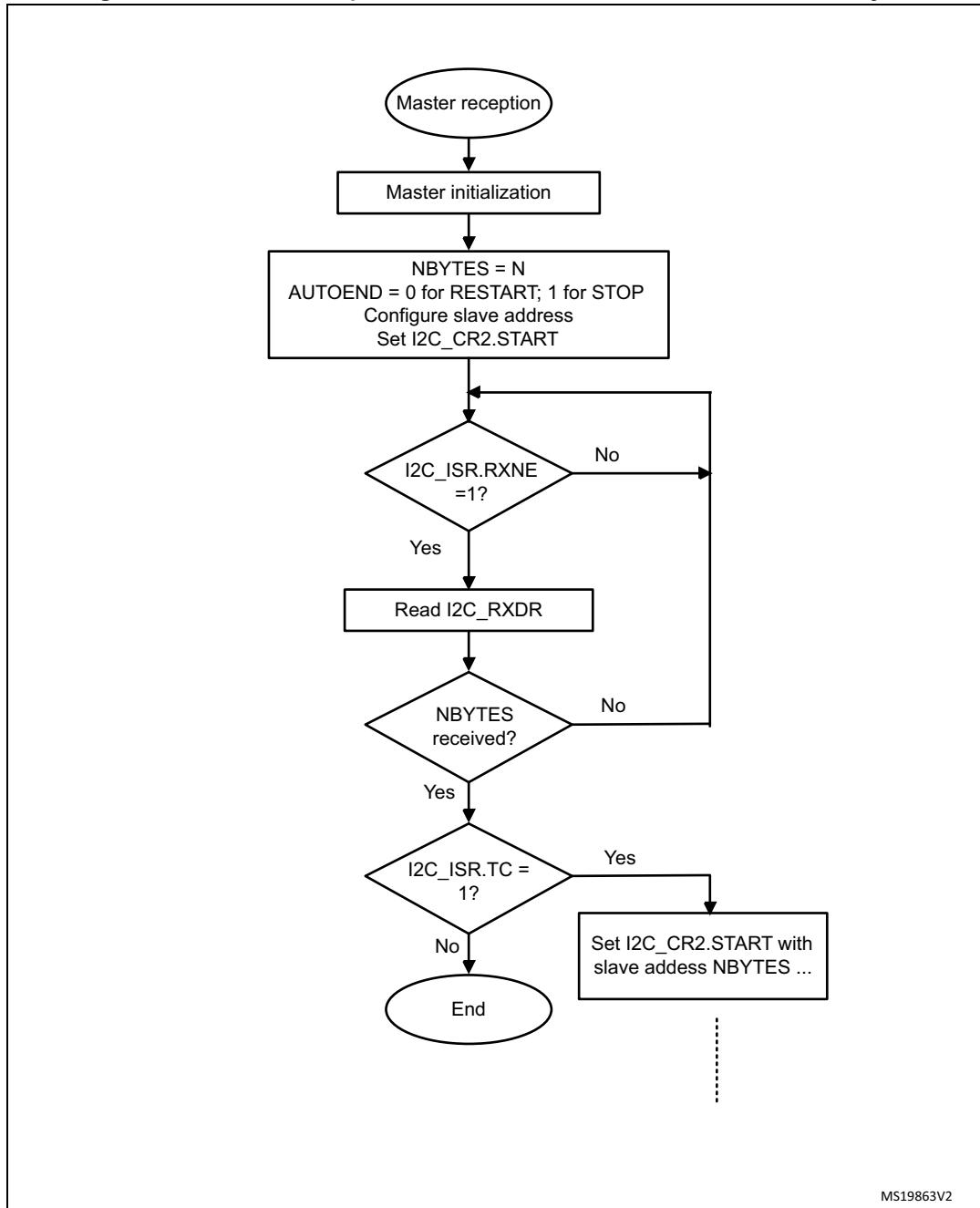
Master receiver

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 of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

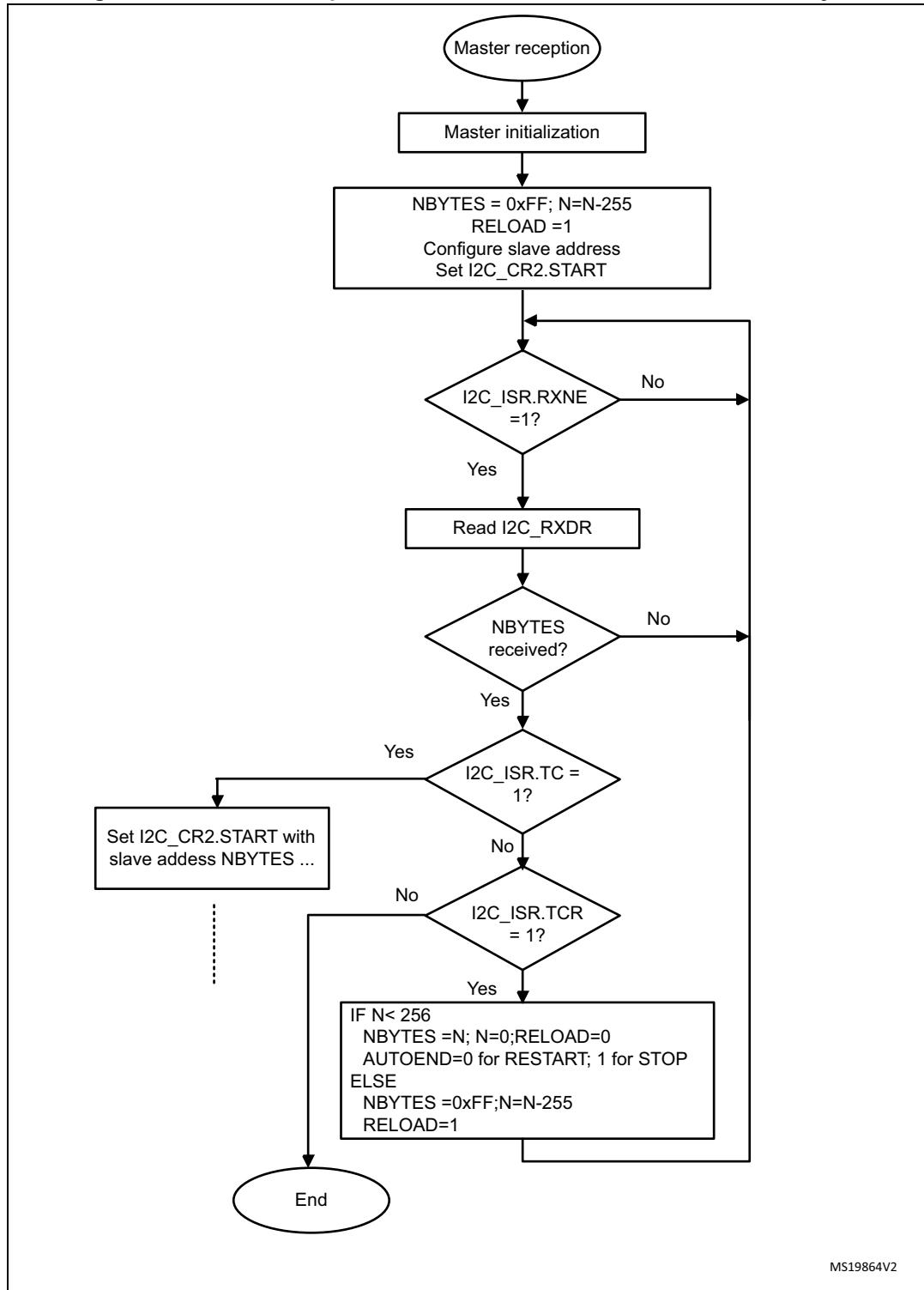
When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- 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 of the I2C_CR2 register, with the proper slave address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the slave address on the bus.
 - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 319. Transfer sequence flow for I2C master receiver, $N \leq 255$ bytes

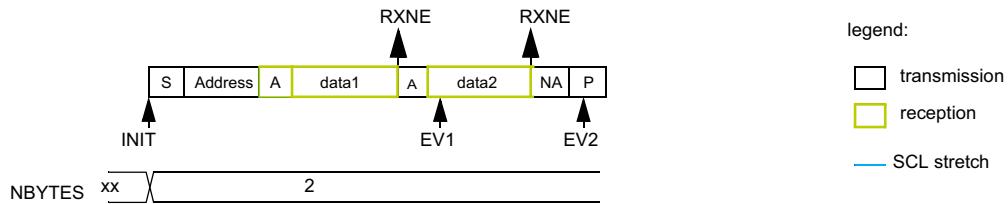
MS19863V2

Figure 320. Transfer sequence flow for I2C master receiver, N > 255 bytes



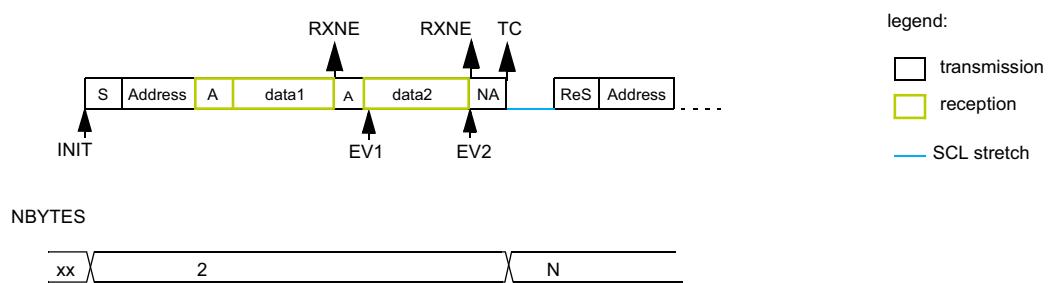
**Figure 321. Transfer bus diagrams for I2C master receiver
(mandatory events only)**

Example I2C master receiver 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2

Example I2C master receiver 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: read data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19865V1

32.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 190. Timing settings for f_{I2CCLK} of 8 MHz

| Parameter | Standard-mode (Sm) | | Fast-mode (Fm) | Fast-mode Plus (Fm+) |
|-----------------|--|--|--|--|
| | 10 kHz | 100 kHz | 400 kHz | 500 kHz |
| PRESC[3:0] | 0x1 | 0x1 | 0x0 | 0x0 |
| SCLL[7:0] | 0xC7 | 0x13 | 0x9 | 0x6 |
| t_{SCLL} | $200 \times 250 \text{ ns} = 50 \mu\text{s}$ | $20 \times 250 \text{ ns} = 5.0 \mu\text{s}$ | $10 \times 125 \text{ ns} = 1250 \text{ ns}$ | $7 \times 125 \text{ ns} = 875 \text{ ns}$ |
| SCLH[7:0] | 0xC3 | 0xF | 0x3 | 0x3 |
| t_{SCLH} | $196 \times 250 \text{ ns} = 49 \mu\text{s}$ | $16 \times 250 \text{ ns} = 4.0 \mu\text{s}$ | $4 \times 125 \text{ ns} = 500 \text{ ns}$ | $4 \times 125 \text{ ns} = 500 \text{ ns}$ |
| $t_{SCL}^{(1)}$ | $\sim 100 \mu\text{s}^{(2)}$ | $\sim 10 \mu\text{s}^{(2)}$ | $\sim 2.5 \mu\text{s}^{(3)}$ | $\sim 2.0 \mu\text{s}^{(4)}$ |
| SDADEL[3:0] | 0x2 | 0x2 | 0x1 | 0x0 |
| t_{SDADEL} | $2 \times 250 \text{ ns} = 500 \text{ ns}$ | $2 \times 250 \text{ ns} = 500 \text{ ns}$ | $1 \times 125 \text{ ns} = 125 \text{ ns}$ | 0 ns |
| SCLDEL[3:0] | 0x4 | 0x4 | 0x3 | 0x1 |
| t_{SCLDEL} | $5 \times 250 \text{ ns} = 1250 \text{ ns}$ | $5 \times 250 \text{ ns} = 1250 \text{ ns}$ | $4 \times 125 \text{ ns} = 500 \text{ ns}$ | $2 \times 125 \text{ ns} = 250 \text{ ns}$ |

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 $4 \times t_{I2CCLK} = 500 \text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 1000 \text{ ns}$.
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 500 \text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 750 \text{ ns}$.
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 500 \text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 655 \text{ ns}$.

Table 191. Timing settings for f_{I2CCLK} of 16 MHz

| Parameter | Standard-mode (Sm) | | Fast-mode (Fm) | Fast-mode Plus (Fm+) |
|-----------------|--|--|--|---|
| | 10 kHz | 100 kHz | 400 kHz | 1000 kHz |
| PRESC[3:0] | 0x3 | 0x3 | 0x1 | 0x0 |
| SCLL[7:0] | 0xC7 | 0x13 | 0x9 | 0x4 |
| t_{SCLL} | $200 \times 250 \text{ ns} = 50 \mu\text{s}$ | $20 \times 250 \text{ ns} = 5.0 \mu\text{s}$ | $10 \times 125 \text{ ns} = 1250 \text{ ns}$ | $5 \times 62.5 \text{ ns} = 312.5 \text{ ns}$ |
| SCLH[7:0] | 0xC3 | 0xF | 0x3 | 0x2 |
| t_{SCLH} | $196 \times 250 \text{ ns} = 49 \mu\text{s}$ | $16 \times 250 \text{ ns} = 4.0 \mu\text{s}$ | $4 \times 125 \text{ ns} = 500 \text{ ns}$ | $3 \times 62.5 \text{ ns} = 187.5 \text{ ns}$ |
| $t_{SCL}^{(1)}$ | $\sim 100 \mu\text{s}^{(2)}$ | $\sim 10 \mu\text{s}^{(2)}$ | $\sim 2.5 \mu\text{s}^{(3)}$ | $\sim 1.0 \mu\text{s}^{(4)}$ |
| SDADEL[3:0] | 0x2 | 0x2 | 0x2 | 0x0 |
| t_{SDADEL} | $2 \times 250 \text{ ns} = 500 \text{ ns}$ | $2 \times 250 \text{ ns} = 500 \text{ ns}$ | $2 \times 125 \text{ ns} = 250 \text{ ns}$ | 0 ns |
| SCLDEL[3:0] | 0x4 | 0x4 | 0x3 | 0x2 |
| t_{SCLDEL} | $5 \times 250 \text{ ns} = 1250 \text{ ns}$ | $5 \times 250 \text{ ns} = 1250 \text{ ns}$ | $4 \times 125 \text{ ns} = 500 \text{ ns}$ | $3 \times 62.5 \text{ ns} = 187.5 \text{ ns}$ |

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 $4 \times t_{I2CCLK} = 250 \text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 1000 \text{ ns}$.
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 250 \text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 750 \text{ ns}$.
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 250 \text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 500 \text{ ns}$.

Table 192. Timing settings for f_{I2CCLK} of 48 MHz

| Parameter | Standard-mode (Sm) | | Fast-mode (Fm) | Fast-mode Plus (Fm+) |
|-----------------|--|--|--|--|
| | 10 kHz | 100 kHz | 400 kHz | 1000 kHz |
| PRESC[3:0] | 0xB | 0xB | 0x5 | 0x5 |
| SCLL[7:0] | 0xC7 | 0x13 | 0x9 | 0x3 |
| t_{SCLL} | $200 \times 250 \text{ ns} = 50 \mu\text{s}$ | $20 \times 250 \text{ ns} = 5.0 \mu\text{s}$ | $10 \times 125 \text{ ns} = 1250 \text{ ns}$ | $4 \times 125 \text{ ns} = 500 \text{ ns}$ |
| SCLH[7:0] | 0xC3 | 0xF | 0x3 | 0x1 |
| t_{SCLH} | $196 \times 250 \text{ ns} = 49 \mu\text{s}$ | $16 \times 250 \text{ ns} = 4.0 \mu\text{s}$ | $4 \times 125 \text{ ns} = 500 \text{ ns}$ | $2 \times 125 \text{ ns} = 250 \text{ ns}$ |
| $t_{SCL}^{(1)}$ | $\sim 100 \mu\text{s}^{(2)}$ | $\sim 10 \mu\text{s}^{(2)}$ | $\sim 2.5 \mu\text{s}^{(3)}$ | $\sim 875 \text{ ns}^{(4)}$ |
| SDADEL[3:0] | 0x2 | 0x2 | 0x3 | 0x0 |
| t_{SDADEL} | $2 \times 250 \text{ ns} = 500 \text{ ns}$ | $2 \times 250 \text{ ns} = 500 \text{ ns}$ | $3 \times 125 \text{ ns} = 375 \text{ ns}$ | 0 ns |
| SCLDEL[3:0] | 0x4 | 0x4 | 0x3 | 0x1 |
| t_{SCLDEL} | $5 \times 250 \text{ ns} = 1250 \text{ ns}$ | $5 \times 250 \text{ ns} = 1250 \text{ ns}$ | $4 \times 125 \text{ ns} = 500 \text{ ns}$ | $2 \times 125 \text{ ns} = 250 \text{ ns}$ |

1. t_{SCL} is greater than $t_{SCLL} + t_{SCLH}$ due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.

2. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 83.3 \text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 1000 \text{ ns}$

3. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 83.3 \text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 750 \text{ ns}$

4. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 83.3 \text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 250 \text{ ns}$

32.4.11 Wake-up from Stop mode on address match

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI16 oscillator must be selected as the clock source for I2CCLK to allow the wake-up from Stop mode.

In Stop mode, the HSI16 oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI16 oscillator and stretches SCL low until the oscillator wakes up.

HSI16 is then used for the address reception.

If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI16 oscillator is stopped again and the device does not wake up.

Note: When the system clock is used as I2C clock, or when WUPEN = 0, the HSI16 oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a master or as an addressed slave after the ADDR flag is set. This can be managed by clearing the 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. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

- Caution:** The feature is only available when the HSI16 oscillator is selected as the I2C clock.
- Caution:** Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.
- Caution:** If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

32.4.12 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as master or addressed slave (that is, not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (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 of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

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 of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

32.5 I2C in low-power modes

Table 193. Effect of low-power modes to I2C

| Mode | Description |
|------------------------|--|
| Sleep | No effect. I2C interrupts cause the device to exit the Sleep mode. |
| Stop ⁽¹⁾⁽²⁾ | The contents of I2C registers are kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI16). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – WUPEN = 0: the I2C must be disabled before entering Stop mode. |
| Standby | The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode. |

- Refer to [Section 32.3: I2C implementation](#) for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.
- Only I2C3 supports wake-up from Stop 2 mode. The other I2C instances must be disabled before entering Stop 2 mode.

32.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 194. 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 | | Yes ⁽¹⁾ | |
| | Address matched | ADDR | ADDRIE | Write ADDRCF = 1 | | | |
| | NACK reception | NACKF | NACKIE | Write NACKCF = 1 | | | |
| I2C_ER | Bus error | BERR | ERRIE | Write BERRCF = 1 | Yes | No | No |
| | Arbitration loss | ARLO | | Write ARLOCF = 1 | | | |
| | Overrun/underrun | OVR | | Write OVRCF = 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 32.3: I2C implementation](#).

32.7 I2C DMA requests

32.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see [Section 9: Direct memory access controller \(DMA\)](#)) 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, 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, 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.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

32.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to [Section 9: Direct memory access controller \(DMA\)](#)) whenever the RXNE 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. When all data are transferred using DMA, 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, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

32.8 I2C registers

Refer to [Section 1.2](#) for the list of abbreviations used in register descriptions.

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

32.8.1 I2C control register 1 (I2C_CR1)

Address offset: 0x000

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 PCLK + 6 x I2CCLK.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---------------|--------------|------|------------|----------|------|------|-----|-------|------|------------|------------|------------|-----------|---------------|-----|
| STOPF ACLR | ADDRA CLR | Res. | Res. | Res. | Res. | Res. | FMP | Res. | Res. | Res. | Res. | GC EN | WUP EN | NO STRETCH | SBC |
| 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 |

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear

0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.

1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH slave mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a slave data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear

0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.

1: ADDR flag remains cleared by hardware. This mode can be used in slave mode, to avoid the ADDR clock stretching if the I2C enables only one slave address. This allows a slave data management by DMA only, without any interrupt from peripheral.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **FMP**: Fast-mode Plus 20 mA drive enable

0: 20 mA I/O drive disabled

1: 20 mA I/O drive enabled

Bits 23:20 Reserved, must be kept at reset value.

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: WUPEN can be set only when DNF[3:0] = 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 peripheral 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 peripheral 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 peripheral 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)

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.

32.8.2 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 PCLK + 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. | AUTO END | RE LOAD | NBYTES[7:0] | | | | | | | |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 31:26 Reserved, must be kept at reset value.

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.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to master mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

- 0: No STOP generation
- 1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in slave mode, by a timeout error detection, or when PE = 0.

- 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)

Condition: 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.

Condition: 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.

32.8.3 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 PCLK + 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. | Res. | OA1 MODE | OA1[9:0] | | | | | | | | | |
| rw | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bit 15 **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.

32.8.4 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 PCLK + 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.

32.8.5 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 section [I2C timings](#)), and for SCL high and low level counters (refer to section [I2C master initialization](#)).

$$t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$$

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 $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$ 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} .

$$t_{SDADEL} = SDADEL \times t_{PRESC}$$

Note: t_{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.

$$t_{SCLH} = (SCLH + 1) \times t_{PRESC}$$

Note: t_{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.

$$t_{SCLL} = (SCLL + 1) \times t_{PRESC}$$

Note: t_{SCLL} is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

32.8.6 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. | ADD CODE[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. | Res. | Res. | Res. | OVR | ARLO | BERR | TCR | TC | STOPF | NACKF | ADDR | RXNE | TXIS | TXE | | | | | | | |
| 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.

Bits 14:11 Reserved, must be kept at reset value.

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 the 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.

32.8.7 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. | Res. | Res. | Res. | OVR CF | ARLO CF | BERR CF | Res. | Res. | STOP CF | NACK CF | ADDR CF | Res. | Res. | Res. |
| | | | | | w | w | w | | | w | w | w | | | |

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

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.

32.8.8 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. | Res. | r | r | r | r | r | r | r |
| RXDATA[7:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | 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 I²C-bus.

32.8.9 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. | Res. | rw |
| | | | | | | | | | | | | | | | |
| TXDATA[7:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

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 I²C-bus

Note: These bits can be written only when TXE = 1.

32.8.10 I2C register map

The table below provides the I2C register map and the reset values.

Table 195. 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 | STOPFACLR | 0 | ADDRACLR | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | 0 | 0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| 0x04 | I2C_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. | 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 | 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. | 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 | 0 | 0 | 0 | 0 | 0 | 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. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 0x10 | I2C_TIMINGR | PRESC[3:0] | SCLDEL [3:0] | | SDADEL [3:0] | | SCLH[7:0] | | | | | | SCLL[7:0] | | | | | | OA1[9:0] | | | | | | OA2[7:1] | | | | | | RXDATA[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 | 0 | 0 | 0 | | |
| 0x14 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x18 | I2C_ISR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 0x1C | I2C_ICR | DIR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | | |
| 0x20 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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. | 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 | 0 | 0 | 0 | 0 | 0 | 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. | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |

Refer to [Section 2.2](#) for the register boundary addresses.

33 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART/UART).

33.1 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.

33.2 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

33.3 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

33.4 USART implementation

The tables below describe USART implementation. It also includes LPUART for comparison.

Table 196. Instance implementation on STM32U0 series

| Instance | STM32U031xx | STM32U073/83xx |
|----------|-------------|----------------|
| USART1 | Full | Full |
| USART2 | Full | Full |
| USART3 | Basic | Basic |
| USART4 | Basic | Basic |
| LPUART1 | Low-power | Low-power |
| LPUART2 | Low-power | Low-power |
| LPUART3 | - | Low-power |

Table 197. USART/LPUART features

| Modes/features ⁽¹⁾ | Full feature set | Basic feature set | Low-power feature set |
|---|------------------|-------------------|-----------------------|
| Hardware flow control for modem | X | X | X |
| Continuous communication using DMA | X | X | X |
| Multiprocessor communication | X | X | X |
| Synchronous mode (master/slave) | X | X | - |
| Smartcard mode | X | - | - |
| Single-wire half-duplex communication | X | X | X |
| IrDA SIR ENDEC block | X | - | - |
| LIN mode | X | - | - |
| Dual clock domain and wake-up from low-power mode | X | - | X |
| Receiver timeout interrupt | X | - | - |
| Modbus communication | X | - | - |
| Auto baud rate detection | X | - | - |
| Driver Enable | X | X | X |
| USART data length | 7, 8 and 9 bits | | |
| Tx/Rx FIFO | X | - | X |
| Tx/Rx FIFO size (bytes) | 8 | - | 8 |
| Prescaler | X | - | X |
| Wake-up from low-power mode | X ⁽²⁾ | - | X ⁽²⁾ |

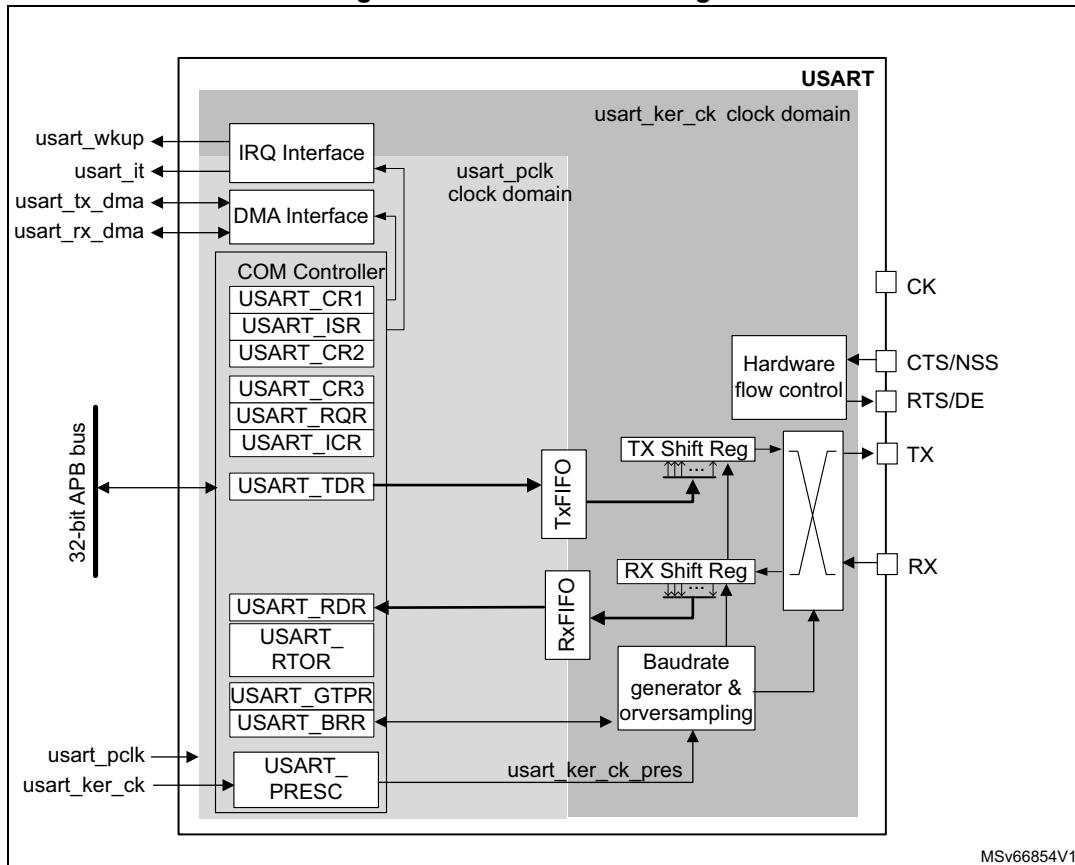
1. "X" = supported, "-" = not supported.

2. Wake-up supported from Stop 0 and Stop 1 modes.

33.5 USART functional description

33.5.1 USART block diagram

Figure 322. USART block diagram



MSv66854V1

33.5.2 USART pins and internal signals

Description USART input/output pins

- 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 **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.
- Synchronous master/slave mode and smartcard mode
The following pins are 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 in 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 (such as 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.

Refer to [Table 198](#) and [Table 199](#) for the list of USART input/output pins and internal signals.

Table 198. USART/UART input/output pins

| Pin name | Signal type | Description |
|---|-------------|---|
| USART_RX/UART_RX | Input | Serial data receive input |
| USART_TX/UART_TX | Output | Transmit data output |
| USART_CTS/UART_CTS | Input | Clear to send |
| USART_RTS/UART_RTS | Output | Request to send |
| USART_DE ⁽¹⁾ /UART_DE ⁽²⁾ | Output | Driver enable |
| USART_CK | Output | Clock output in synchronous master and smartcard modes. |
| USART_NSS ⁽³⁾ | Input | Slave select input in synchronous slave mode. |

1. USART_DE and USART_RTS share the same pin.
2. USART_DE and USART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 199. USART internal input/output signals

| Pin name | Signal type | Description |
|-------------|--------------|------------------------------------|
| uart_pclk | Input | APB clock |
| uart_ker_ck | Input | USART kernel clock |
| uart_wkup | Output | USART provides a wake-up interrupt |
| uart_it | Output | USART global interrupt |
| uart_tx_dma | Input/output | USART transmit DMA request |
| uart_rx_dma | Input/output | USART receive DMA request |

33.5.3 USART clocks

The simplified block diagram given in [Section 33.5.1: USART block diagram](#) shows two fully-independent clock domains:

- The **uart_pclk** clock domain
The **uart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **uart_ker_ck** kernel clock domain.
The **uart_ker_ck** is the USART clock source. It is independent from **uart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **uart_ker_ck** clock is stopped.
When the dual clock domain feature is not supported, the **uart_ker_ck** clock is the same as the **uart_pclk** clock.

There is no constraint between **uart_pclk** and **uart_ker_ck**: **uart_ker_ck** can be faster or slower than **uart_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 **uart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

33.5.4 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 323](#)):

- 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.

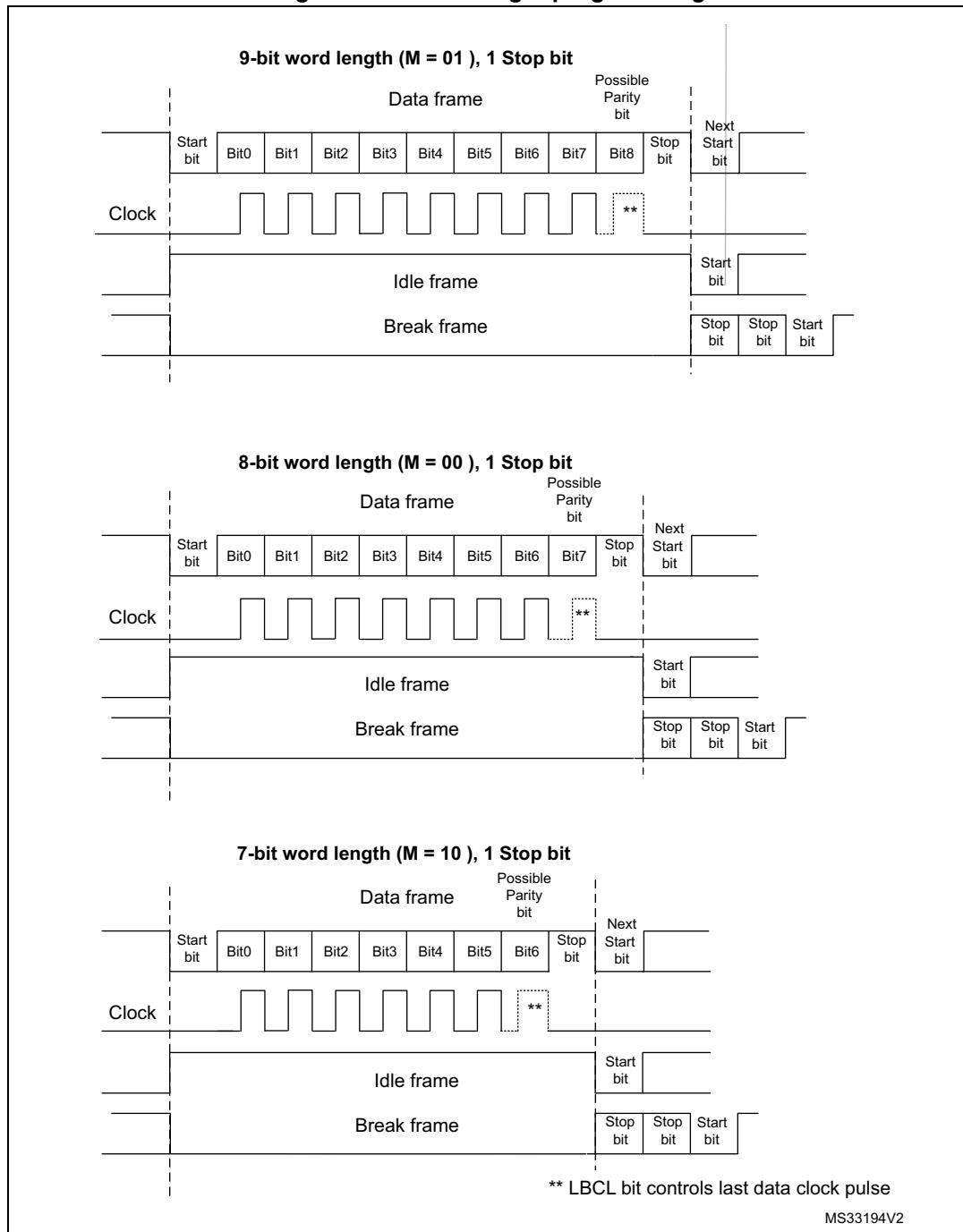
An **Idle character** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

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 323. Word length programming



33.5.5 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 Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bitfields.

In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG data have been received: 1 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 does not set the overrun flag.

- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bitfields.

33.5.6 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 must 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

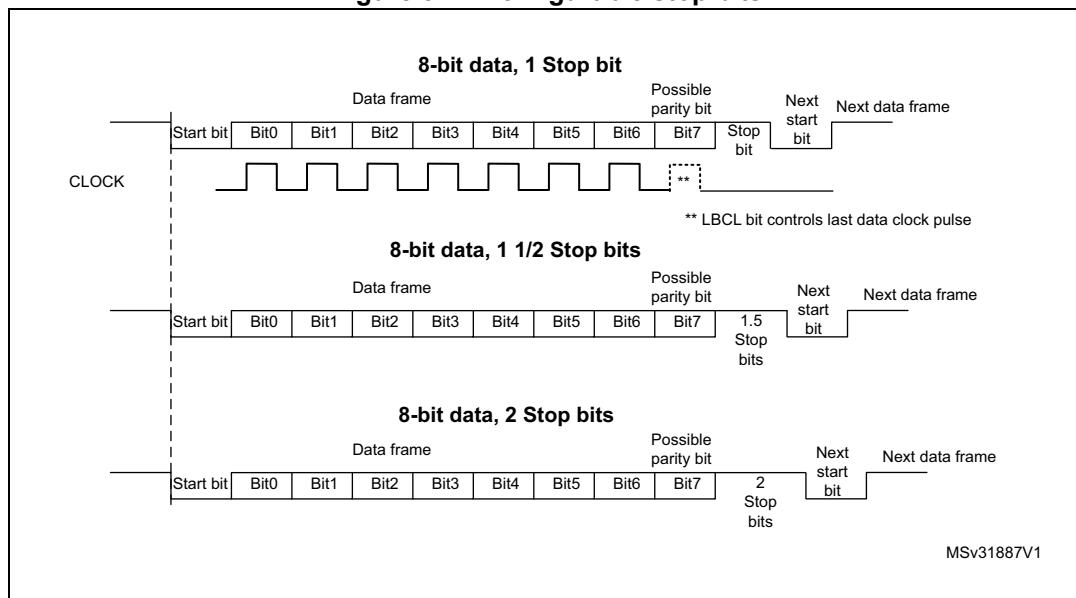
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- **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 is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see [Section 33.5.1: USART block diagram](#)). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 324. 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 33.5.20: 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 has completed.
 - 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 TXIE 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.

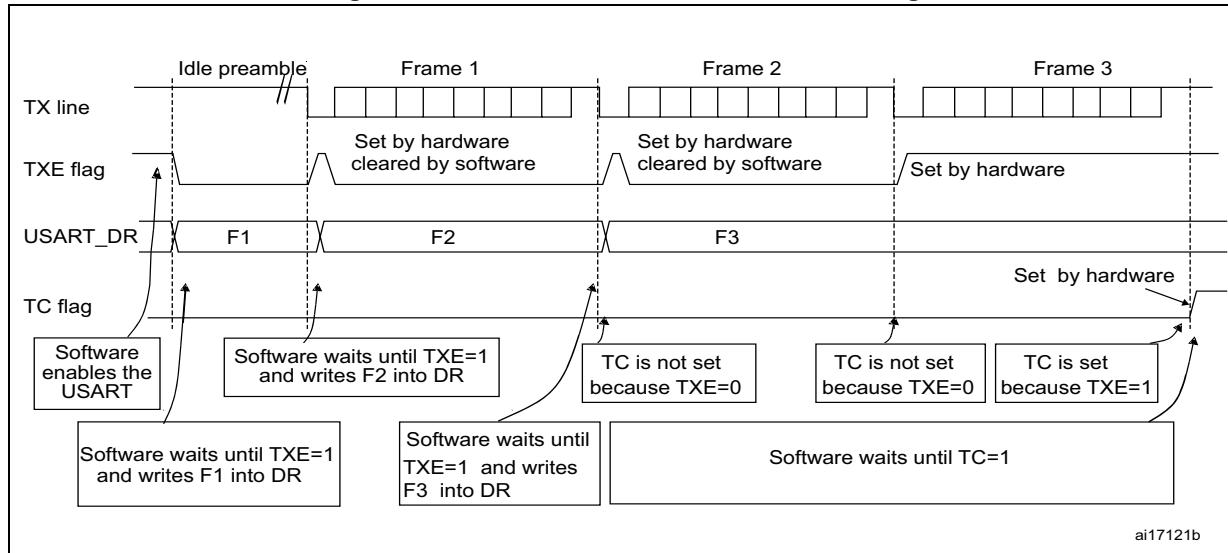
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

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 microcontroller to enter the low-power mode (see [Figure 325: TC/TXE behavior when transmitting](#)).

Figure 325. 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 323](#)).

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 complete (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.

33.5.7 USART receiver

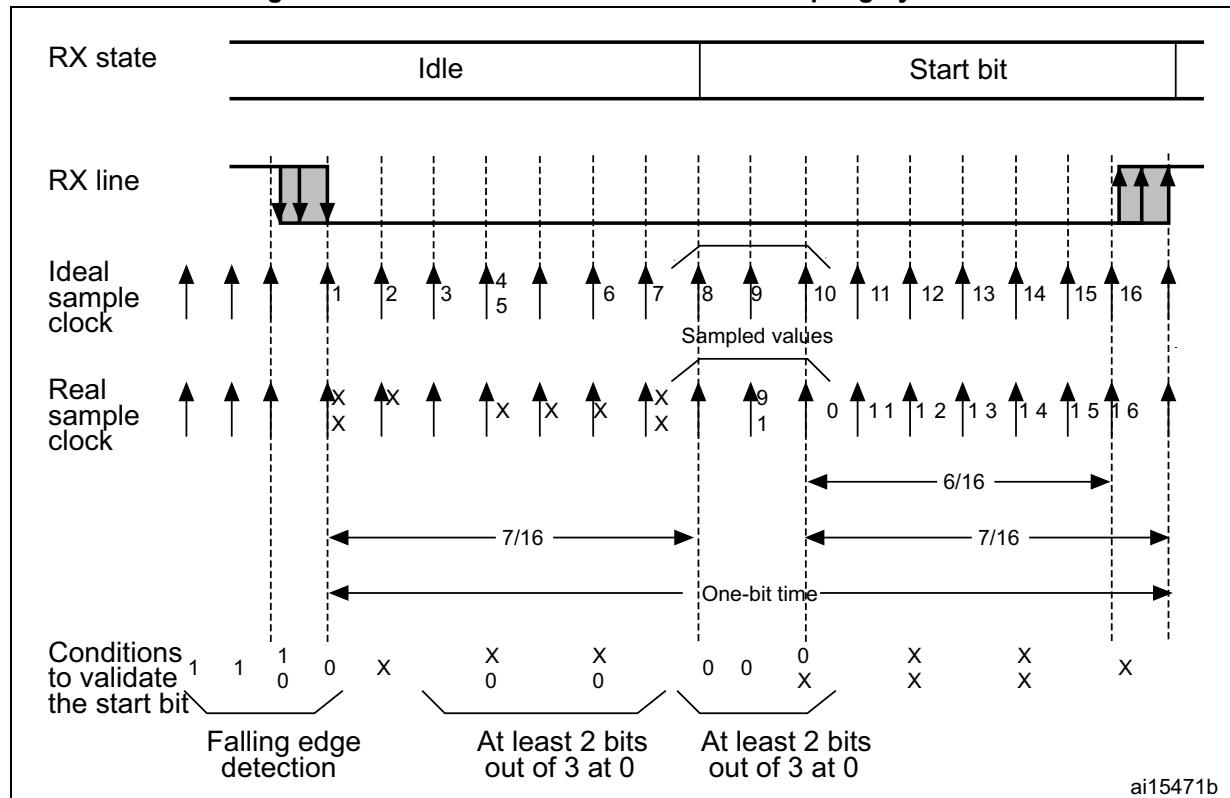
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

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 0 X 0 X 0 X 0.

Figure 326. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, 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,

- 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
- 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 33.5.20: 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, that is 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:

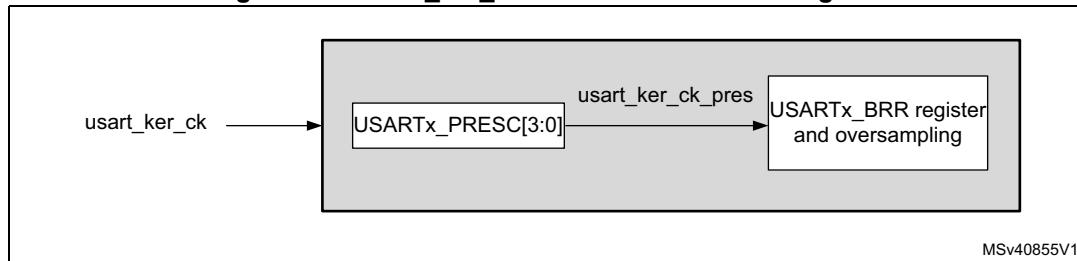
- Possible use of the USART in low-power mode
- Communication speed.

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 327. 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 328](#) and [Figure 329](#)).

Depending on the 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 33.5.9: Tolerance of the USART receiver to clock deviation on page 1030](#))
- 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

`uart_ker_ck_pres/16` (where `uart_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 the 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 [Table 200](#)) 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 33.5.9: Tolerance of the USART receiver to clock deviation on page 1030](#)). 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 NFCF bit in 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 328. Data sampling when oversampling by 16

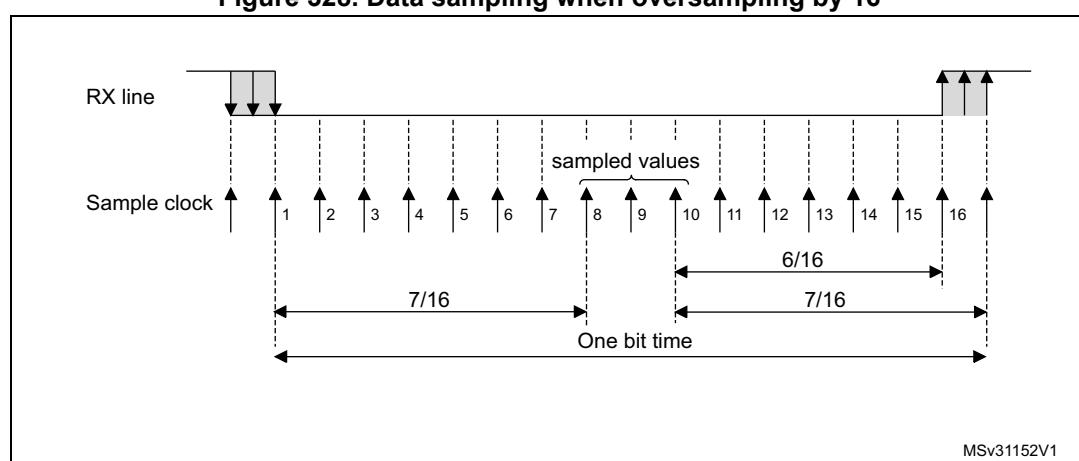


Figure 329. Data sampling when oversampling by 8

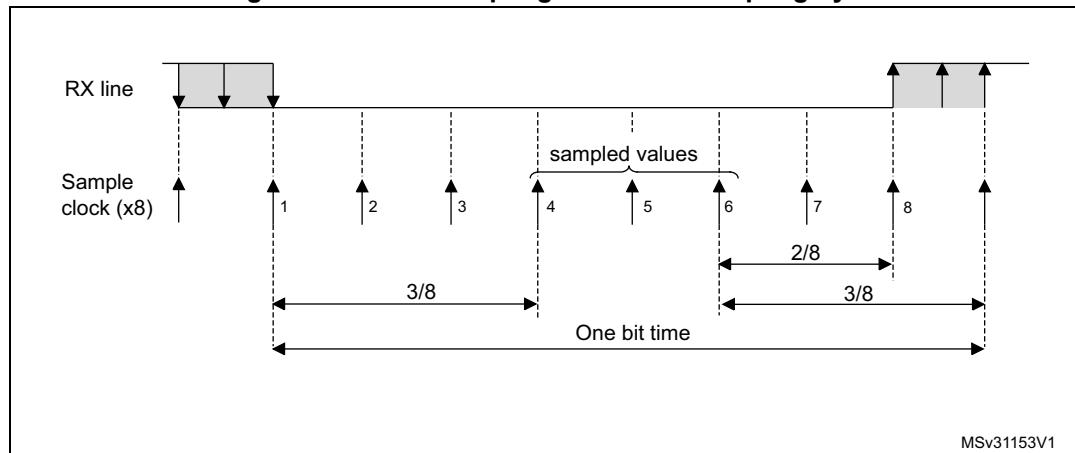


Table 200. Noise detection from sampled data

| 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 33.5.17: USART receiver timeout on page 1043](#) 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.

33.5.8 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:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

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 must 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 bauds 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 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
USARTDIV = 48 000 000/921 600
BRR = USARTDIV = 0x52 = 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

33.5.9 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)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver tolerance}$$

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 201](#), [Table 202](#), 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 201. 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 202. 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 201](#) and [Table 202](#) 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).

33.5.10 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/32763 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 must 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.*

33.5.11 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.

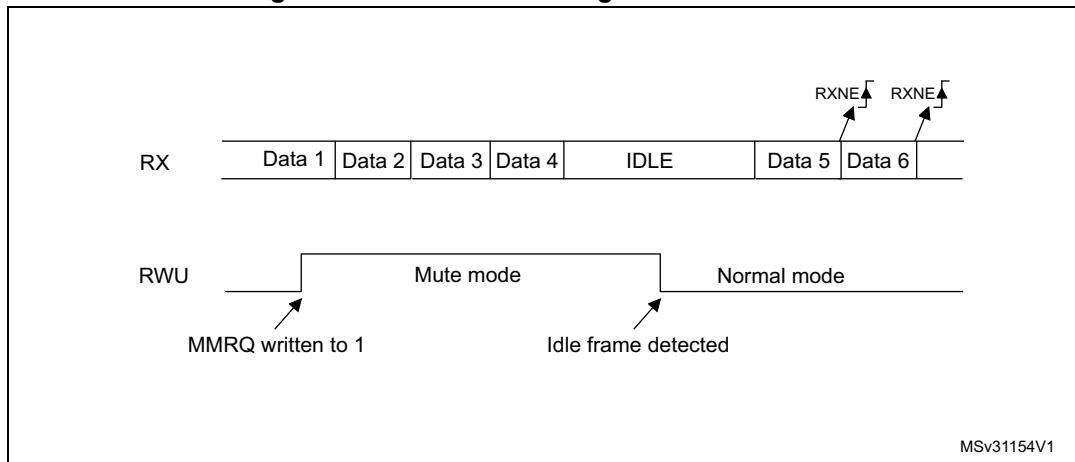
The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

- 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 330](#).

Figure 330. 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.

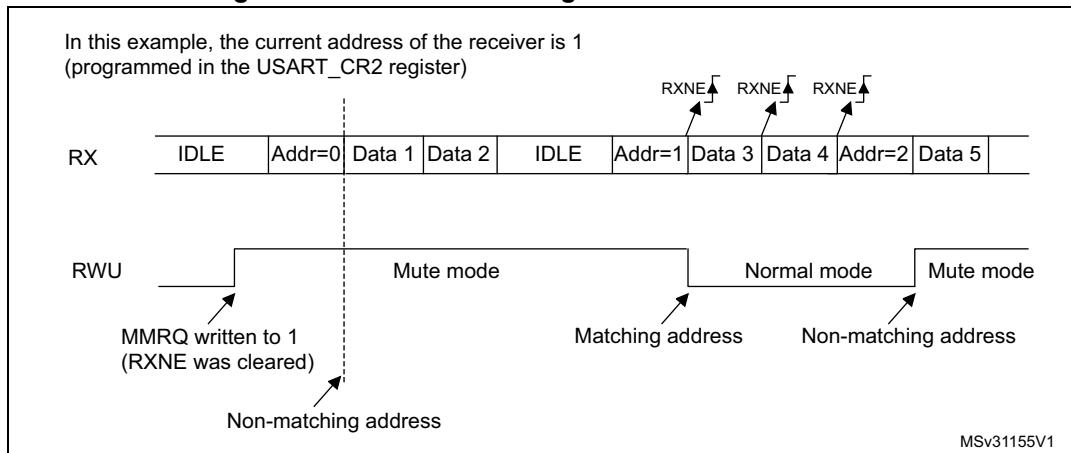
The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must 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 331](#).

Figure 331. Mute mode using address mark detection

33.5.12 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 has not 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.

33.5.13 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 203](#).

Table 203. 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 |

- Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is 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)).

33.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to [Section 33.4: USART implementation on page 1011](#).

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:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in [Section 33.5.5](#) 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 zero bits as a break character. Then 2 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.

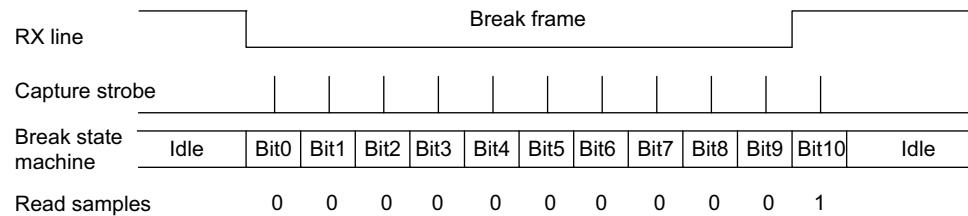
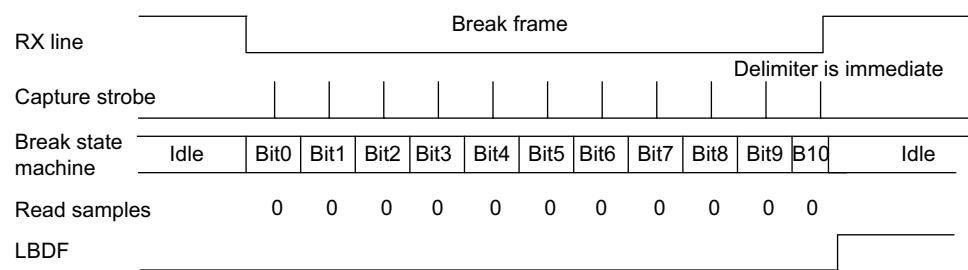
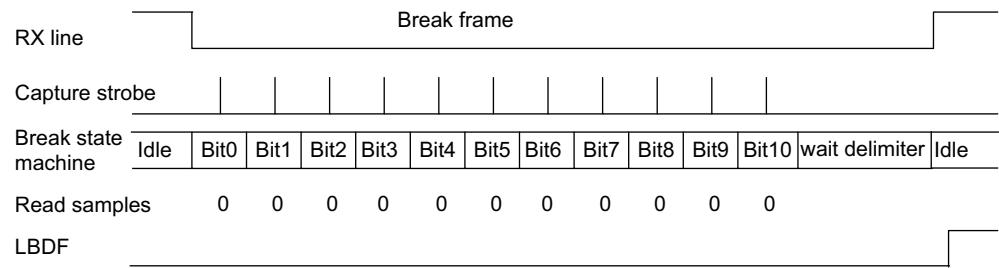
If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

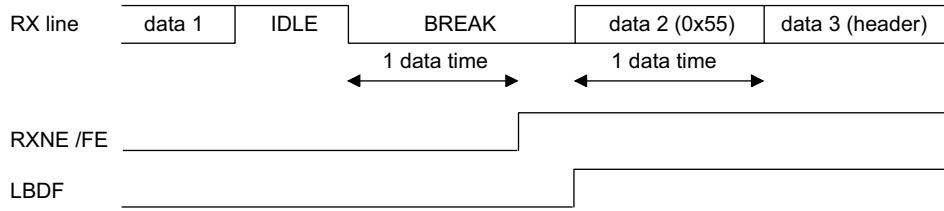
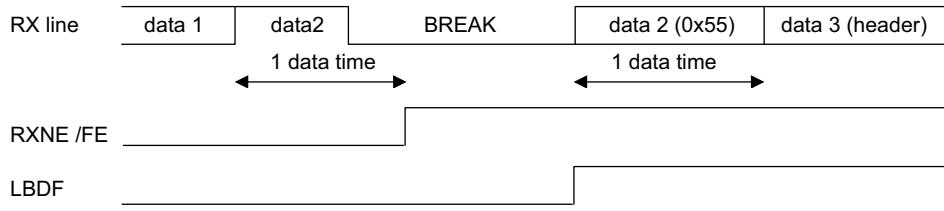
If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a '1', if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the [Figure 332: Break detection in LIN mode \(11-bit break length - LBDL bit is set\) on page 1038](#).

Examples of break frames are given on [Figure 333: Break detection in LIN mode vs. Framing error detection on page 1039](#).

Figure 332. Break detection in LIN mode (11-bit break length - LBDL bit is set)**Case 1: break signal not long enough => break discarded, LBDF is not set****Case 2: break signal just long enough => break detected, LBDF is set****Case 3: break signal long enough => break detected, LBDF is set**

MSv31156V1

Figure 333. Break detection in LIN mode vs. Framing error detection**Case 1: break occurring after an Idle****Case 2: break occurring while data is being received**

MSv31157V1

33.5.15 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 334](#), [Figure 335](#) and [Figure 336](#)).

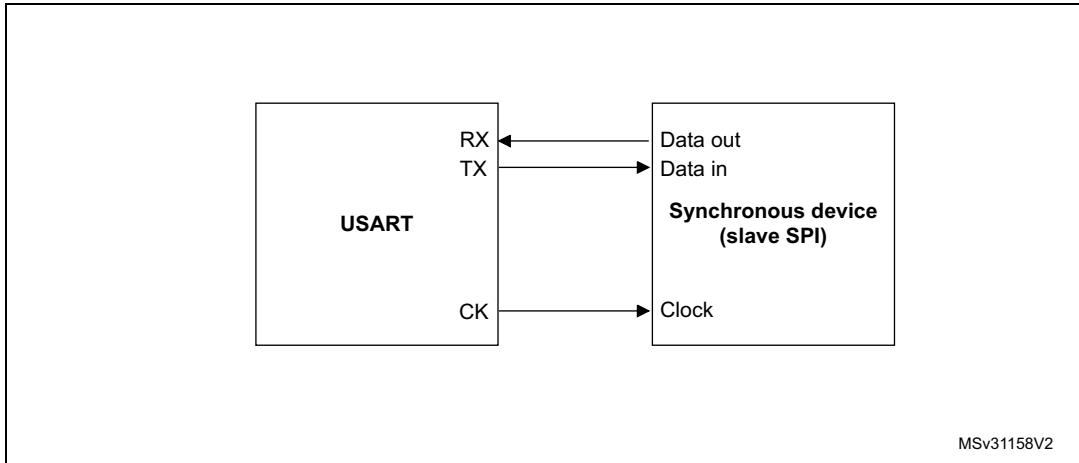
During the Idle state, preamble and send break, the external CK clock is not activated.

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.

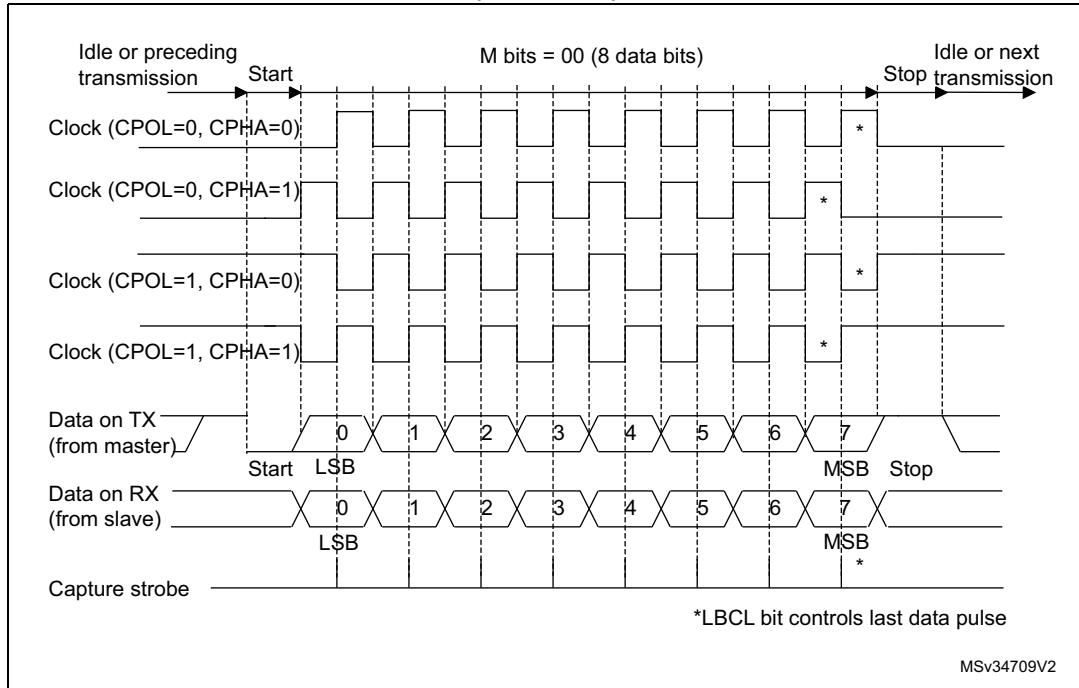
In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the 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.

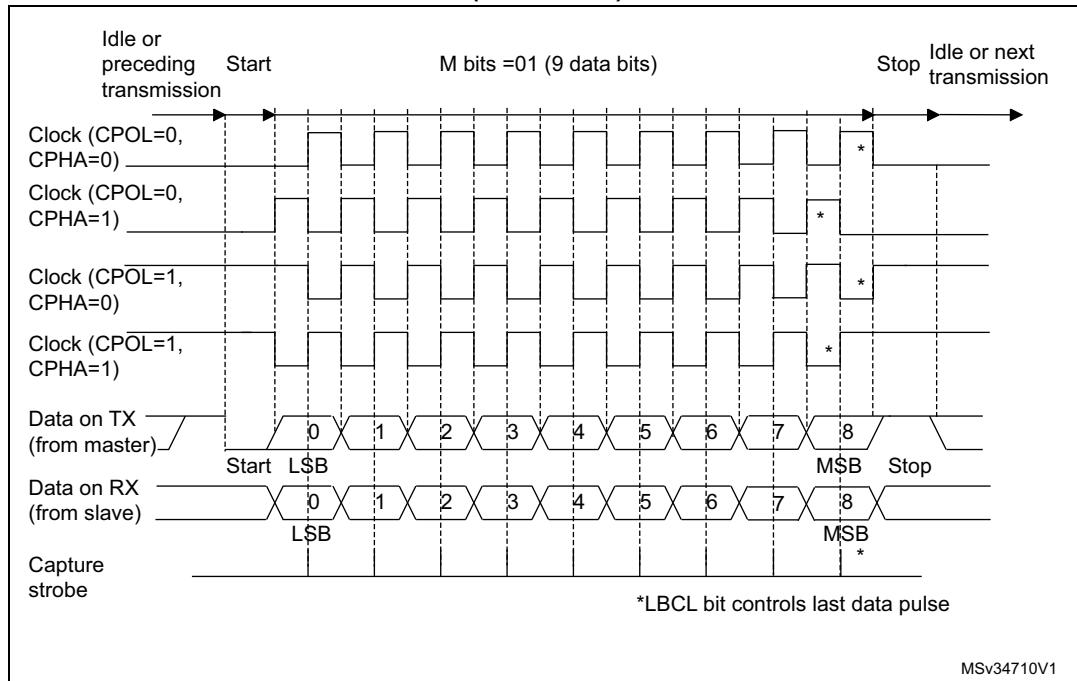
Figure 334. USART example of synchronous master transmission



**Figure 335. USART data clock timing diagram in synchronous master mode
(M bits =00)**



**Figure 336. USART data clock timing diagram in synchronous master mode
(M bits = 01)**



MSv34710V1

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:

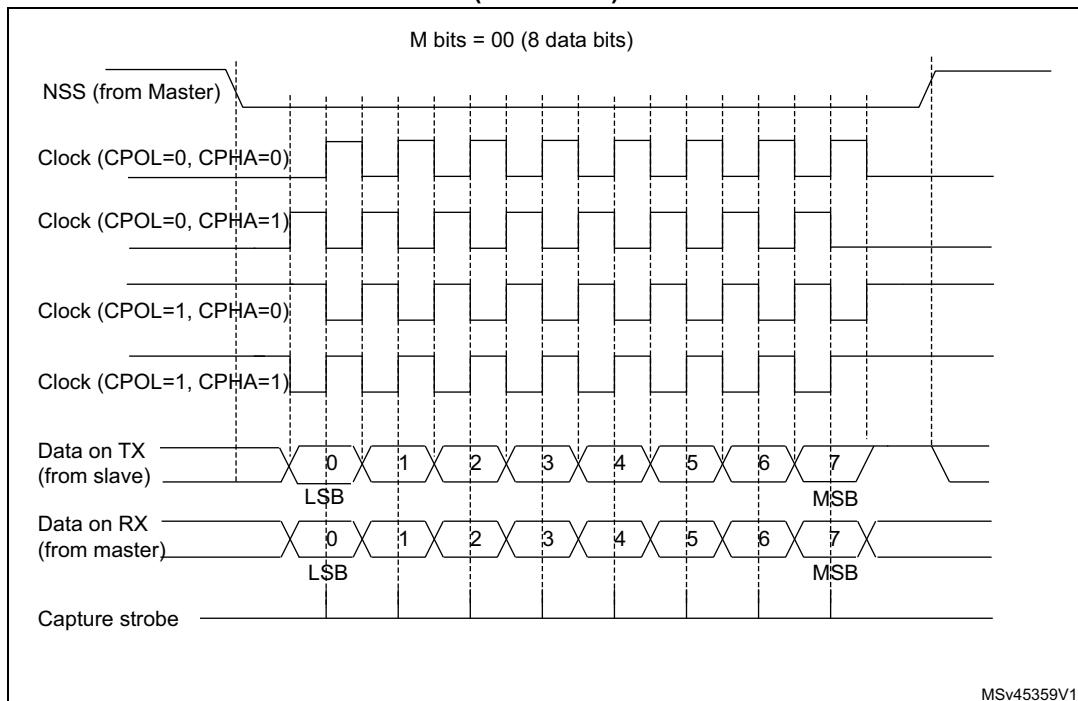
When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (uart_ker_ck_pres) must be greater than 3 times the CK input frequency.

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 337](#)).

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 337. 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)
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 underrun 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 must be written 3 usart_ker_ck cycles before the first CK edge.*

33.5.16 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.

33.5.17 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.

33.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to [Section 33.4: USART implementation on page 1011](#).

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.

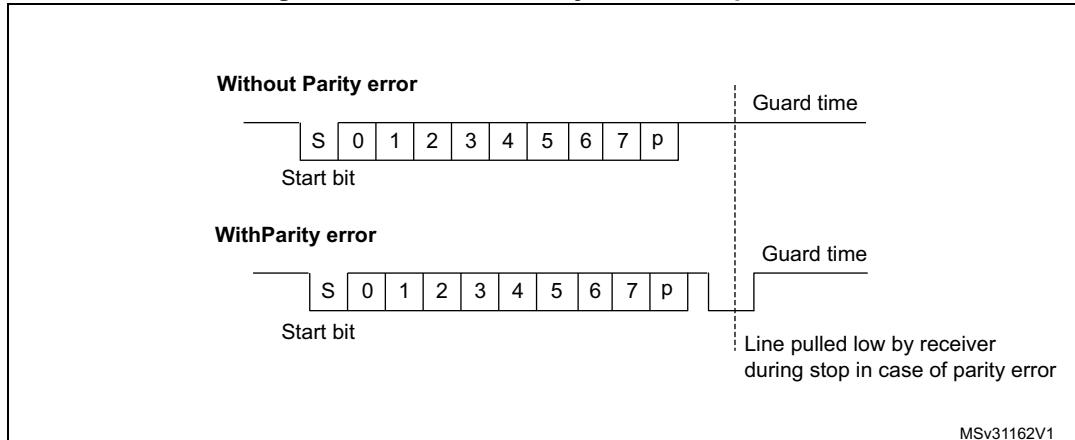
The USART must be configured as:

- 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 338](#) shows examples of what can be seen on the data line with and without parity error.

Figure 338. ISO 7816-3 asynchronous protocol



MSv31162V1

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.

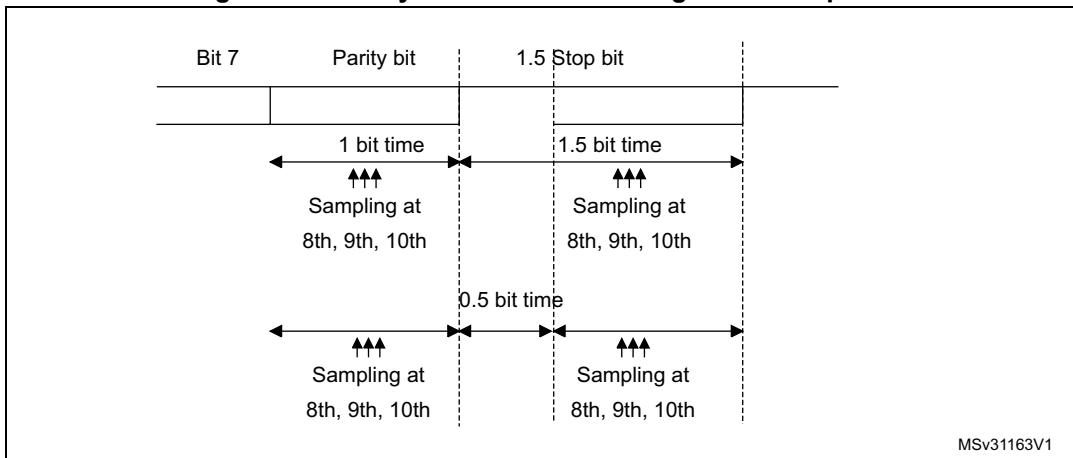
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it 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 de-assertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note:

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 339 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 339. Parity error detection using the 1.5 stop bits

MSv31163V1

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 must 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 EOBIIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: *The error checking code (LRC/CRC) must be computed/verified by software.*

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

Note: *When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.*

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card 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.

33.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to [Section 33.4: USART implementation on page 1011](#).

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 340](#)).

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 kbauds 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 must be avoided as the data to be transmitted may 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 341](#)).
 - 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 μ s. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods are accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
 - The receiver can communicate with a low-power transmitter.
 - In IrDA mode, the stop bits in the USART_CR2 register must be configured to '1 stop bit'.

IrDA low-power mode

- Transmitter

In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

- Receiver

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must 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 must 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 340. IrDA SIR ENDEC block diagram

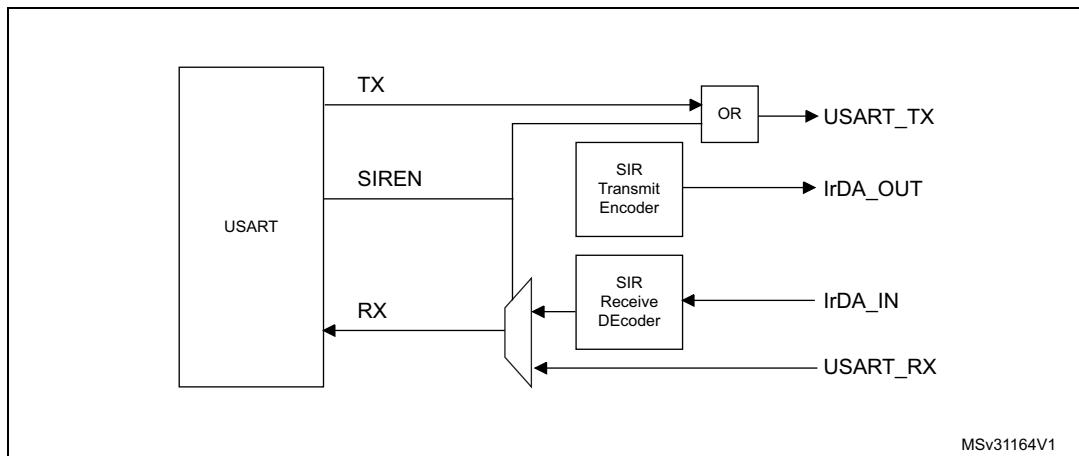
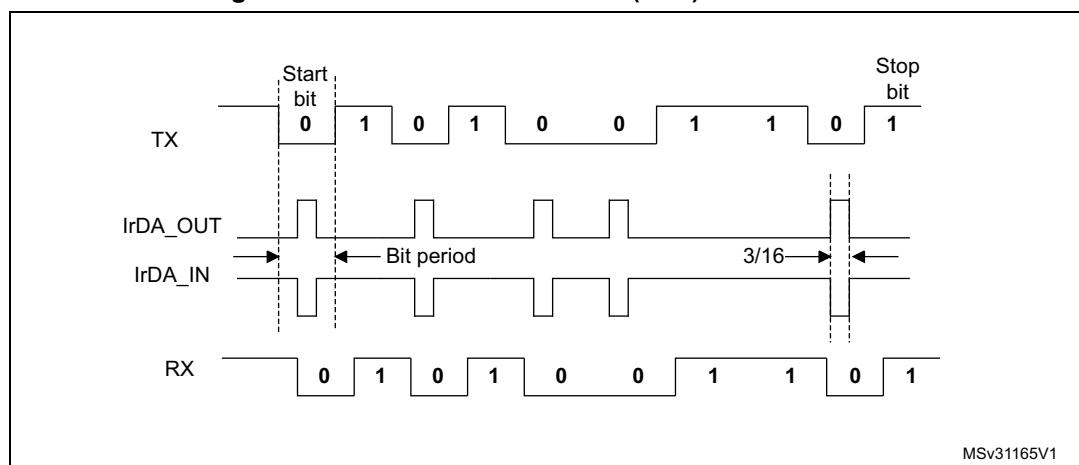


Figure 341. IrDA data modulation (3/16) - normal mode



33.5.20 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 33.4: USART implementation on page 1011](#) to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in [Section 33.5.7](#). 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 [section Direct memory access controller \(DMA\)](#)) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

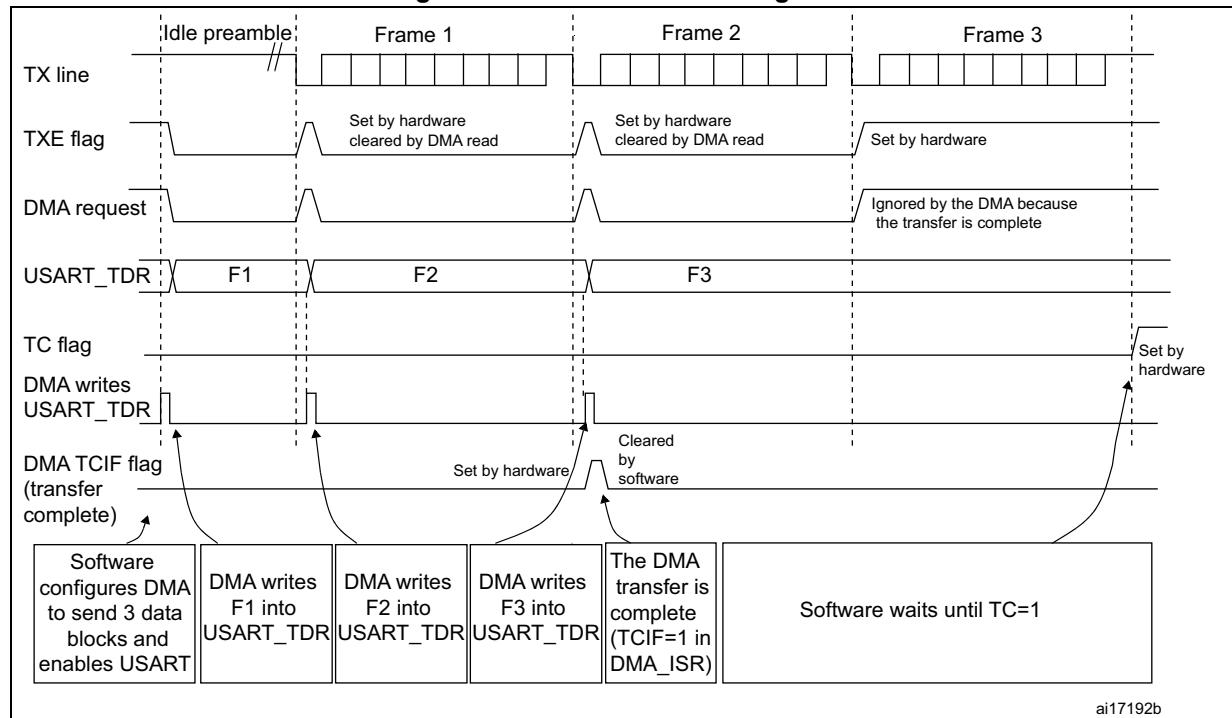
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication has completed. 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.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 342. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

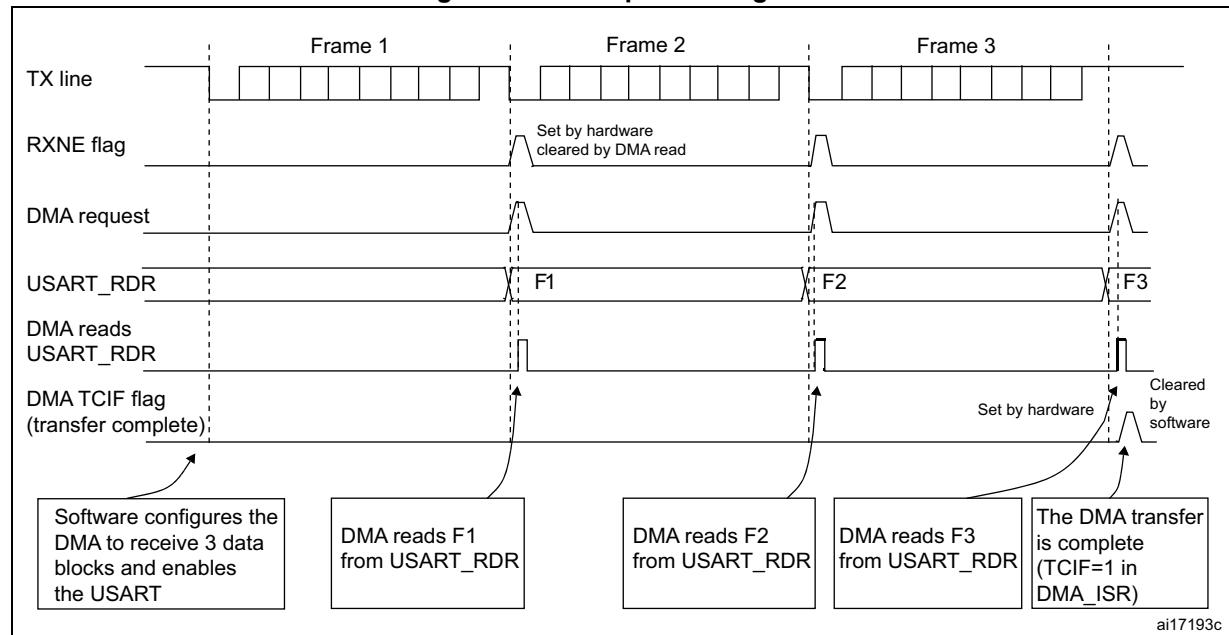
DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to section *Direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 343. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (that is, RXFNE = 1).

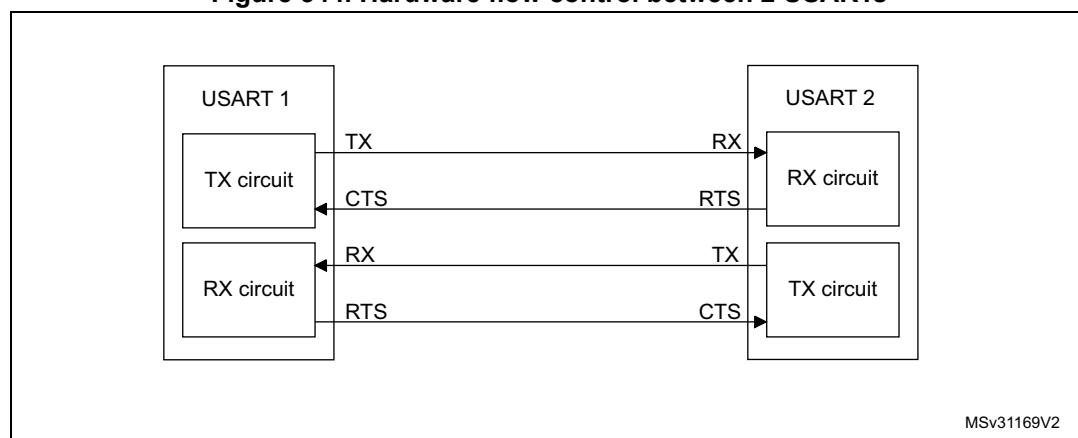
Error flagging and interrupt generation in multibuffer communication

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.

33.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The [Figure 344](#) shows how to connect two devices in this mode:

Figure 344. Hardware flow control between 2 USARTs

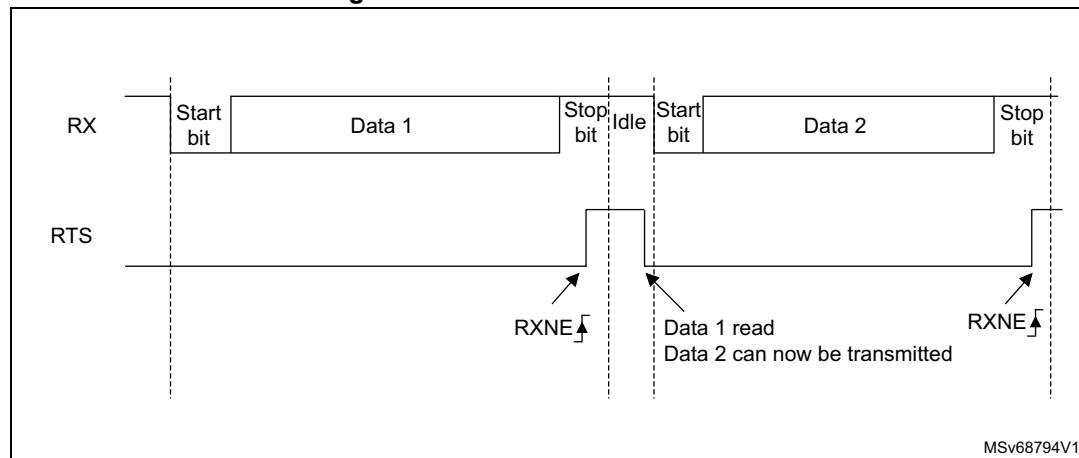


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

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 345](#) shows an example of communication with RTS flow control enabled.

Figure 345. RS232 RTS flow control



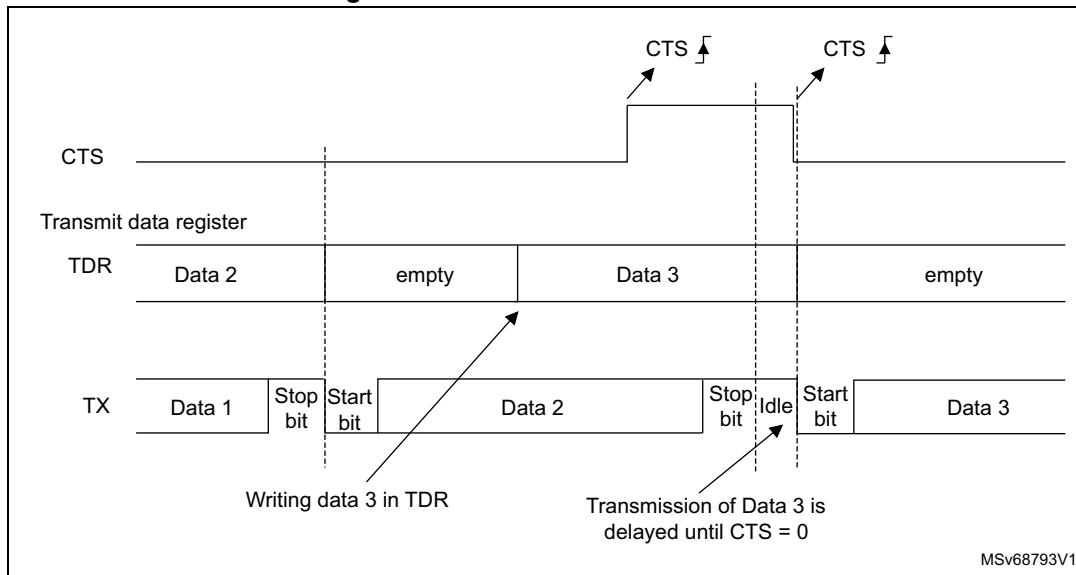
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

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 completes 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 346](#) shows an example of communication with CTS flow control enabled.

Figure 346. 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 must 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 de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] 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).

33.5.22 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 (that is, 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 347 shows the USART behavior when the wake-up event is verified.

Figure 347. Wake-up event verified (wake-up event = address match, FIFO disabled)

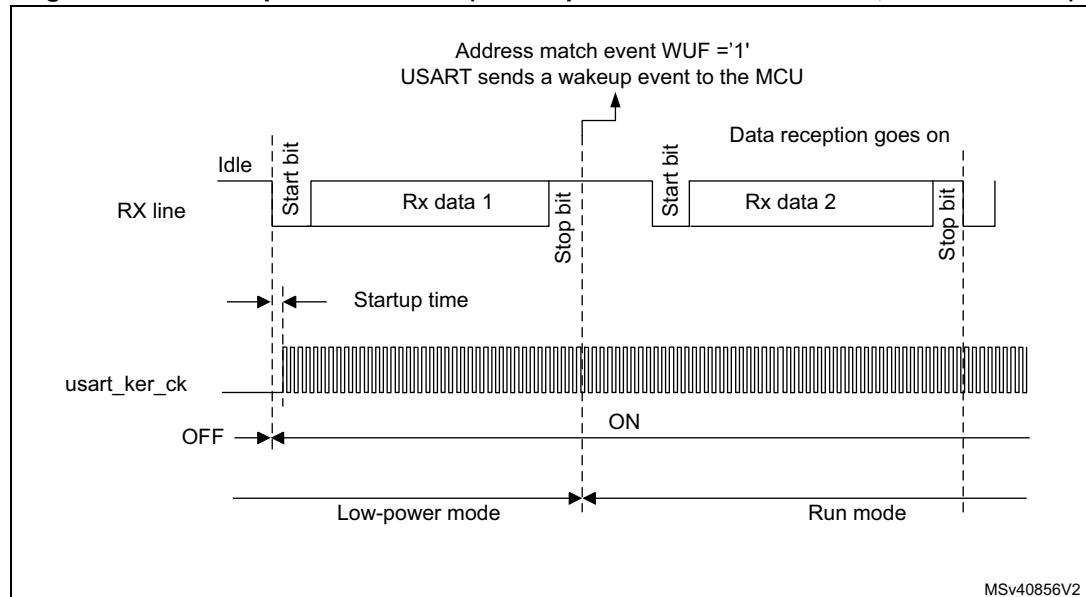
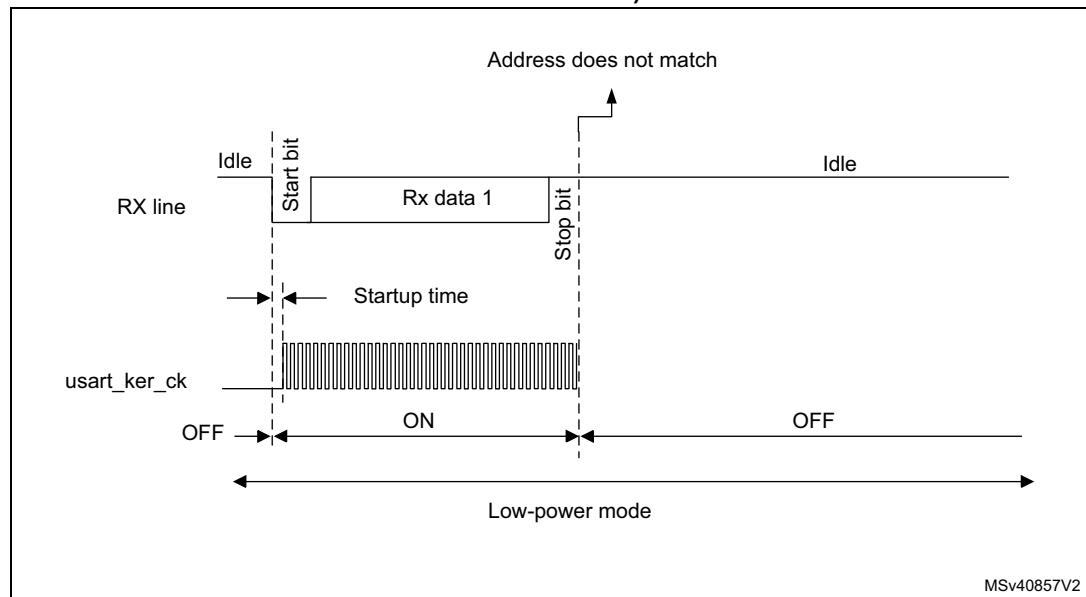


Figure 348 shows the USART behavior when the wake-up event is not verified.

Figure 348. 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 microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see [Section 33.5.9: 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 201: 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

$$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$$

$$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$$

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 fact, we need to consider at least the `uart_ker_ck` inaccuracy (DREC).

For example, if HSI is used as `uart_ker_ck`, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART} = 3 \mu s$ (values provided only as examples; for correct values, refer to the device datasheet).

$$D_{WUmax} = \text{USART receiver tolerance} - DREC = 3.41\% - 1\% = 2.41\%$$

$$T_{bitmin} = 3 \mu s / (11 \times 2.41\%) = 11.32 \mu s.$$

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: $1/11.32 \mu s = 88.36$ kbauds.

33.6 USART in low-power modes

Table 204. 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. 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. |

- Refer to [Section 33.4: 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.

33.7 USART interrupts

Refer to [Table 205](#) for a detailed description of all USART interrupt requests.

Table 205. 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 | TCBGTE | Write TDR or write 1 in TCBGT | | No | |

Table 205. USART interrupt requests (continued)

| 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 | 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 | RX-NEIE/RX-FNEIE | Write 1 in ORECF | | No | |
| | Idle line detected | IDLE | IDLEIE | Write 1 in IDLECF | | No | |
| | Parity error | PE | PEIE | Write 1 in PEFC | | 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 | |

- 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 33.4: USART implementation](#) for the list of supported Stop modes.
- 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).
- When OVRDIS = 0.

33.8 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).

33.8.1 USART control register 1 (USART_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------------|--------|------------|----|-------|-------|-----------|------|--------|------|---------|--------|-----------|----|------|----|
| RXF FIE | TXFEIE | 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 | 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 | TXFNIE | 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 exiting low-power mode.

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.

33.8.2 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, FIFOEN = 0

| 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 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 exiting low-power mode.

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.

33.8.3 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 | MSBFI RST | 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 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 33.4: USART implementation on page 1011](#).

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 -> Frame = Start10xxxxx)

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 ($V_{DD} = 1/\text{idle}$, Gnd=0/mark)
- 1: TX pin signal values are inverted. ($(V_{DD} = 0/\text{mark}$, 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 ($V_{DD} = 1/\text{idle}$, Gnd=0/mark)
- 1: RX pin signal values are inverted. ($(V_{DD} = 0/\text{mark}$, 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 USART.

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 33.4: USART implementation](#) on page 1011.

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

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 328](#) and [Figure 329](#))

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011.

Bit 4 ADDM7: 7-bit Address Detection/4-bit Address Detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE=0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 DIS_NSS:

When the 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.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1011.

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.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1011.

Note: The CPOL, CPHA and LBCL bits must not be written while the transmitter is enabled.

33.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

| 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 | WUS1 | WUS0 | 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

Others: Reserved, must not be used

This bitfield can only be written when the USART is disabled (UE = 0).

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

Others: Reserved, must not be used

This bitfield can only be written when the USART is disabled (UE = 0).

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 33.4: USART implementation on page 1011.

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 33.4: USART implementation on

page 1011.

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).

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 33.4: USART implementation on page 1011](#).

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 of 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 transmission 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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. Refer to [Section 33.4: USART implementation on page 1011](#).

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 in 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 are 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 33.4: USART implementation on page 1011.

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 completes 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 33.4: USART implementation on page 1011.

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 33.4: USART implementation on page 1011.

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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.

33.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

| | | | | | | | | | | | | | | | |
|------|------|------|------------|------------|-------|------|-------------|------|-------|------|------|--------------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | TCBG TIE | Res. | WUFIE | WUS1 | WUS0 | SCARCNT[2:0] | | | Res. |
| | | | | | | | rw | | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DEP | DEM | DDRE | OVR DIS | ONE BIT | CTSIE | CTSE | RTSE | DMAT | DMAR | SCEN | NACK | HD SEL | IRLP | IREN | EIE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

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 33.4: USART implementation on page 1011.

Bit 23 Reserved, must be kept at reset value.

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 33.4: USART implementation on page 1011.

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).

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 33.4: USART implementation on page 1011.

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 of 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 transmission 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 in 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 are 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 33.4: USART implementation on page 1011](#).

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 completes 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011.*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 33.4: USART implementation on page 1011.*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.

33.8.6 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] correspond to 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.

33.8.7 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 33.4: USART implementation on page 1011](#).

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA normal and Low-power baud rate

This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming the prescaler for dividing the USART source clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This 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 33.4: USART implementation on page 1011](#).

33.8.8 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 bit duration.

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 33.4: USART implementation on page 1011](#).

33.8.9 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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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, that is 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 must 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 33.4: USART implementation on page 1011](#).

33.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

| 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 | REACK | TEACK | 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, that is, 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, that is, 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 has completed 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 has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
 1: Transmission has completed 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011.

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 33.4: USART implementation on page 1011.

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 33.4: USART implementation on page 1011.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE=1 in 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 ongoing

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 has 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 33.4: USART implementation on page 1011](#).

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 EOBIIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to 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 33.4: USART implementation on page 1011](#).

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 **LBDIF**: 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 33.4: USART implementation on page 1011](#).*

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 TXFNIE 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 must be checked prior to writing in TXFIFO (TXFNF and TXFE is 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 has completed, and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by a write to the USART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, 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 (that is, 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 NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the 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 33.5.9: Tolerance of the USART receiver to clock deviation on page 1030](#)).

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 reception mode. It is cleared by software, writing 1 to the PECE bit 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.

33.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

| 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 | CTSF | 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 has completed, 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 has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 WUJCF 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

- 0: No break character transmitted
- 1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE=1in the USART_CR1 register.

- 0: No Character match detected
- 1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

- 0: USART is idle (no reception)
- 1: Reception ongoing

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 has 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 33.4: USART implementation on page 1011](#).

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 EOBIIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 empty

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. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE=1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

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 (that is, 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 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

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 NFCF bit in the USART_ICR register.

- 0: No noise is detected
- 1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE 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 33.5.9: Tolerance of the USART receiver to clock deviation on page 1030](#)).

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 reception mode. It is cleared by software, writing 1 to the PECE bit 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.

33.8.12 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 | TCBGT CF | TCCF | TXFEC F | 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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#)

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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 33.4: USART implementation on page 1011](#).

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.

33.8.13 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | RDR[8:0] | | | | | | | |
| | | | | | | | | 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 [Section 33.5.1: USART block diagram](#)).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

33.8.14 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TDR[8:0] | | | | | | | |
| | | | | | | | | 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 [Section 33.5.1: USART block diagram](#)).

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.

33.8.15 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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | PRESCALER[3:0] |
| | | | | | | | | | | | | | | | 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

Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1011](#).

33.8.16 USART register map

Table 206. USART register map and reset values

Table 206. USART 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 | | | |
|--------|---------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 0x20 | USART_ICR | Res. | WUCF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Reset value | Res. | |
| 0x24 | USART_RDR | Res. | |
| | Reset value | Res. | |
| 0x28 | USART_TDR | Res. | |
| | Reset value | Res. | |
| 0x2C | USART_PRES | Res. |
| | Reset value | Res. |

Refer to [Section 2.2 on page 55](#) for the register boundary addresses.

34 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

34.1 Introduction

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

34.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data
 - Each FIFO can be enabled/disabled by software and come with status flags for FIFO states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted 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 RS485 transceiver

- Transfer detection flags:
 - Receive buffer full
 - Transmit buffer empty
 - Busy and end of transmission flags
- Parity control:
 - Transmits parity bit
 - Checks parity of received data byte
- Four error detection flags:
 - Overrun error
 - Noise detection
 - Frame error
 - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

34.3 LPUART implementation

The tables below describe LPUART implementation. It also includes USARTs and UARTs for comparison.

Table 207. Instance implementation on STM32U0 series

| Instance | STM32U031xx | STM32U073/83xx |
|----------|-------------|----------------|
| USART1 | Full | Full |
| USART2 | Full | Full |
| USART3 | Basic | Basic |
| USART4 | Basic | Basic |
| LPUART1 | Low-power | Low-power |
| LPUART2 | Low-power | Low-power |
| LPUART3 | - | Low-power |

Table 208. USART/LPUART features

| Modes/features ⁽¹⁾ | Full feature set | Basic feature set | Low-power feature set |
|---|------------------|-------------------|-----------------------|
| Hardware flow control for modem | X | X | X |
| Continuous communication using DMA | X | X | X |
| Multiprocessor communication | X | X | X |
| Synchronous mode (master/slave) | X | X | - |
| Smartcard mode | X | - | - |
| Single-wire half-duplex communication | X | X | X |
| IrDA SIR ENDEC block | X | - | - |
| LIN mode | X | - | - |
| Dual clock domain and wake-up from low-power mode | X | - | X |
| Receiver timeout interrupt | X | - | - |
| Modbus communication | X | - | - |
| Auto baud rate detection | X | - | - |
| Driver Enable | X | X | X |
| USART data length | 7, 8 and 9 bits | | |
| Tx/Rx FIFO | X | - | X |
| Tx/Rx FIFO size (bytes) | 8 | - | 8 |
| Prescaler | X | - | X |
| Wake-up from low-power mode | X ⁽²⁾ | - | X ⁽²⁾ |

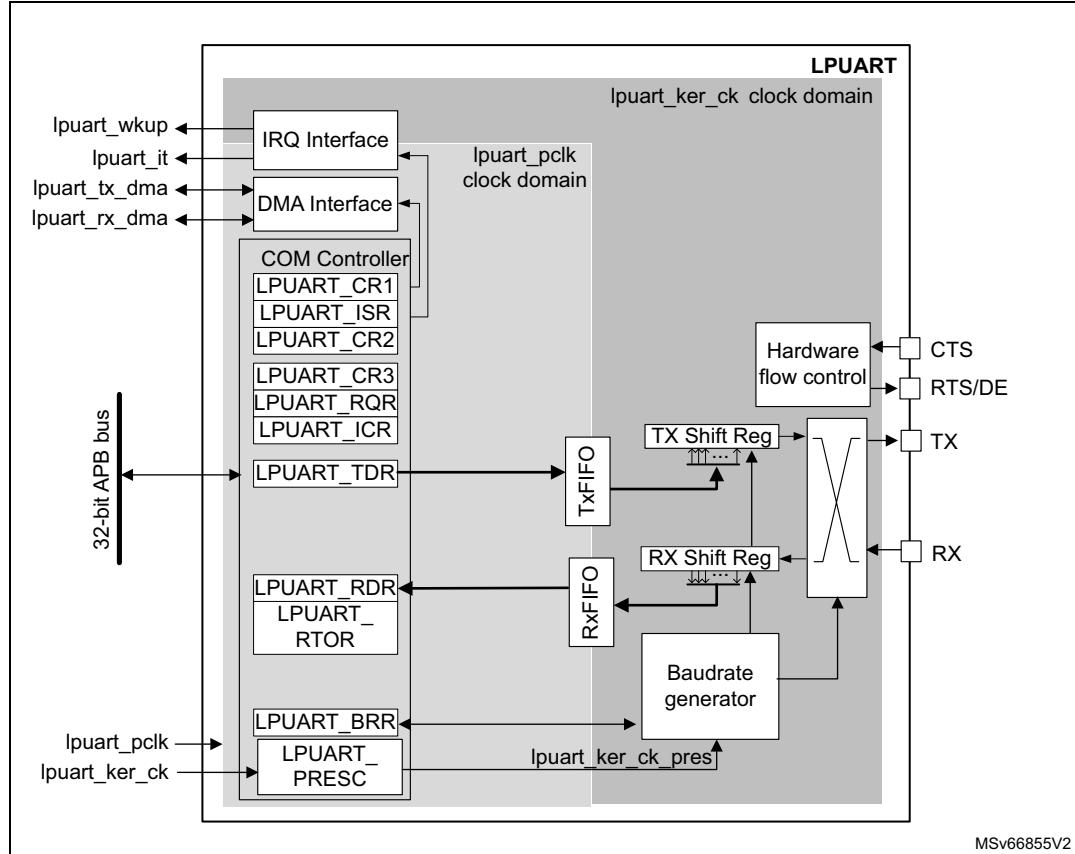
1. X = supported.

2. Wake-up supported from Stop 0 and Stop 1 modes.

34.4 LPUART functional description

34.4.1 LPUART block diagram

Figure 349. LPUART block diagram



34.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

LPUART bidirectional communications requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- RX** (Receive Data Input):
RX is the serial data input.
- TX** (Transmit Data Output)

When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode

The RS232 hardware flow control mode requires the following pins:

- 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 LPUART is ready to receive data.

- RS485 hardware flow control mode

The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to [Table 209](#) and [Table 210](#) for the list of LPUART input/output pins and internal signals.

Table 209. LPUART input/output pins

| Pin name | Signal type | Description |
|--------------------------|-------------|----------------------------|
| LPUART_RX | Input | Serial data receive input. |
| LPUART_TX | Output | Transmit data output. |
| LPUART_CTS | Input | Clear to send |
| LPUART_RTS | Output | Request to send |
| LPUART_DE ⁽¹⁾ | Output | Driver enable |

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 210. LPUART internal input/output signals

| Pin name | Signal type | Description |
|---------------|-------------|-------------------------------------|
| lpuart_pclk | Input | APB clock |
| lpuart_ker_ck | Input | LPUART kernel clock |
| lpuart_wkup | Output | LPUART provides a wake-up interrupt |

Table 210. LPUART internal input/output signals (continued)

| Pin name | Signal type | Description |
|---------------|--------------|-----------------------------|
| lpuart_it | Output | LPUART global interrupt |
| lpuart_tx_dma | Input/output | LPUART transmit DMA request |
| lpuart_rx_dma | Input/output | LPUART receive DMA request |

34.4.3 LPUART clocks

The simplified block diagram given in [Section 34.4.1: LPUART block diagram](#) shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.

- The **lpuart_ker_ck** kernel clock domain

The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

When the dual clock domain feature is not supported, the **lpuart_ker_ck** is the same as the **lpuart_pclk** clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

34.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see [Figure 323](#)).

- 7-bit character length: M[1:0] = '10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

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.

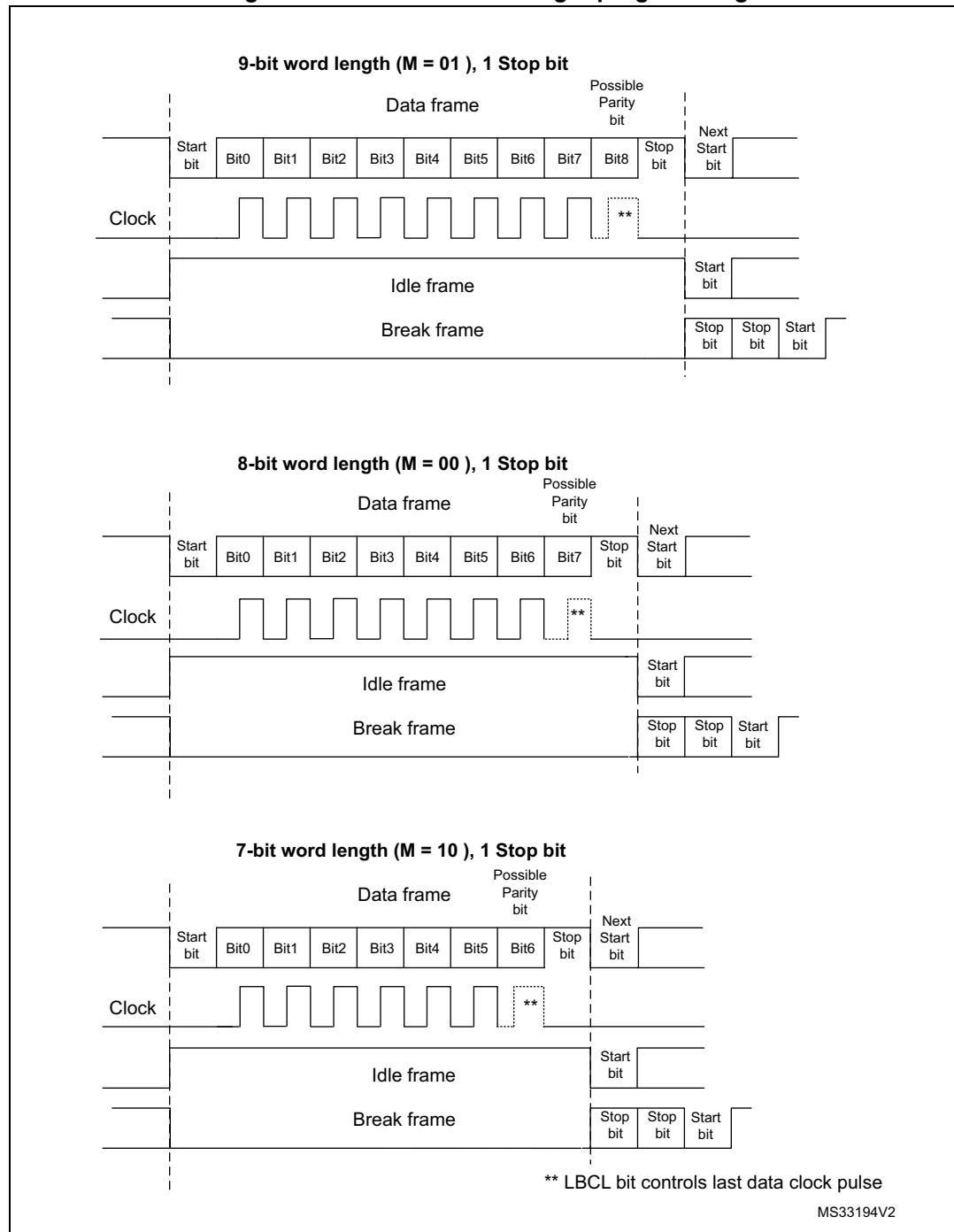
An **Idle character** is interpreted as an entire frame of "1"s (the number of "1"s includes the number of stop bits).

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 clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 350. LPUART word length programming



34.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bits 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 LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bitfields.

In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG data have been received: 1 data in LPUART_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 LPUART_RDR. As a result, the next received data does not set the overrun flag.

- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bitfields.

34.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see [Section 34.4.1: LPUART block diagram](#)).

When FIFO mode is enabled, the data written to the LPUART_TDR register 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 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit must 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 is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

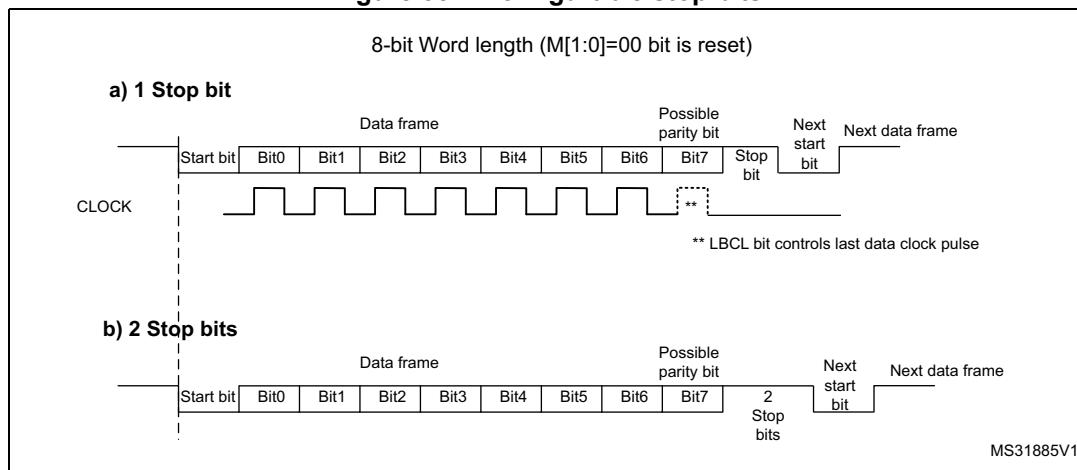
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13,12).

- **1 stop bit:** This is the default value of number of stop bits.
- **2 Stop bits:** This is supported by normal LPUART, single-wire and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 351. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 34.4.13: Continuous communication using DMA and LPUART](#).
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
 - When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
 - When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_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 LPUART_TDR register, wait until TC=1. This indicates that the transmission of the last frame has completed.
 - When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
 - 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 LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:

- the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
- the LPUART_TDR register is empty;
- the next data can be written to the LPUART_TDR register without overwriting the previous data.

The TXE flag generates an interrupt if the TXEIE bit is set.

When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO Not Full) flag is set by hardware to indicate that:
 - the TXFIFO is not full;
 - the LPUART_TDR register is empty;
 - the next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the

LPUART_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.

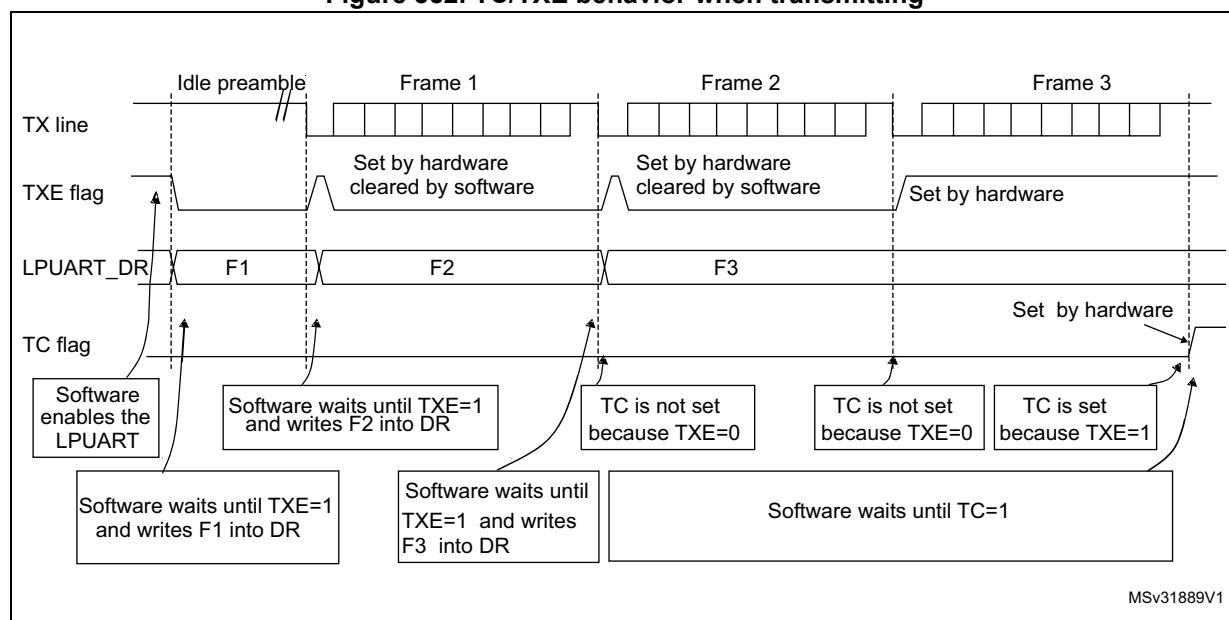
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC=1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see [Figure 352: TC/TXE behavior when transmitting](#)).

Figure 352. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see [Figure 350](#)).

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 complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

34.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 34.4.13: Continuous communication using DMA and LPUART](#).
6. Set the RE bit LPUART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_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 in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication mode:
 - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
 - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
 - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
 - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, 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 when a character is received when RXNE has not been reset.

Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- the ORE bit is set;
- the RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.;
- the shift register is overwritten. After that, any data received during overrun is lost.
- an interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

Data can not be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

- the ORE bit is set;
- the first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
- the shift register is overwritten. After that, any data received during overrun is lost.
- an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note:

The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE=0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the Clock Control system (see Section *Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

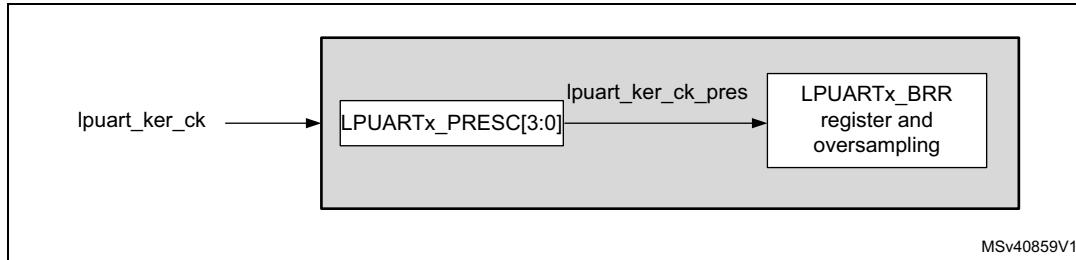
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the Ipuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the Ipuart_ker_ck is the same as Ipuart_pclk.

The Ipuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 353. Ipuart_ker_ck clock divider block diagram



Some Ipuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and Wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note:

There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the LPUART_RDR register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit:** sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits:** sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

34.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

f_{CK} must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. For example, if the LPUART clock source frequency is 100 MHz, the maximum baud rate that can be reached is about 33 Mbauds.

Table 211. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz

| Baud rate | | lpuart_ker_ck_pres= 32.768 kHz | | |
|-----------|-------------|--------------------------------|--|--|
| S.No | Desired | Actual | Value programmed in the baud rate register | % Error = (Calculated - Desired) B.rate / Desired B.rate |
| 1 | 0.3 kbaud | 0.3 kbaud | 0x6D3A | 0 |
| 2 | 0.6 kbaud | 0.6 kbaud | 0x369D | 0 |
| 3 | 1200 bauds | 1200.087 bauds | 0x1B4E | 0.007 |
| 4 | 2400 bauds | 2400.17 bauds | 0xDA7 | 0.007 |
| 5 | 4800 bauds | 4801.72 bauds | 0x6D3 | 0.035 |
| 6 | 9600 kbauds | 9608.94 bauds | 0x369 | 0.093 |

Table 212. Error calculation for programmed baud rates at $f_{CK} = 100$ MHz

| Baud rate | | $f_{CK} = 100\text{MHz}$ | | |
|-----------|--------------|--------------------------|--|--|
| S.No | Desired | Actual | Value programmed in the baud rate register | % Error = (Calculated - Desired) B.rate / Desired B.rate |
| 1 | 38400 bauds | 38400,04 bauds | A2C2A | 0,0001 |
| 2 | 57600 bauds | 57600,06 bauds | 6C81C | 0,0001 |
| 3 | 115200 bauds | 115200,12 bauds | 3640E | 0,0001 |
| 4 | 230400 bauds | 230400,23 bauds | 1B207 | 0,0001 |
| 5 | 460800 bauds | 460804,61 bauds | D903 | 0,001 |
| 6 | 921600 bauds | 921625,81 bauds | 6C81 | 0,0028 |
| 7 | 4000 kbauds | 4000000,00 bauds | 1900 | 0 |
| 8 | 10000 kbauds | 10000000,00 bauds | A00 | 0 |
| 9 | 20000 kbauds | 20000000,00 bauds | 500 | 0 |
| 10 | 30000 kbauds | 33032258,06 bauds | 307 | 0,1 |

34.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART 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)

$$\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when $M[1:0] = 01$:

$$\text{DWU} = \frac{t_{\text{WULPUART}}}{11 \times \text{Tbit}}$$

when $M[1:0] = 00$:

$$\text{DWU} = \frac{t_{\text{WULPUART}}}{10 \times \text{Tbit}}$$

when $M[1:0] = 10$:

$$\text{DWU} = \frac{t_{\text{WULPUART}}}{9 \times \text{Tbit}}$$

t_{WULPUART} 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 LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in [Table 213](#):

- Number of Stop bits defined through $\text{STOP}[1:0]$ bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 213. Tolerance of the LPUART receiver

| M bits | 768 < BRR < 1024 | 1024 < BRR < 2048 | 2048 < BRR < 4096 | 4096 ≤ BRR |
|-------------------------------|-------------------------------|--------------------------------|--------------------------------|-------------------|
| 8 bits ($M=00$), 1 Stop bit | 1.82% | 2.56% | 3.90% | 4.42% |
| 9 bits ($M=01$), 1 Stop bit | 1.69% | 2.33% | 2.53% | 4.14% |
| 7 bits ($M=10$), 1 Stop bit | 2.08% | 2.86% | 4.35% | 4.42% |
| 8 bits ($M=00$), 2 Stop bit | 2.08% | 2.86% | 4.35% | 4.42% |
| 9 bits ($M=01$), 2 Stop bit | 1.82% | 2.56% | 3.90% | 4.42% |
| 7 bits ($M=10$), 2 Stop bit | 2.34% | 3.23% | 4.92% | 4.42% |

Note: The data specified in [Table 213](#) 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).

34.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. 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 LPUART 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 LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_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 LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

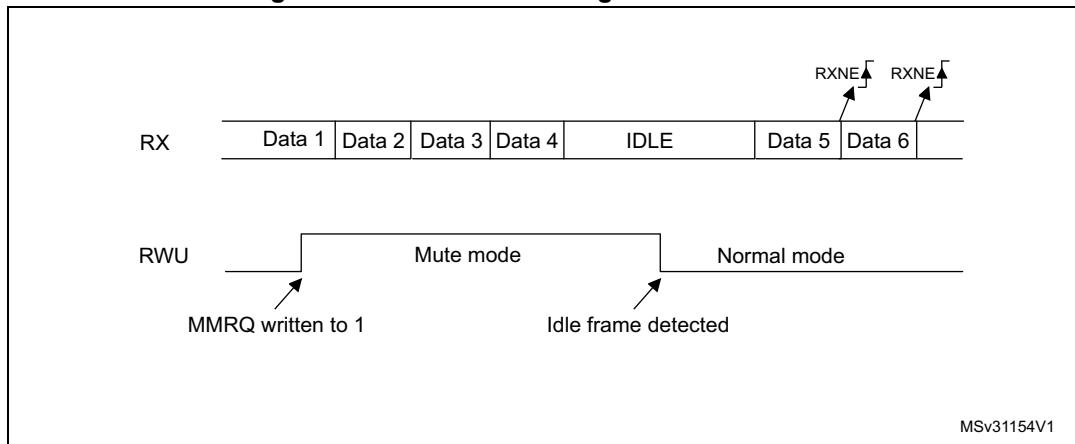
The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART 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 LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in [Figure 354](#).

Figure 354. Mute mode using Idle line detection

Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

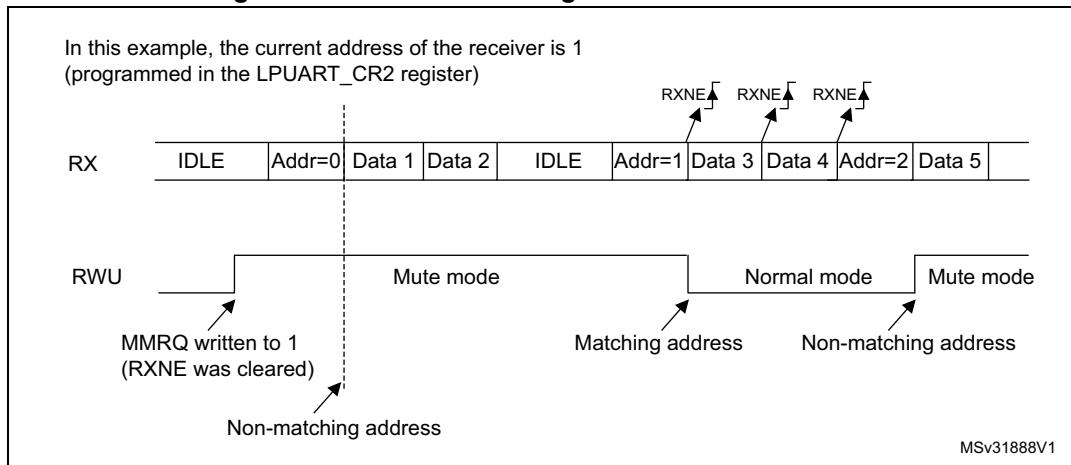
The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in [Figure 355](#).

Figure 355. Mute mode using address mark detection

34.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in [Table 214](#).

Table 214: LPUART frame formats

| M bits | PCE bit | LPUART frame ⁽¹⁾ |
|--------|---------|-----------------------------|
| 00 | 0 | SB 8 bit data STB |
| 00 | 1 | SB 7-bit data PB STB |
| 01 | 0 | SB 9-bit data STB |
| 01 | 1 | SB 8-bit data PB STB |
| 10 | 0 | SB 7bit data STB |
| 10 | 1 | SB 6-bit data PB STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECE in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

34.4.12 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any 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 is written in the data register while the TE bit is set.

Note: *In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.*

34.4.13 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: *Refer to [Section 34.3: LPUART implementation on page 1103](#) to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in [Section 34.4.7](#). To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.*

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section *Direct memory access controller*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

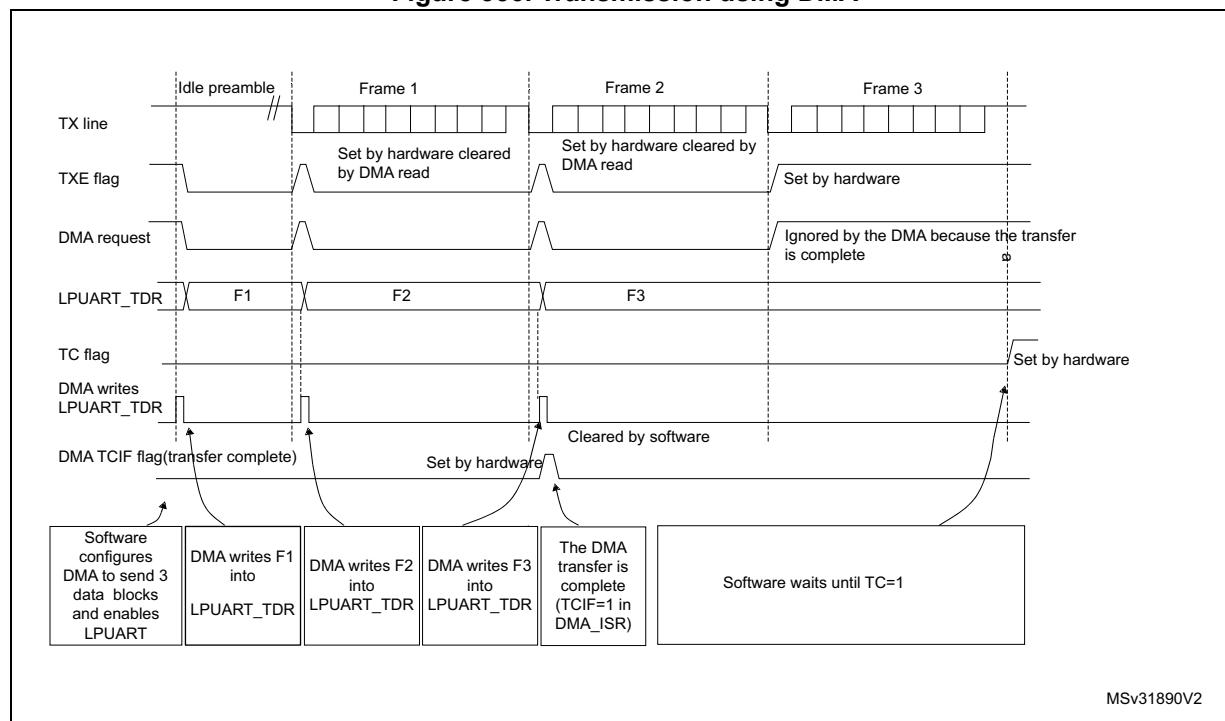
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 356. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

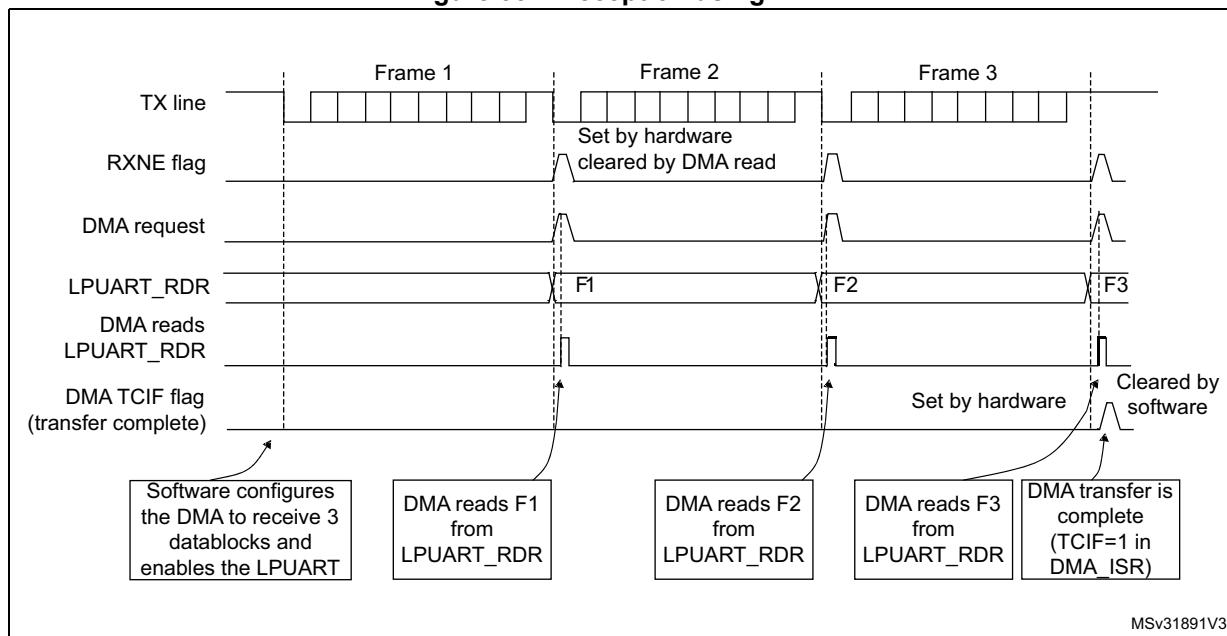
DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *Direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 357. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

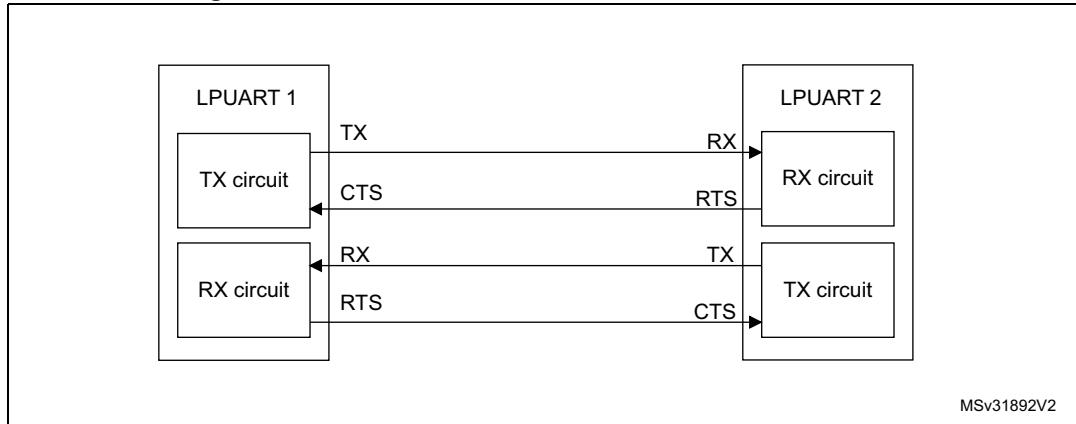
If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt

enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

34.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. [Figure 358](#) shows how to connect two devices in this mode.

Figure 358. Hardware flow control between two LPUARTs

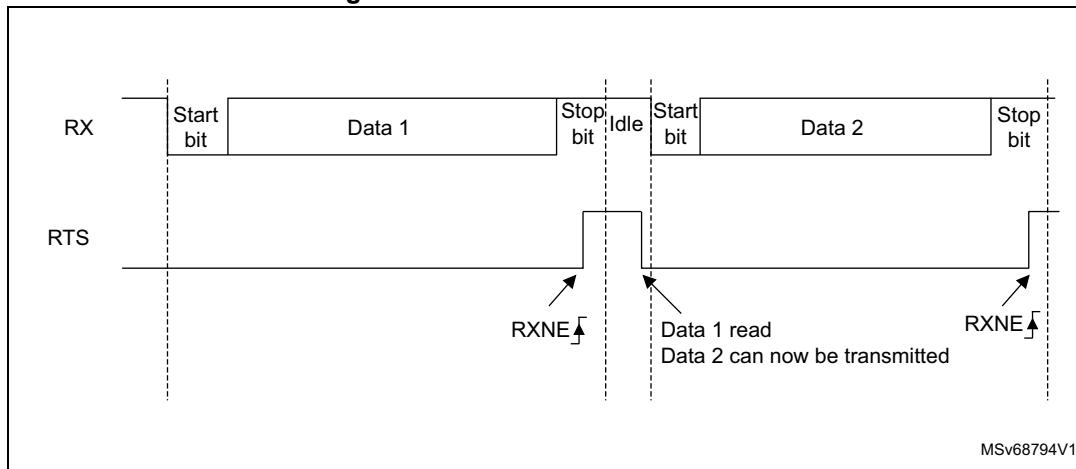


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then RTS is deasserted (tied low) as long as the LPUART 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 359](#) shows an example of communication with RTS flow control enabled.

Figure 359. RS232 RTS flow control



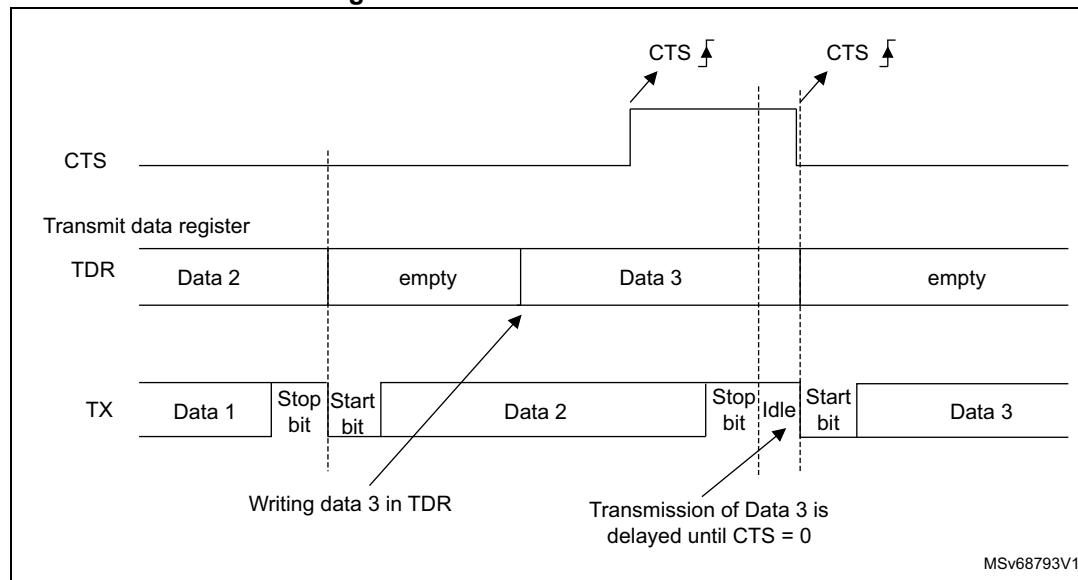
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

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 completes 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 LPUART_CR3 register is set. [Figure 360](#) shows an example of communication with CTS flow control enabled.

Figure 360. RS232 CTS flow control



Note:

For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating 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 LPUART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source (f_{CK}) cycles:

- The Driver enable assertion time equals
 - $(1 + (\text{DEAT} \times P)) \times f_{CK}$, if $P \neq 0$
 - $(1 + \text{DEAT}) \times f_{CK}$, if $P = 0$
- The Driver enable de-assertion time equals
 - $(1 + (\text{DEDT} \times P)) \times f_{CK}$, if $P \neq 0$
 - $(1 + \text{DEDT}) \times f_{CK}$, if $P = 0$

where $P = \text{BRR}[20:11]$

34.4.15 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the Ipuart_pclk clock is disabled.

The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set. When the Ipuart_pclk is gated, the LPUART provides a wake-up interrupt (**Ipuart_wkup**) if a specific action requiring the activation of the **Ipuart_pclk** clock is needed:

- If FIFO mode is disabled

Iuart_pclk clock has to be activated to empty the LPUART data register.
In this case, the Ipuart_wkup interrupt source is the RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

Iuart_pclk clock has to be activated

 - to fill the TXFIFO
 - or to empty the RXFIFO
 In this case, the Ipuart_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 Ipuart_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 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 LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **Ipuart_wkup** interrupt may be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and **Ipuart_wkup** interrupt is generated if the WUFIE bit is set.

Note: *Before entering low-power mode, make sure that no LPUART transfer is 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 in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled upon exit from low-power mode.

When FIFO is enabled, the wake-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 LPUART 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 from 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 is used with wake-up from low-power mode without any constraints (that is, 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 LPUART kernel clock Ipuart_ker_ck is OFF in low-power mode

If during low-power mode, the Ipuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the Ipuart_ker_ck clock to be switched ON thanks to the Ipuart_ker_ck_req signal. The Ipuart_ker_ck is then used for the frame reception.

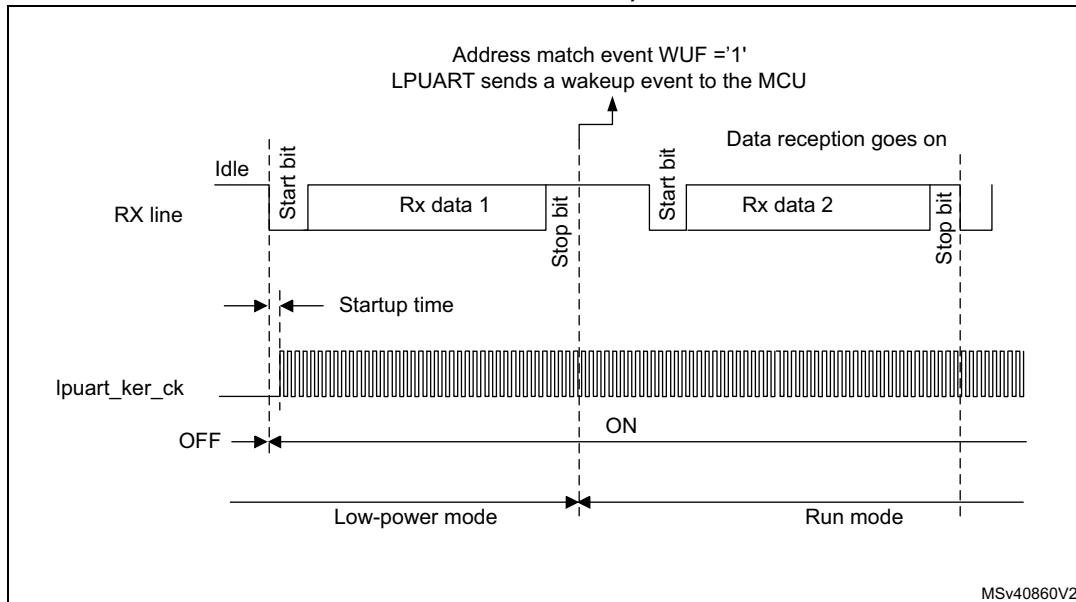
If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the Ipuart_ker_ck is switched OFF again, the MCU is not waken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 361 shows the behavior when the wake-up event is verified.

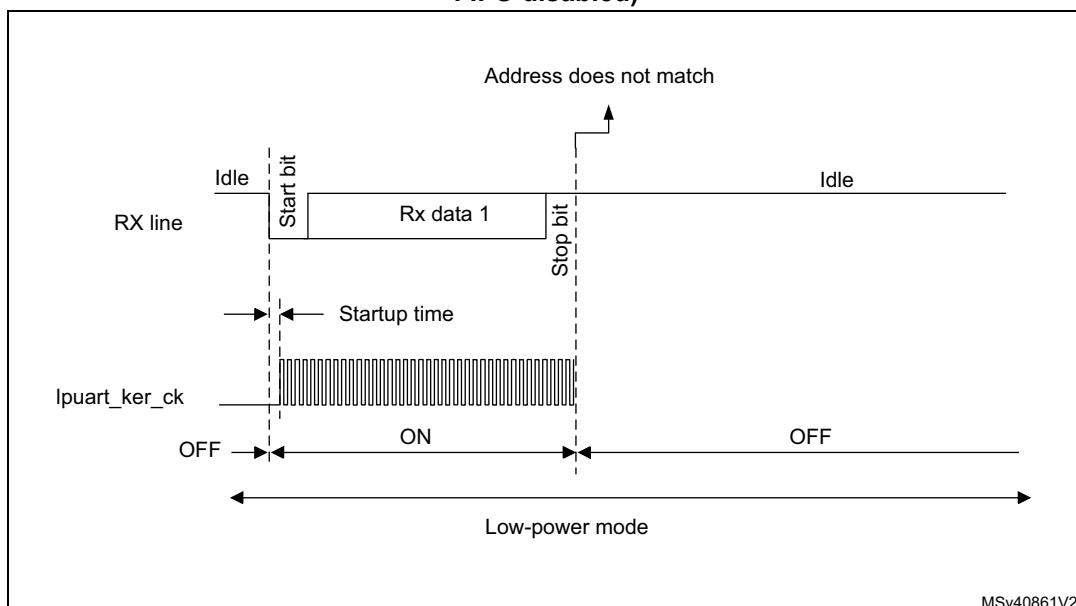
Figure 361. Wake-up event verified (wake-up event = address match, FIFO disabled)



MSv40860V2

[Figure 362](#) shows the behavior when the wake-up event is not verified.

Figure 362. Wake-up event not verified (wake-up event = address match, FIFO disabled)



MSv40861V2

Note:

The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see [Section 34.4.9: Tolerance of the LPUART 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 213: Tolerance of the LPUART receiver](#), the LPUART receiver tolerance equals 3.41%.

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

$$D_{WU\max} = t_{WULPUART} / (11 \times T_{bit\ Min})$$

$$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WU\max})$$

where $t_{WULPUART}$ 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 lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART} = 3 \mu s$ (values provided only as examples; for correct values, refer to the device datasheet).

$$D_{WU\max} = 3.41\% - 1\% = 2.41\%$$

$$T_{bit\ min} = 3 \mu s / (11 \times 2.41\%) = 11.32 \mu s.$$

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: $1/11.32 \mu s = 88.36$ kbauds.

34.5 LPUART in low-power modes

Table 215. Effect of low-power modes on the LPUART

| Mode | Description |
|---------------------|--|
| Sleep | No effect. LPUART interrupts cause the device to exit Sleep mode. |
| Stop ⁽¹⁾ | The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode. |
| Standby | The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode. |

- Refer to [Section 34.3: LPUART 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.

34.6 LPUART interrupts

Refer to [Table 216](#) for a detailed description of all LPUART interrupt requests.

Table 216. LPUART 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 |
|------------------|---|---------------------|--------------------|---|----------------------|-------------------------------------|------------------------|
| LPUART | 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 | |
| | Receive data register not empty (data ready to be read) | RXNE | RXNEIE | Read RDR or write 1 in RXFRQ | Yes | Yes | |
| | 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 | RX-NEIE/RX-FNEIE | Write 1 in ORECF | | No | |
| | Idle line detected | IDLE | IDLEIE | Write 1 in IDLECF | | No | |
| | Parity error | PE | PEIE | Write 1 in PECF | Yes | No | Yes |
| | 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 | |
| | Wake-up from low-power mode | WUF | WUFIE | Write 1 in WUC | | Yes | |

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to [Section 34.3: LPUART implementation](#) for the list of supported Stop modes.
2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

34.7 LPUART 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).

34.7.1 LPUART control register 1 (LPUART_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, FIFOEN = 1

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---------|--------|---------|----|------|------|-----------|------|----------|------|----------|--------|----|----|-----------|----|
| RXF FIE | TXFEIE | FIFO EN | M1 | Res. | Res. | DEAT[4:0] | | | | | | | | DEDT[4:0] | |
| rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | CMIE | MME | M0 | WAKE | PCE | PS | PEIE | TXFN FIE | TCIE | RXFN EIE | IDLEIE | TE | RE | UESM | UE |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **RXFFIE**:RXFIFO Full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated when RXFF=1 in the LPUART_ISR register

Bit 30 **TXFEIE**:TXFIFO empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated when TXFE=1 in the LPUART_ISR register

Bit 29 **FIFOEN**:FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit 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 LPUART is disabled (UE=0).

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.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in Ipuart_ker_ck clock cycles. For more details, refer [Section 33.5.21: RS232 hardware flow control and RS485 driver enable](#).

This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in Ipuart_ker_ck clock cycles. For more details, refer [Section 34.4.14: RS232 hardware flow control and RS485 driver enable](#).

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit 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 LPUART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART 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 LPUART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

- 0: Parity control disabled
- 1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

- 0: Even parity
- 1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever ORE=1 or RXFNE=1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

- 0: Transmitter is disabled
- 1: Transmitter is enabled

Note: During transmission, a 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 LPUART_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**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART 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 exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

34.7.2 LPUART control register 1 [alternate] (LPUART_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, FIFOEN = 0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|---------|----|------|------|-----------|------|-------|------|--------|--------|-----------|----|------|----|
| Res. | Res. | FIFO EN | M1 | Res. | Res. | DEAT[4:0] | | | | | | DEDT[4:0] | | | |
| | | 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. | 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.

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 LPUART is disabled (UE=0).

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.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer [Section 33.5.21: RS232 hardware flow control and RS485 driver enable](#).

This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart_ker_ck clock cycles. For more details, refer [Section 34.4.14: RS232 hardware flow control and RS485 driver enable](#).

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit 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 LPUART is disabled (UE=0).

Bit 11 WAKE: Receiver wake-up method

This bit determines the LPUART 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 LPUART is disabled (UE=0).

Bit 10 PCE: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 PS: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8 PEIE: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 TXEIE: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE =1 in the LPUART_ISR register

Bit 6 TCIE: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 RXNEIE: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE=1 or RXNE=1 in the LPUART_ISR register

Bit 4 IDLEIE: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a 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 LPUART_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**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART 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 exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

34.7.3 LPUART control register 2 (LPUART_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] | | | | | | | | Res. | Res. | Res. | Res. | MSBFI RST | DATAIN V | TXINV | RXINV |
| rw | rw | rw | rw | rw | rw | rw | rw | | | | | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWAP | Res. | STOP[1:0] | | Res. | ADDM7 | Res. | Res. | Res. | Res. |
| rw | | rw | rw | | | | | | | | rw | | | | |

Bits 31:24 ADD[7:0]: Address of the LPUART node

These bits give the address of the LPUART 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 LPUART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 MSBFIRST: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 18 DATAINV: Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)

1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 17 TXINV: TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels ($V_{DD} = 1/\text{idle}$, Gnd=0/mark)

1: TX pin signal values are inverted. ($(V_{DD} = 0/\text{mark}, \text{Gnd}=1/\text{idle})$).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 16 RXINV: RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels ($V_{DD} = 1/\text{idle}$, Gnd=0/mark)

1: RX pin signal values are inverted. ($(V_{DD} = 0/\text{mark}, \text{Gnd}=1/\text{idle})$).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 SWAP: Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: STOP bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: Reserved.

10: 2 stop bits

11: Reserved

This bitfield can only be written when the LPUART is disabled (UE=0).

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

Bit 4 **ADDM7:7-bit Address Detection/4-bit Address Detection**

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the LPUART is disabled (UE=0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

34.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-----|-----|------|--------------|--------|-------|--------------|------|------|--------|-------|------|-------|------|------|------|
| | | | TXFTCFG[2:0] | RXFTIE | | RXFTCFG[2:0] | | Res. | TXFTIE | WUFIE | WUS1 | WUS0 | Res. | Res. | Res. |
| 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 | OVRDIS | Res. | CTSIE | CTSE | RTSE | DMAT | DMAR | Res. | Res. | HDSEL | Res. | Res. | EIE |
| 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.
- 110: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.

Others: Reserved, must not be used.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration

- 000:Receive FIFO reaches 1/8 of its depth.
- 001:Receive FIFO reaches 1/4 of its depth.
- 110: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.

Others: Reserved, must not be used.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: LPUART interrupt generated whenever WUF=1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 34.3: LPUART implementation on page 1103](#).

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 LPUART is disabled (UE=0).

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 34.3: LPUART implementation on page 1103](#).

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode

This bit 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 LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **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 LPUART_RDR register.

This bit can only be written when the LPUART is disabled (UE=0).

Note: This control bit enables checking the communication flow w/o reading the data.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes 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 LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 DMAT: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 DMAR: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 HDSEL: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the LPUART is disabled (UE=0).

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

Bit 0 EIE: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NE=1 in the LPUART_ISR register).

0: Interrupt is inhibited

1: An interrupt is generated when FE=1 or ORE=1 or NE=1 in the LPUART_ISR register.

34.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 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. | WUFIE | WUS1 | WUS0 | Res. | Res. | Res. | Res. |
| | | | | | | | | | rw | rw | rw | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DEP | DEM | DDRE | OVRDIS | Res. | CTSIE | CTSE | RTSE | DMAT | DMAR | Res. | Res. | HDSEL | Res. | Res. | EIE |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | | | rw | | | rw |

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: LPUART interrupt generated whenever WUF=1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 34.3: LPUART implementation on page 1103](#).

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 LPUART is disabled (UE=0).

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 34.3: LPUART implementation on page 1103](#).

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode

This bit 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 LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **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 LPUART_RDR register.

This bit can only be written when the LPUART is disabled (UE=0).

Note: This control bit enables checking the communication flow w/o reading the data.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes 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 LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the LPUART is disabled (UE=0).

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

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NE=1 in the LPUART_ISR register).

0: Interrupt is inhibited

1: An interrupt is generated when FE=1 or ORE=1 or NE=1 in the LPUART_ISR register.

34.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 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. | BRR[19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BRR[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

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

Bits 19:0 **BRR[19:0]**: LPUART baud rate division (LPUARTDIV)

Note: *It is forbidden to write values lower than 0x300 in the LPUART_BRR register.*

Provided that LPUART_BRR must be $\geq 0x300$ and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].

34.7.7 LPUART request register (LPUART_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | TXFRQ | RXFRQ | MMRQ | SBKRQ | Res. |
| | | | | | | | | | | | w | w | w | w | |

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

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: *In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.*

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

34.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|-------|------|-------|-------|-------|------|-----|------|-----|------|
| Res. | Res. | Res. | Res. | TXFT | RXFT | Res. | RXFF | TXFE | REACK | TEACK | WUF | RWU | SBKF | CMF | BUSY |
| | | | | r | r | | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | CTS | CTSIF | Res. | TXFNF | TC | RXFNE | IDLE | ORE | NE | FE | PE |
| | | | | | r | r | | r | r | r | r | r | r | r | r |

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

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG in LPUART_CR3 register, that is, the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the LPUART_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 RXFIFO reaches the threshold programmed in RXFTCFG in LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register).

An interrupt is generated if the RXFFIE bit =1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit =1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: 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 LPUART_ICR register.

An interrupt is generated if WUFIIE=1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 34.3: LPUART implementation on page 1103.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART 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 LPUART_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 LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1 in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

- 0: LPUART is idle (no reception)
- 1: reception ongoing

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

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the 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 LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_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 Reserved, must be kept at reset value.

Bit 7 **TXFNF**: TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

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 must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit =1 in the LPUART_CR1 register.

- 0: Data register is full/Transmit FIFO is full.
- 1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by a write to the LPUART_TDR register.

- 0: Transmission has not completed
- 1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 RXFNE: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE=1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 IDLE: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 ORE: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE=1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the 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.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE:** Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register. 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 LPUART_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 LPUART_RDR.

Bit 0 **PE:** Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECE bit in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

34.7.9 LPUART interrupt and status register [alternate] (LPUART_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, FIFOEN = 0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|-------|------|------|-------|-------|------|-----|------|-----|------|
| Res. | Res. | Res. | REACK | TEACK | WUF | RWU | SBKF | CMF | BUSY |
| | | | | | | | | | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | CTS | CTSIF | Res. | TXE | TC | RXNE | IDLE | ORE | NE | FE | PE |
| | | | | | r | r | | r | r | r | r | r | r | r | r |

Bits 31:23 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 LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 TEACK: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 WUF: 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 LPUART_ICR register. An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 34.3: LPUART implementation on page 1103.

Bit 19 RWU: Receiver wake-up from mute mode

This bit indicates if the LPUART 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 LPUART_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 LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 SBKF: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 CMF: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1 in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 BUSY: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception ongoing

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

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the 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 LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_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 Reserved, must be kept at reset value.

Bit 7 **TXE**: Transmit data register empty

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit =1 in the LPUART_CR1 register.

0: Data register full

1: Data register empty

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE=1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE=1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

0: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE**: Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register. 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 LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECE bit in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

34.7.10 LPUART interrupt flag clear register (LPUART_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. | 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. | Res. | Res. | Res. | Res. | CTSCF | Res. | Res. | TCCF | Res. | IDLECF | ORECF | NECF | FECF | PECF |
| | | | | | | 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 LPUART_ISR register.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 34.3: LPUART implementation on page 1103.

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 LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag

Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag

Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag

Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

34.7.11 LPUART receive data register (LPUART_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 |
| RDR[8:0] | | | | | | | | | | | | | | | |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 [Section 34.4.1: LPUART block diagram](#)).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

34.7.12 LPUART transmit data register (LPUART_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. | Res. | rw |
| TDR[8:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

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

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see [Section 34.4.1: LPUART block diagram](#)).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF=1.

34.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART 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. | Res. | Res. | Res. | Res. |
| PRESCALER[3:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

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

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

- 0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256

Note: When PRESCALER is programmed with a value different of the allowed ones, the timer will be disabled. If 10110 is programmed, the timer will be disabled. If 00000 is programmed, the timer will be enabled.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 21.2: LPUART implementation on page 1102.

34.7.14 IPUART register map

Table 217 | PUART register map and reset values

Table 217. LPUART 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 | | | |
|-----------|----------------------------------|----------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|--|--|--|
| 0x10-0x14 | | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x18 | LPUART_RQR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1C | LPUART_ISR FIFO mode enabled | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1C | LPUART_ISR FIFO mode disabled | Res. | Res. | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x20 | LPUART_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. | Res. | Res. | Res. | Res. | Res. | Res. | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x24 | LPUART_RDR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x28 | LPUART_TDR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2C | LPUART_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. | Res. | Res. | Res. | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Refer to [Section 2.2](#) for the register boundary addresses.

35 Serial peripheral interface (SPI)

35.1 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.

35.2 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

35.3 SPI implementation

The following table describes all the SPI instances and their features embedded in the devices.

Table 218. STM32U0 series SPI implementation

| SPI Features | SPI1 | SPI2 | SPI3 ⁽¹⁾ |
|---|-------------------|-------------------|---------------------|
| Enhanced NSSP & TI modes | Yes | Yes | Yes |
| Hardware CRC calculation | Yes | Yes | Yes |
| Data size configuration | from 4 to 16 bits | from 4 to 16 bits | from 4 to 16 bits |
| Rx/Tx FIFO size | 32 bits | 32 bits | 32 bits |
| Wake-up capability from Low-power Sleep | Yes | Yes | Yes |

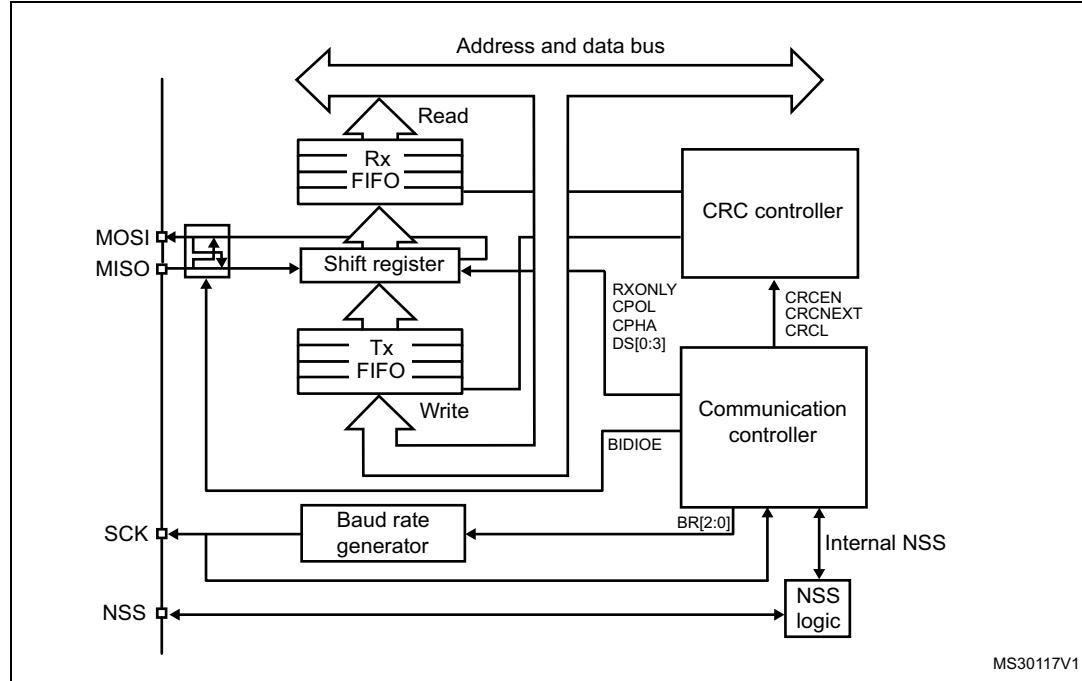
1. Applies to STM32U073xx and STM32U083xx devices only.

35.4 SPI functional description

35.4.1 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 363](#).

Figure 363. 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 35.4.5: 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.

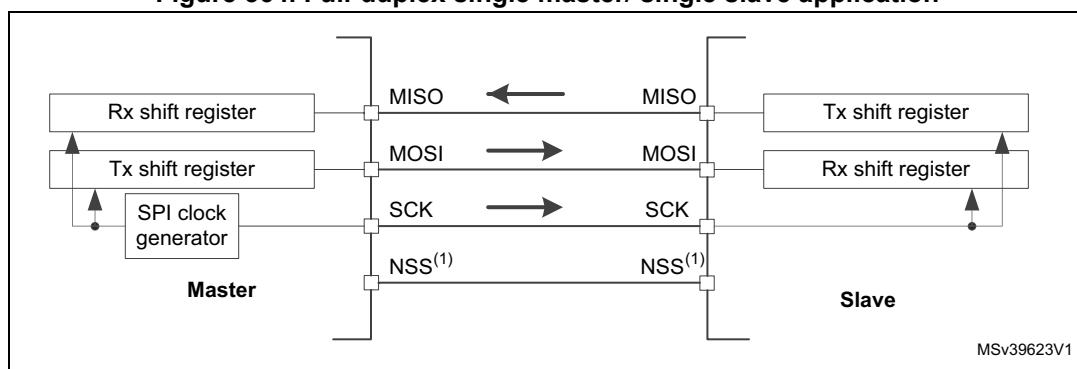
35.4.2 Communications between one master and one slave

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 364. Full-duplex single master/ single slave application

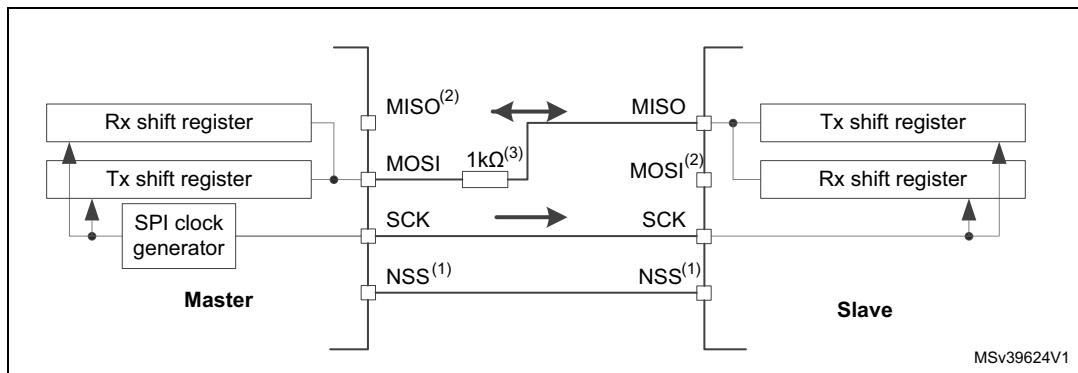


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 35.4.5: Slave select \(NSS\) pin management](#).

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 365. Half-duplex single master/ single slave application



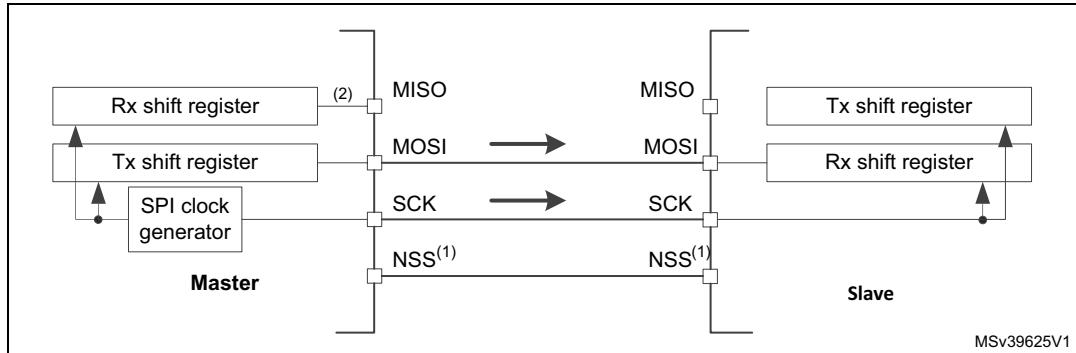
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 35.4.5: Slave select \(NSS\) pin management](#).
2. In 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 bidirectional 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 [35.4.5: 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 366. 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 35.4.5: 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.

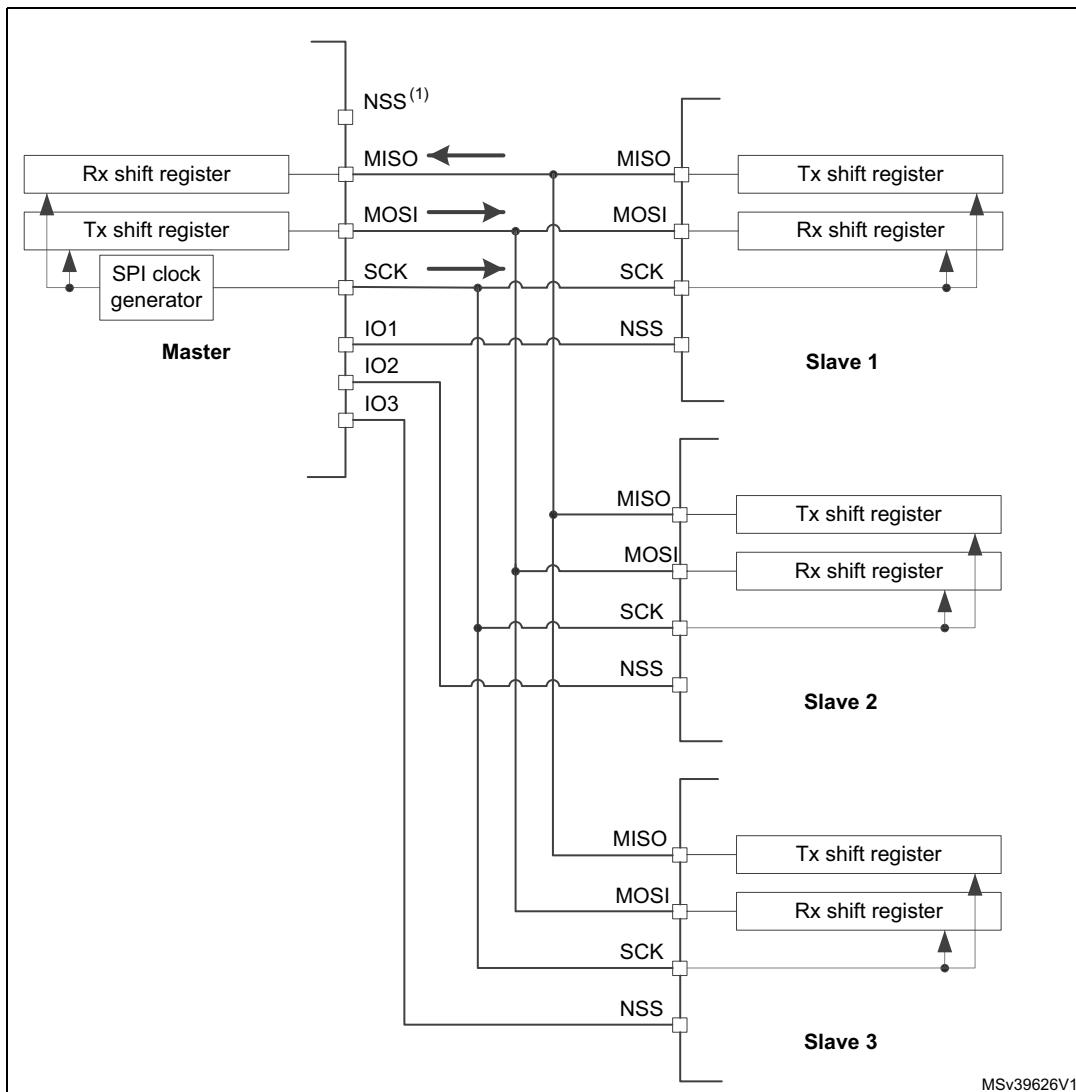
Note:

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).

35.4.3 Standard multislide 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 367](#)). 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 367. 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)).

35.4.4 Multimaster communication

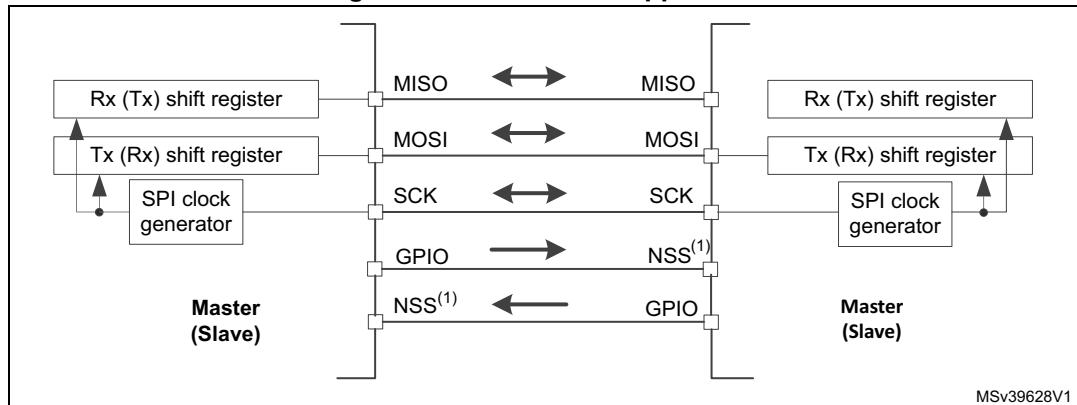
Unless SPI bus is not designed for a multimaster capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

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 368. Multimaster application



MSV39628V1

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.

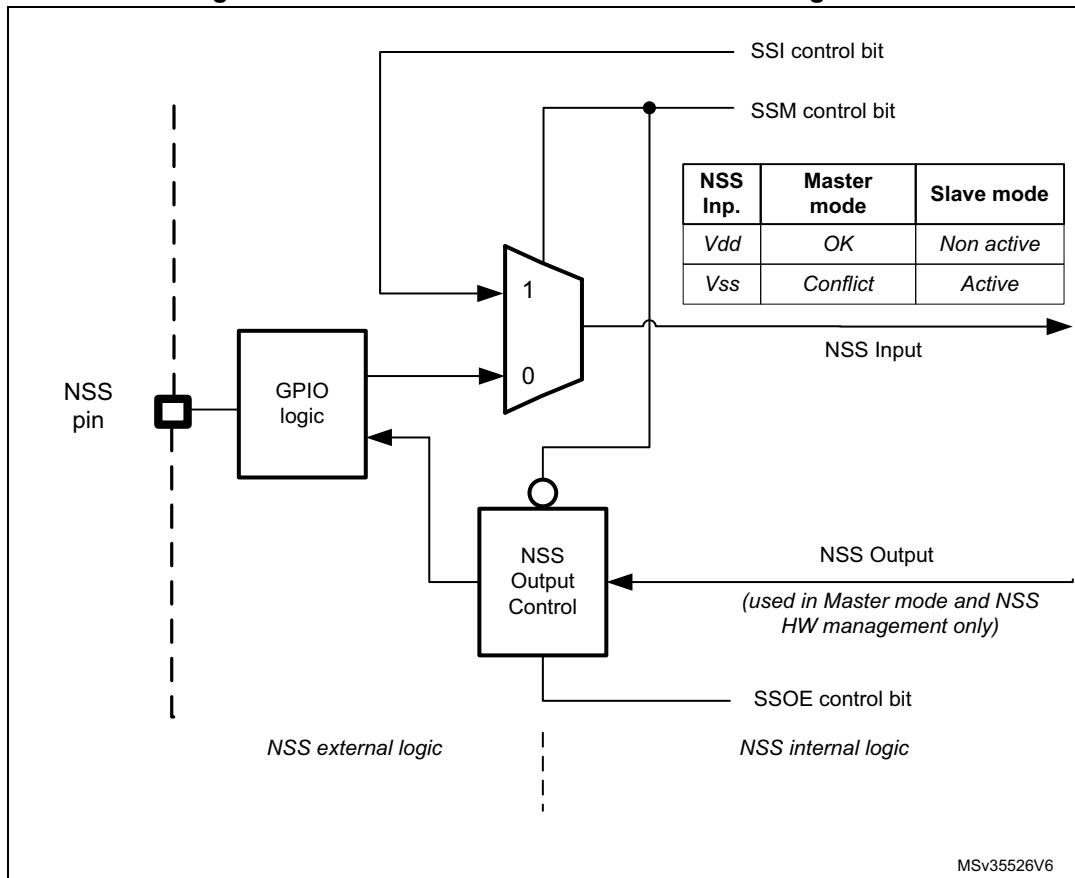
35.4.5 Slave select (NSS) pin management

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 369. Hardware/software slave select management



35.4.6 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.

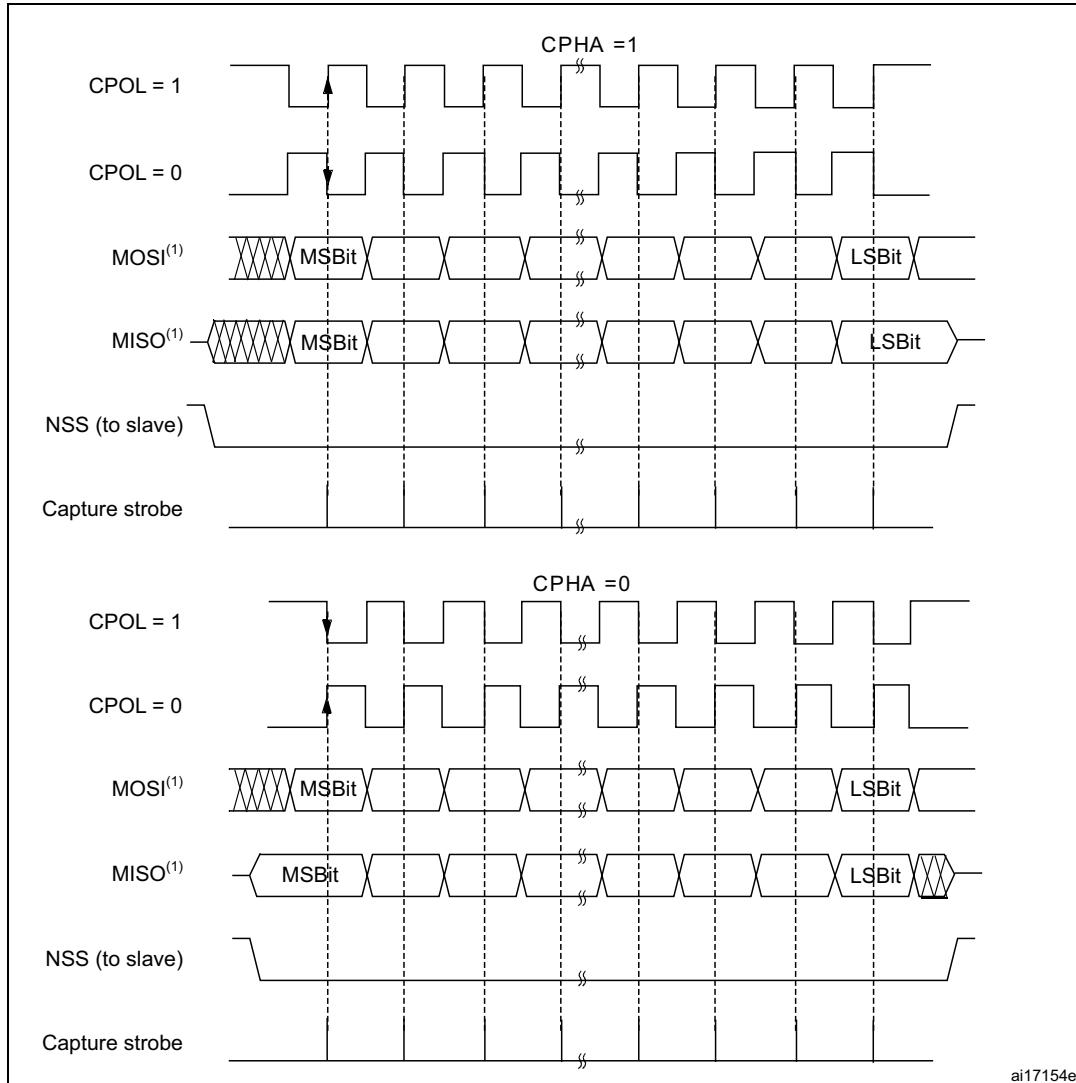
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

[Figure 370](#), 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).

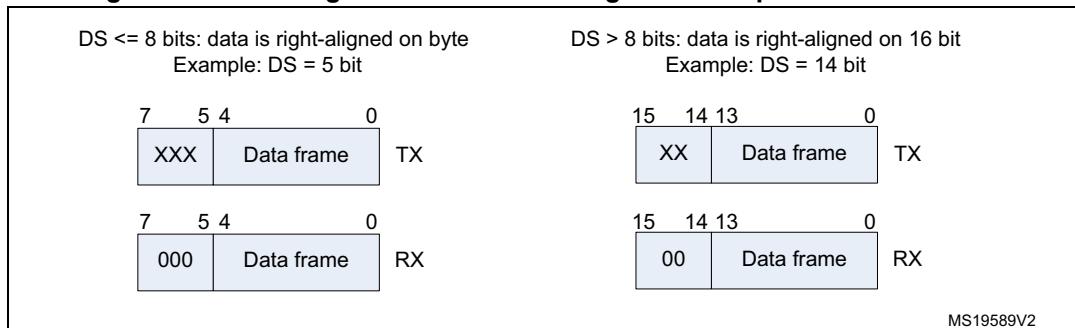
Figure 370. Data clock timing diagram



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 371](#)). During communication, only bits within the data frame are clocked and transferred.

Figure 371. 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.

35.4.7 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 RXTH 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

35.4.8 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.

35.4.9 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 35.4.14: 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 35.4.13: 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 373](#) through [Figure 376](#).

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 35.4.10: 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 multislide 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 35.4.5: 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.

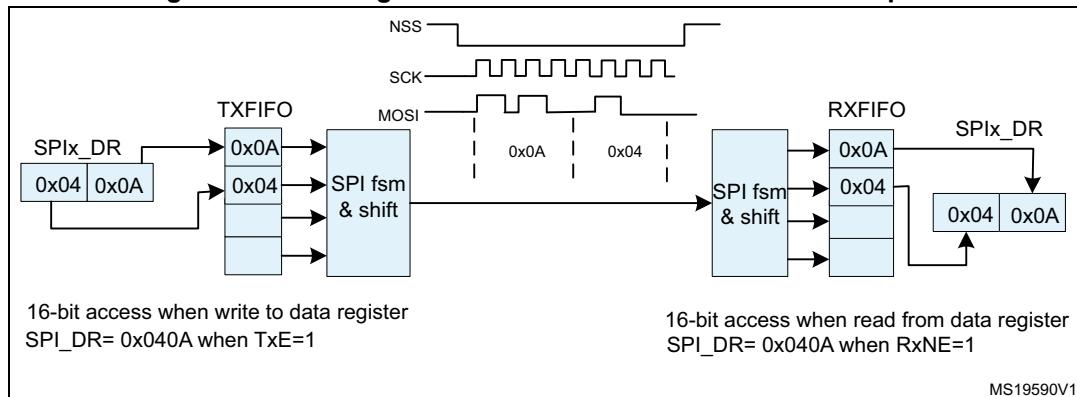
Data packing

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 372](#) 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 372. 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 373](#) through [Figure 376](#).

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 1172](#).)

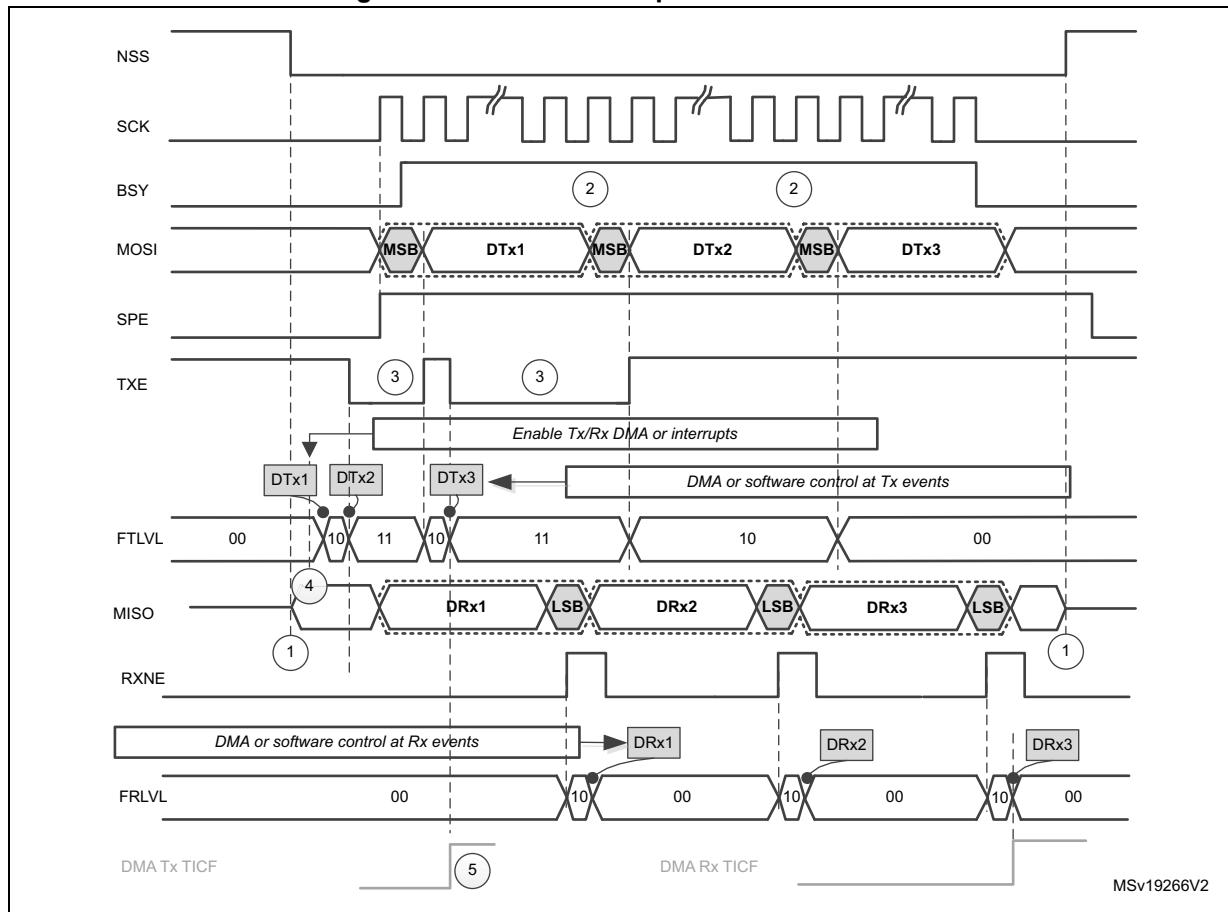
Communication diagrams

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 373 on page 1176](#) through [Figure 376 on page 1179](#):

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 Rx FIFO 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 Rx FIFO, 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 Tx FIFO is $\frac{3}{4}$ full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the Tx FIFO becomes $\frac{1}{2}$ full. This frame is stored into Tx FIFO 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 373. Master full-duplex communication



Assumptions for master full-duplex communication example:

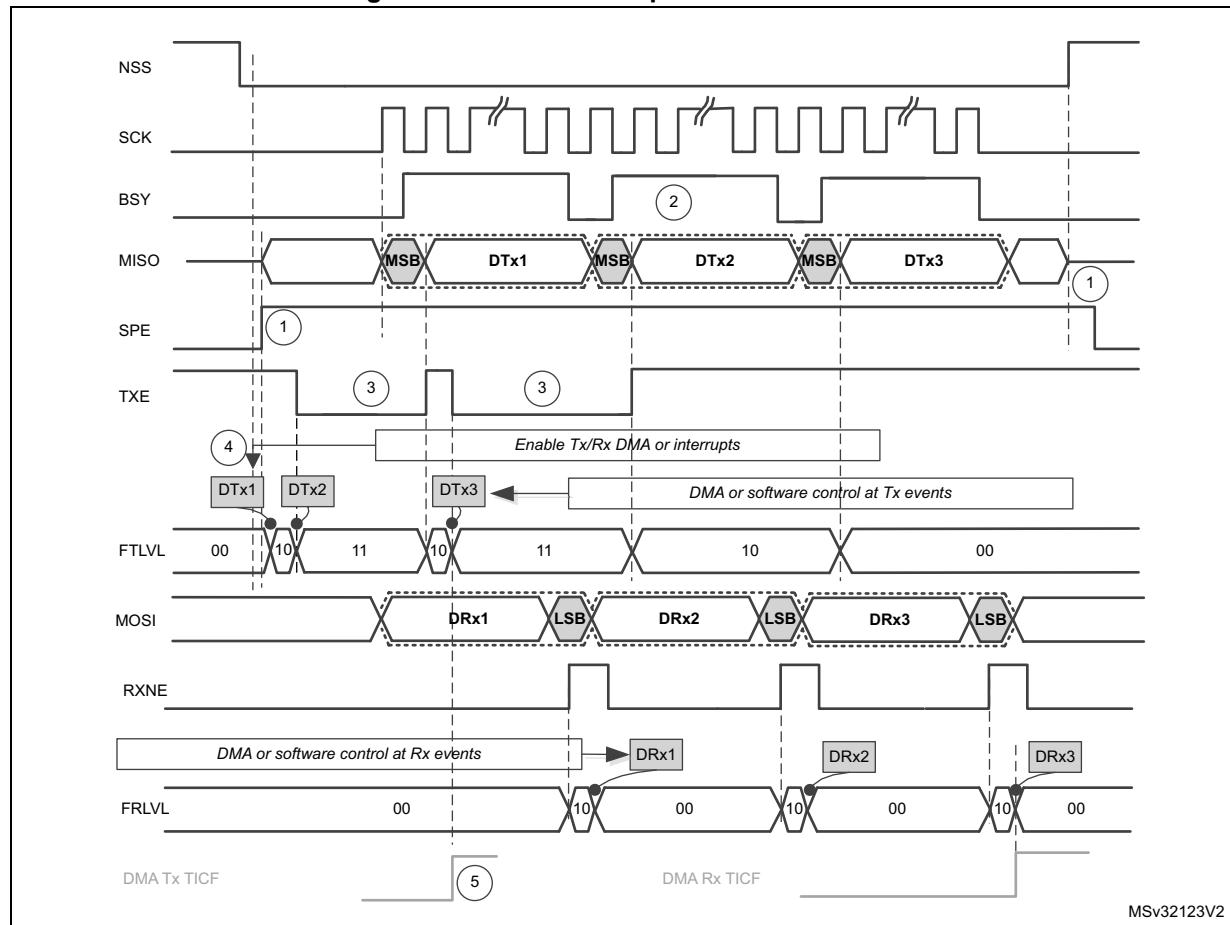
- Data size > 8 bit

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 1175](#) for details about common assumptions and notes.

Figure 374. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

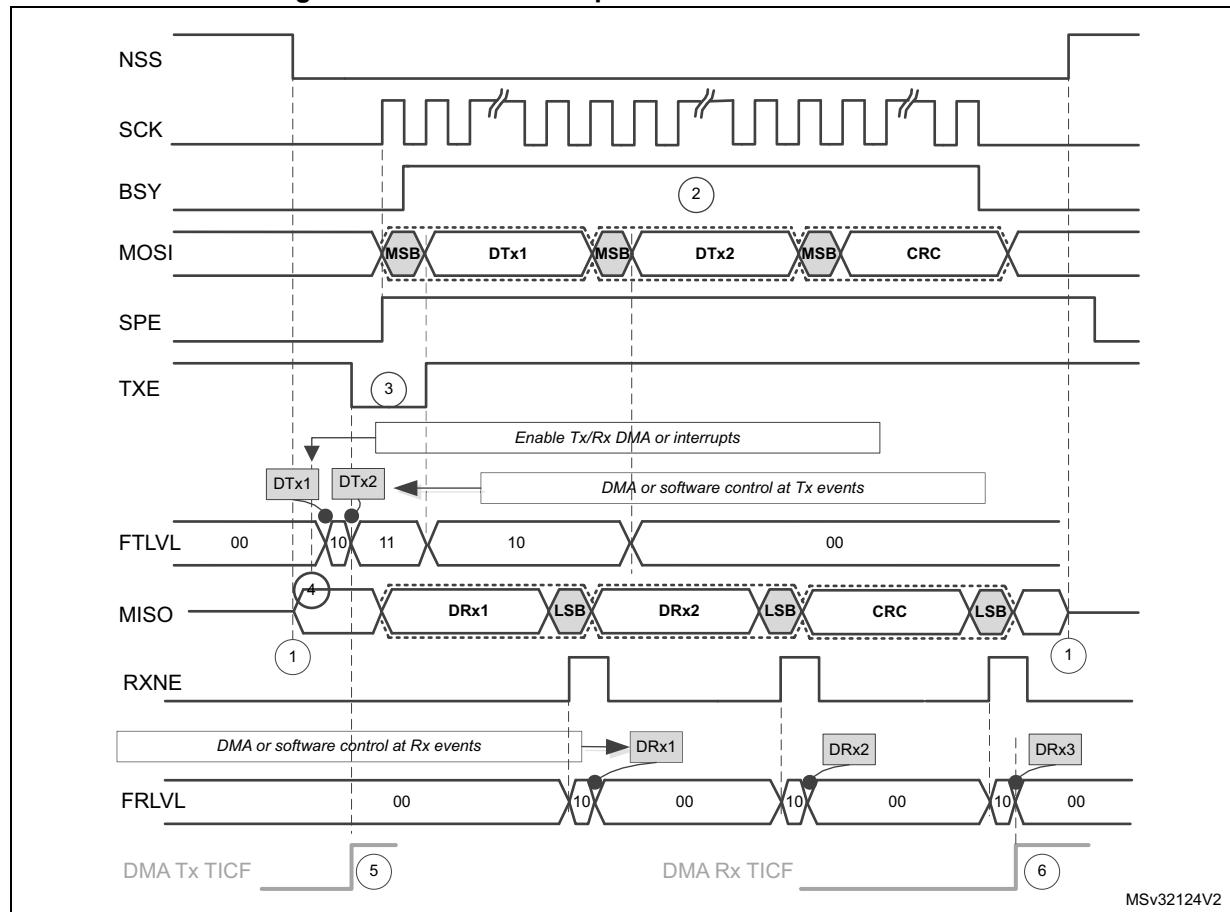
- Data size > 8 bit

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 1175](#) for details about common assumptions and notes.

Figure 375. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

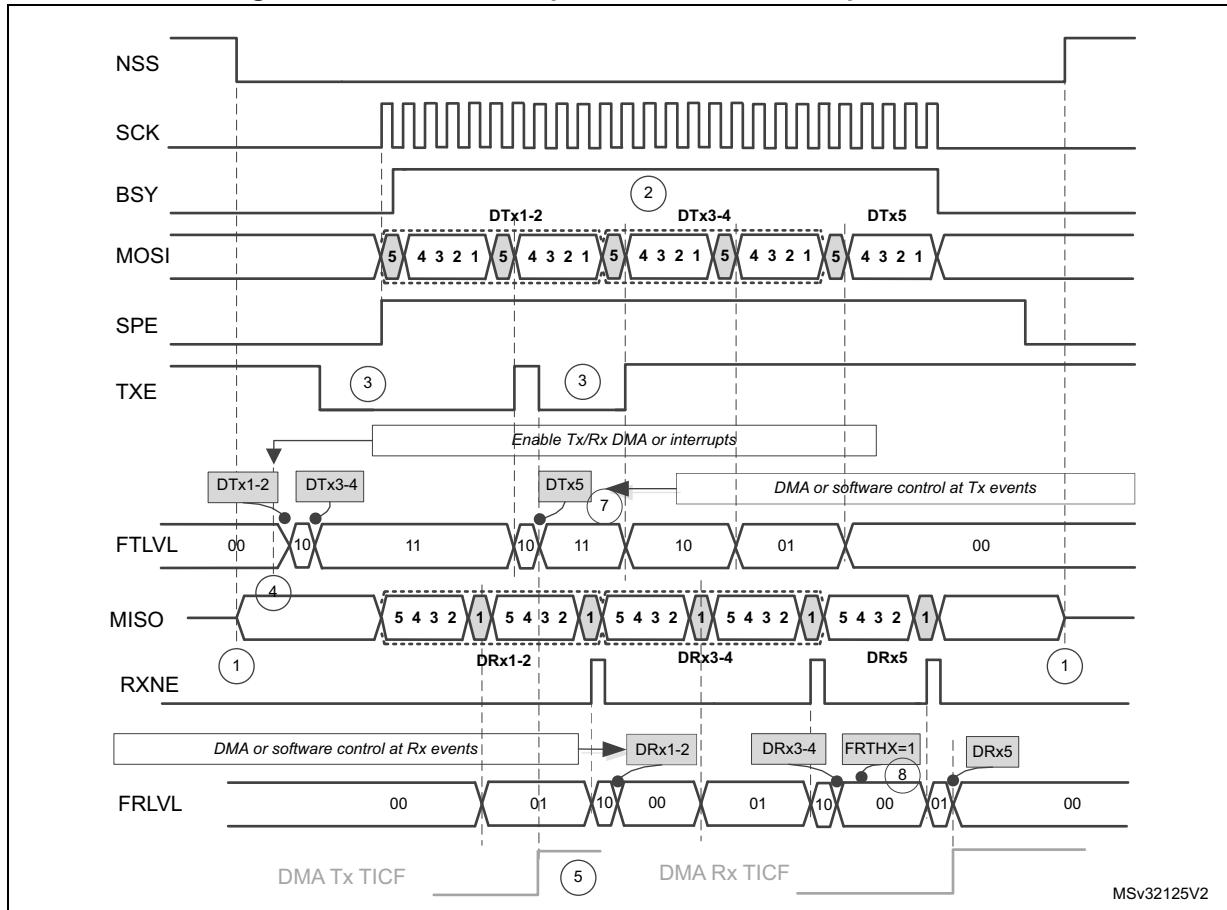
- Data size = 16 bit
- CRC enabled

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 1175](#) for details about common assumptions and notes.

Figure 376. 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 1175](#) for details about common assumptions and notes.

35.4.10 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.

35.4.11 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 35.4.14: 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.

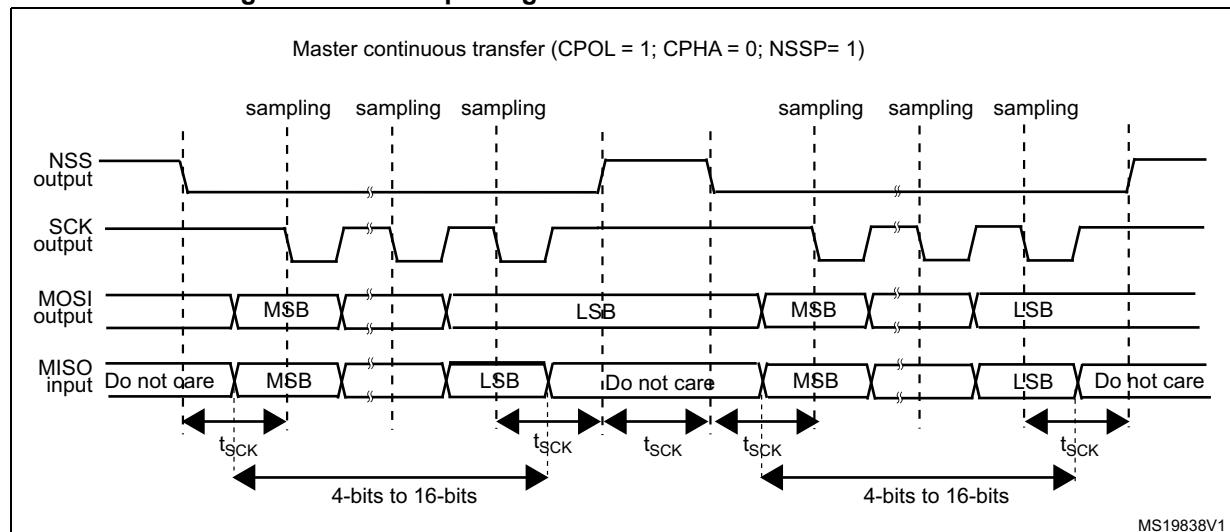
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

35.4.12 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 377 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 377. NSSP pulse generation in Motorola SPI master mode



35.4.13 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 378*). 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:

$$\frac{t_{\text{baud_rate}}}{2} + 4 \times t_{\text{pclk}} < t_{\text{release}} < \frac{t_{\text{baud_rate}}}{2} + 6 \times t_{\text{pclk}}$$

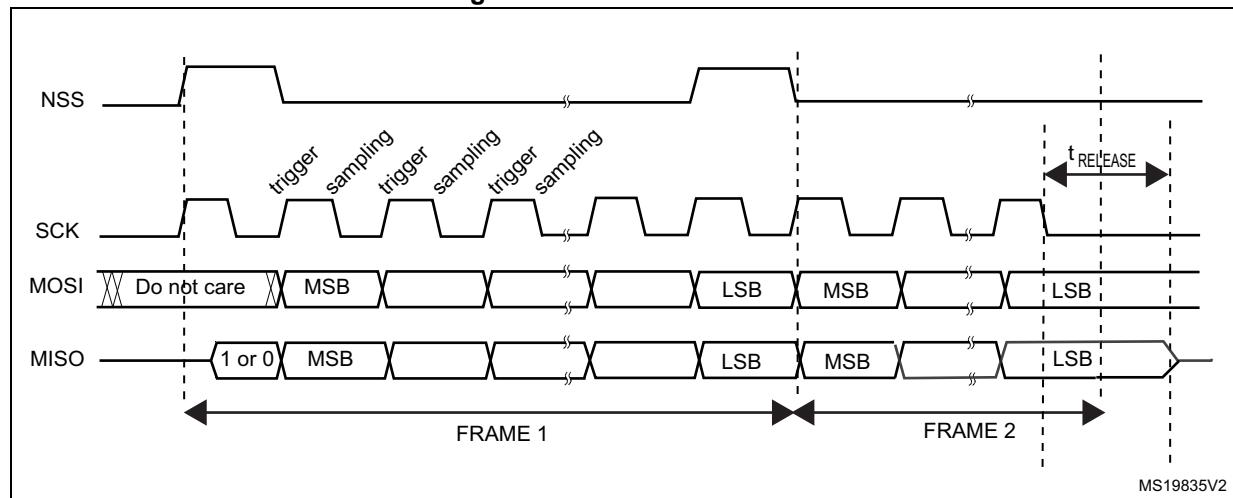
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

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 378: TI mode transfer shows the SPI communication waveforms when TI mode is selected.

Figure 378. TI mode transfer



35.4.14 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:

$$\text{DMA_RX} = \text{Numb_of_data} + 2$$

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.

35.5 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 219. 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 | |

In slave mode, the way the frame synchronization is detected, depends on the value of ASTRTEN bit.

If ASTRTEN = 0, when the audio interface is enabled (I2SE = 1), then the hardware waits for the appropriate transition on the incoming WS signal, using the CK signal.

If ASTRTEN = 1, the user has to enable the audio interface before the WS becomes active. This means that the I2SE bit must be set to 1 when WS = 1 for I²S Philips standard, or when WS = 0 for other standards.

35.6 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.

35.6.1 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 **BIDI MODE**: 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

Bit 14 **BIDIOE**: Output enable in bidirectional mode

This bit combined with the BIDI MODE 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 1171](#).

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.

35.6.2 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 | 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 1171](#) 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 1171](#) 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

35.6.3 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 | CRCE RR | 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 35.4.11: 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 35.4.10: SPI status flags and Procedure for disabling the SPI](#) on page 1171.

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 1181](#) 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

35.6.4 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 35.4.9: 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.

35.6.5 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.

35.6.6 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.

35.6.7 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.

35.6.8 SPI register map

Table 220 shows the SPI register map and reset values.

Table 220. 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 | ERRIE | FRF | NSSP | MSTR | CPOL | CPHA |
| | 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 | 0 0 0 0 | 0 0 0 0 | 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 | SPIx_CR2 | Res. LDMA_TX | Res. LDMA_RX | Res. FRXTH | DS[3:0] | | | | TXEIE | Res. RXNEIE | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| | Reset value | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 1 1 1 | 0 1 1 1 | 0 1 1 1 | 0 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 0 0 | 0 0 0 0 | 0 0 0 0 |
| 0x08 | SPIx_SR | Res. FTLVL[1:0] | Res. FRLVL[1:0] | Res. FRE | BSY | OVR | MODF | CRCERR | TXE | Res. RXNE | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 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 | 0 0 0 0 | 0 0 0 0 | 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 | SPIx_DR | DR[15:0] | | | | | | | | | | | | | | BR [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 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 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 | SPIx_CRCPR | CRCPOLY[15:0] | | | | | | | | | | | | | | SSOE | |
| | 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 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 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 | SPIx_RXCRCR | RXCRC[15:0] | | | | | | | | | | | | | | TXDMAEN | |
| | 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 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 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 | SPIx_TXCRCR | TXCRC[15:0] | | | | | | | | | | | | | | RXDMAEN | |
| | 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 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 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 55](#) for the register boundary addresses.

36 Universal serial bus full-speed device interface (USB)

This section applies to STM32U073/83 devices only.

36.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB bus.

USB suspend/resume are supported, which permits to stop the device clocks for low-power consumption.

36.2 USB main features

- USB specification version 2.0 full-speed compliant
- Supports Device mode
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 2048 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint/channel support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support (Device mode only)
- Battery Charging Specification Revision 1.2 support (Device mode only)
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB_DP line)

36.3 USB implementation

Table 221 describes the USB implementation in the devices.

Table 221. STM32U073/83 USB implementation

| USB features ⁽¹⁾ | USB |
|---|------------|
| Host mode | - |
| Number of endpoints | 8 |
| Size of dedicated packet buffer memory SRAM | 1024 bytes |
| Dedicated packet buffer memory SRAM access scheme | 32 bits |
| USB 2.0 Link Power Management (LPM) support in device | X |
| Battery Charging Detection (BCD) support for device | X |
| Embedded pull-up resistor on USB_DP line | X |

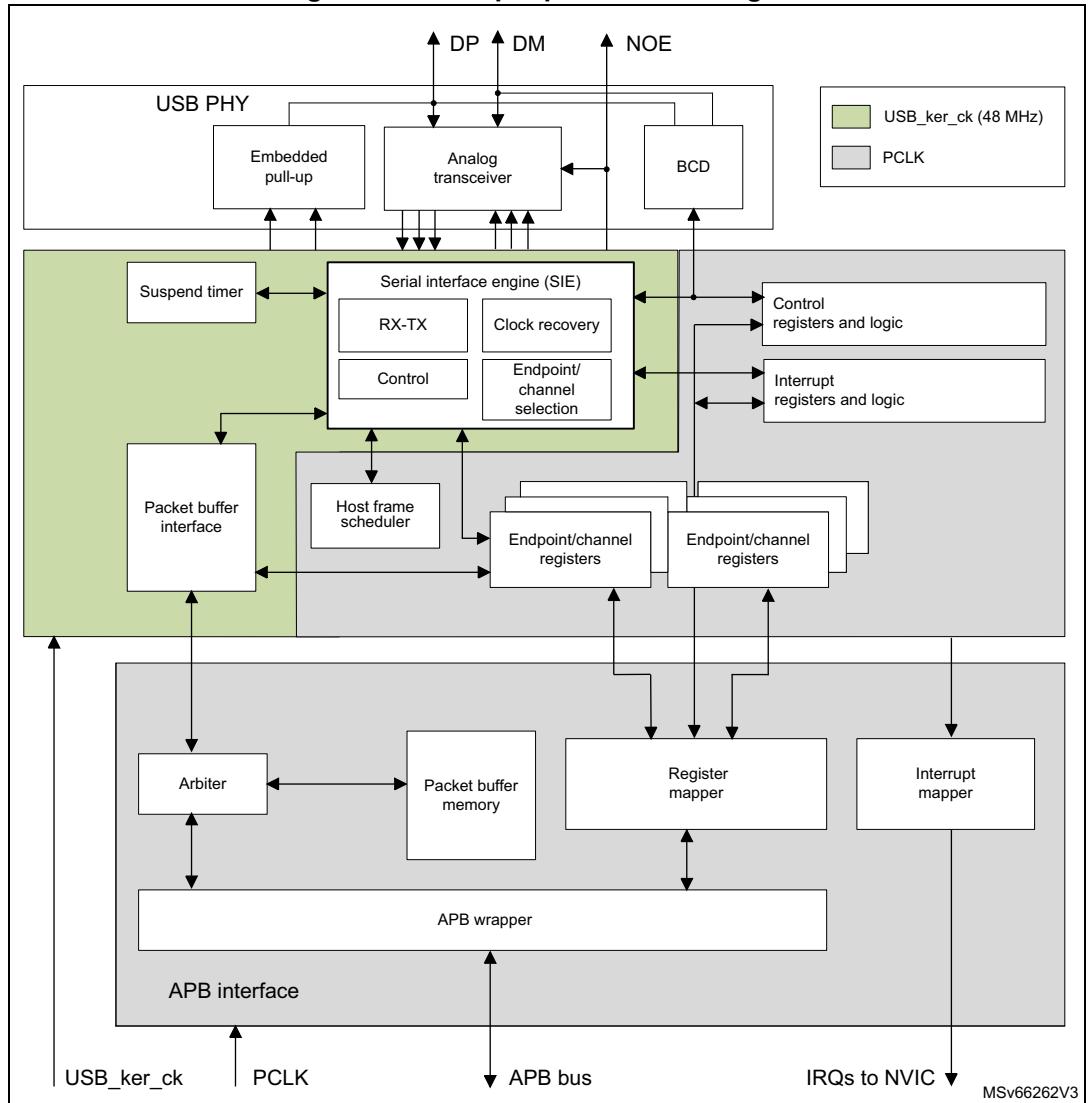
1. X= supported

36.4 USB functional description

36.4.1 USB block diagram

Figure 379 shows the block diagram of the USB peripheral.

Figure 379. USB peripheral block diagram



36.4.2 USB pins and internal signals

Table 222. USB input/output pins

| Signal name | Signal type | Description |
|-------------|----------------------|-----------------------------------|
| USB_DP | Digital input/output | D+ line |
| USB_DM | Digital input/output | D- line |
| USB_NOE | Digital output | NOE (output enable of data lines) |

36.4.3 USB reset and clocks

A single reset is present on USB. The RCC allows a reset to be forced by software.

There are two clocks:

- PCLK for the APB bus interface and registers.
- USB_ker_ck (48 MHz) for the main protocol logic including notably the serial interface engine (SIE), see USB_ker_ck clock domain in block diagram.

36.4.4 General description and Device mode functionality

The USB peripheral provides a USB-compliant connection between the function implemented by the microcontroller and an external USB function which can be a host PC but also a USB Device. Data transfer between the external USB host or device and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 2048 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the external USB Host or Device, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint/channel is associated with a buffer description block indicating where the endpoint/channel-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint/channel is configured) takes place. The data buffered by the USB peripheral are loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data have been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint/channel-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint/channel has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which permits to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

A special bit THR512 in register USB_ISTR allows notification of 512 bytes being received in (or transmitted from) the buffer. This bit must be used for long ISO packets (from 512 to 1023 bytes) as it facilitates early start or read/write of data. In this way, the first 512 bytes can be handled by software while avoiding use of double buffer mode. This bit works when only one ISO endpoint is configured.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to permit the system to immediately restart the normal clock generation and/or support direct clock start/stop.

Host mode and specific functionality

A single bit, HOST, in register USB_CNTR permits Host mode to be activated. Host mode functionality permits the USB to talk to a remote peripheral. Supported functionality is aligned to Device mode and uses the same register structures to manage the buffers. The same number of endpoints can be supported in Host mode, however in Host mode the terminology "channel" is preferred, as each channel is in reality a combination of the connected device and the endpoint on that device. The basic mechanisms for packet transmission and reception are the same as those supported in Device mode.

When operating in Host mode, the USB is in charge of the bus and in order to do this must issue transaction requests corresponding to active periodic and non-periodic endpoints. A host frame scheduler assures efficient use of the frame. Connection to hubs is supported. Connection to low speed devices is supported, both with a direct connection and through a hub.

Double-buffered mode, as previously described in Device mode, is also supported in Host mode, in both bulk and isochronous channels. The THR512 functionality is also supported (but as in Device mode) only for ISO traffic.

Note: *Unlike in Device mode, where there is a detection of battery charging capability (in order to facilitate fast charging), there is no integrated support in Host mode to present battery charging capability (CDP or DCP cases in the standard), the host port is always presented as a default standard data port (SDP).*

Note: *For LPM (link power management) this feature is not supported in Host mode.*

36.4.5 Description of USB blocks used in both Device and Host modes

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB physical interface (USB PHY): this block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB_DP line) and support for battery charging detection (BCD), multiplexed on same USB_DP and USB_DM lines. The output enable control signal of the analog transceiver (active low) is provided externally on USB_NOE. It can be used to drive some activity LED or to provide information about the actual communication direction to some other circuitry.
- Serial interface engine (SIE): the functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for

local data storage. This unit also generates signals according to USB peripheral events, such as start of frame (SOF), USB_Reset, data errors etc. and to endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.

- Timer: this block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- Packet buffer interface: this block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the endpoint/channel registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.
- Endpoint/channel-related registers: each endpoint/channel has an associated register containing the endpoint/channel type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.
- Control registers: these are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt registers: these contain the interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

Note:

* *Endpoint/channel 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB bus through an APB interface, containing the following blocks:

- Packet memory: this is the local memory that physically contains the packet buffers. It can be used by the packet buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the packet memory is 2048 bytes, structured as 512 words of 32 bits.
- Arbiter: this block accepts memory requests coming from the APB bus and from the USB interface. It resolves the conflicts by giving priority to APB accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB transfers of any length are also allowed by this scheme.
- Register mapper: this block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 32-bit wide word set addressed by the APB.
- APB wrapper: this provides an interface to the APB for the memory and register. It also maps the whole USB peripheral in the APB address space.
- Interrupt mapper: this block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

36.4.6 Description of host frame scheduler (HFS) specific to Host mode

The host frame scheduler is the hardware machine in charge to submit host channel requests on the bus according to the USB priority order and bandwidth access rules.

Host channels are divided in two categories:

- Periodic channels: isochronous and interrupt traffic types. With guaranteed bandwidth access.
- Non-periodic channels: bulk and control traffic types. With best effort service.

The host frame scheduler organizes the full-speed frame in 3 sequential windows

- Periodic service window
- Non-periodic service window
- Black security window

At the start of a new frame the host scheduler:

1. First considers all periodic channels which were active (STAT bits VALID) at the start of frame
2. Executes single round of service of periodic channels, the periodic service window, in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first
3. When the periodic round is finished, HFS closes the periodic service window and stops servicing periodic traffic even if some periodic channel was re-enabled or some new channel was enabled after the SOF.
4. Starts servicing all non-periodic channels which are currently active (STAT bits VALID) in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first.
5. Executes multiple round-robin service cycles of non-periodic channels until almost the end of frame
6. Non periodic traffic can be requested at any time and is serviced by HFS with best effort latency, with the exception of a black security window at the end of the frame where new injected requests are directly postponed to the next frame to avoid babbles. This is also true for pending transactions which have not been serviced ahead of the security window.

36.5 Programming considerations for Device and Host modes

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

36.5.1 Generic USB Device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and isochronous transfers. Apart from system reset, an action is always initiated by the USB peripheral, driven by one of the USB events described below.

36.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software must perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{STARTUP}$ specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the USBRST bit in the CNTR register). Clearing the ISTR register removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint/channel must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB bus reset (RST_DCON interrupt) in Device mode

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral does not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the enable function (EF) bit of the USB_DADDR register and initializing the CHP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RST_DCON interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of the reset sequence which triggered the interrupt.

USB bus reset in Host mode

In Host mode a bus reset is activated by setting the USBRST bit of the USB_CNTR register. It must subsequently be cleared by software once the minimum active reset time from the standard has been respected.

Structure and usage of packet buffers

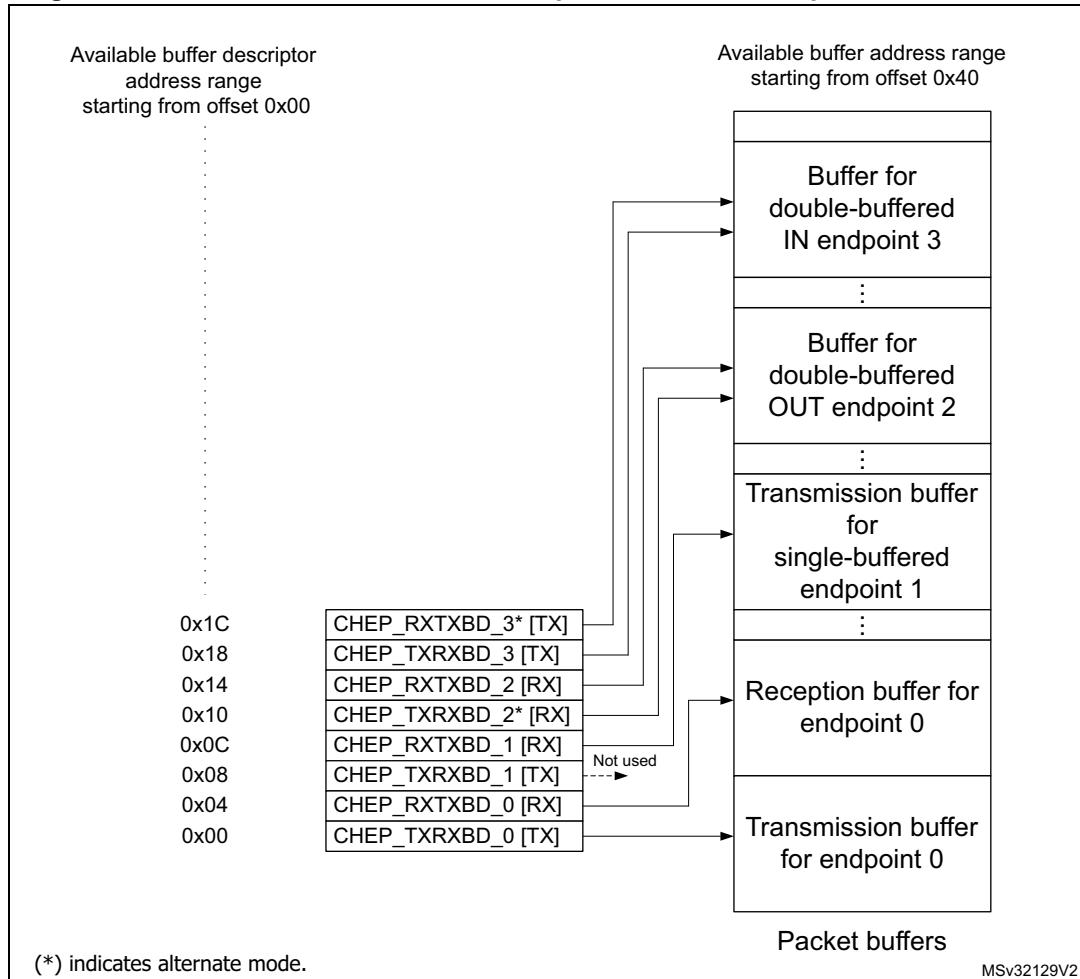
Each bidirectional endpoint may receive or transmit data over the bus. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request

and waits for its acknowledgment. Since the packet buffer memory has also to be accessed by the microcontroller, an arbitration logic takes care of the access conflicts, using half APB cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both agents can operate as if the packet memory would be a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB bus. Different clock configurations are possible where the APB clock frequency can be higher or lower than the USB peripheral one.

Note: *Due to USB data rate and packet memory interface requirements, the APB clock must have a minimum frequency of 12 MHz to avoid data overrun/underrun problems.*

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory. Each table entry is associated to an endpoint register and it is composed of two 32-bit words so that table start address must always be aligned to an 8-byte boundary. Buffer descriptor table entries are described in [Section 36.6.2: USBSRAM registers](#). If an endpoint is unidirectional and it is neither an isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to [Section 36.5.5: Isochronous transfers in Device mode](#) and [Section 36.5.3: Double-buffered endpoints and usage in Device mode](#) respectively). The relationship between buffer description table entries and packet buffer areas is depicted in [Figure 380](#).

For Host mode different sections explain the buffer usage model, notably [Section 36.5.6: Isochronous transfers in Host mode](#) and [Section 36.5.4: Double buffered channels: usage in Host mode](#).

Figure 380. Packet buffer areas with examples of buffer description table locations

Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral never changes the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data is copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX fields in the CHEP_TXBD_n and CHEP_RXBD_n registers (in SRAM) so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The UTYPE bits in the USB_CHEPnR register must be set according to the endpoint type, eventually using the EPKIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STATTX bits in the USB_CHEPnR register and COUNTn_TX must be initialized. For reception, STATRX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BLSIZE and NUM_BLOCK fields. Unidirectional endpoints, except isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_CHEPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), must not be modified by the application software, as the hardware can

change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

Data transmission in Device mode (IN packets)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of CHEP_TXBD_n (fields ADDRn_TX and COUNTn_TX) inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16-bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (refer to [Structure and usage of packet buffers on page 1201](#)) and the USB peripheral starts sending a DATA0 or DATA1 PID according to USB_CHEPnR bit DTOGTX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STATTX bits in the USB_CHEPnR register.

The ADDRn_TX field in the internal register CHEP_TXBD_n is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/4 words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled, the endpoint is made invalid by setting STATTX = 10 (NAK) and bit VTTX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. Servicing of the VTTX event starts, clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STATTX to 11 (VALID) to re-enable transmission. While the STATTX bits are equal to 10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

Data transmission in Host mode (OUT packets)

Data transmission in Host mode follows the same general principles as Device mode. The main differences are due to the protocol. For example the host initiates the transmission whereas the device responds to the incoming token.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents of an OUT packet are then written to that address in the packet memory and COUNTn_TX must be updated (when necessary) to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATTX must be set to 11 (VALID) in order to trigger the transmit. The transmission is then scheduled by the HFS.

After a successful transmission the CTR interrupt (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the

indicated channel, the STATTX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATTX is now in NAK. In the case of a STALL response, STATTX is in STALL. In this last case, the bus must be reset.

On receiving the ACK receipt by the device, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled.

An error condition is signaled via the bits VTTX and ERR_TX in the case of:

- No handshake being received in time
- False EOP
- Bit stuffing error
- Invalid handshake PID

Data reception in Device mode (OUT and SETUP packets)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX fields inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX field is stored directly in its internal register ADDR. Internal register COUNT is now reset and the values of BLSIZE and NUM_BLOCK bit fields, which are read within USB_CHEP_RXBD_n content, are used to initialize BUF_COUNT, an internal 16-bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but the ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STATRX in the USB_CHEPnR register, and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, or up to the last allocated memory location, as defined by BLSIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer description table entry, leaving unaffected BLSIZE and NUM_BLOCK fields, which normally

do not require to be re-written, and the USB_CHEPnR register is updated in the following way: DTOGRX bit is toggled, the endpoint is made invalid by setting STATRX = 10 (NAK) and bit VTRX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. The VTRX event is serviced by first determining the transaction type (SETUP bit in the USB_CHEPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being processed. After the received data is processed, the application software must set the STATRX bits to 11 (VALID) in the USB_CHEPnR, enabling further transactions. While the STATRX bits are equal to 10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Data reception in Host mode (IN packets)

Data reception in Host mode follows the same general principles as Device mode. The main differences are again due to the protocol. In the device, data can be received or not, depending on readiness after previous operations, whereas the host only requests receive data when it is ready and able to store them.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents received in the data phase response to the IN token packet are then written to that address in the packet memory and COUNTn_TX gets updated by hardware during this process to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATRX must be set to VALID in order to trigger the reception. The reception is then scheduled by the HFS.

After a successful reception the interrupt CTR (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATRX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATRX now is in NAK. In the case of a STALL response, STATRX is in STALL. In this last case, the bus must be reset. During an IN packet an error condition is signaled via the bits VTRX and ERR_RX in case of:

- False EOP
- Bit stuffing error
- Wrong CRC

Control transfers in Device mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STATTX and STATRX are set to 10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_CHEPnR register at each VTRX event to distinguish normal OUT transactions from SETUP ones. A USB Device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is

required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction must be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction must be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software changes NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS_OUT (EPKIND in the USB_CHEPnR register) bit must be set, so that an error is generated if a status transaction is performed with non-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STATRX to VALID (to accept a new command) and STATTX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic does not permit a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STATRX bits are set to 01 (STALL) or 10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued VTRX request not yet acknowledged by the application (for example VTRX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the VTRX interrupt.

Control transfers in Host mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only. A control endpoint must set the SETUP bit in the USB_CHEPnR register. The values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 0. Depending on whether it is a control write or control read then STATTX or STATRX are set to 11 (ACTIVE) in order to trigger the control transfer via the host frame scheduler.

On receiving a CTR interrupt the channel (device address and endpoint) can be determined by examining IDN and DIR bits. Devices are expected to NAK every control unless the packet is corrupted in which case they do not acknowledge. The situation is reflected in the value of STATTX.

In the case of an error condition the ERR bit gets set. One possible case is where a CRC error is seen at the device, in this case no ACK is returned to the host. The host sees no ACK and after an appropriate delay this generates a timeout error with ERR_TX set (which can generate an interrupt).

36.5.3 Double-buffered endpoints and usage in Device mode

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it answers with a NAK handshake and the host PC issues the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_CHEPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from 00 (DISABLED): STATRX if the double-buffered bulk endpoint is enabled for reception, STATTX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOGRX (bit 14 of USB_CHEPnR register) for ‘reception’ double-buffered bulk endpoints or DTOGTX (bit 6 of USB_CHEPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral must know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB_CHEPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_CHEPnR register bits and DTOG/SW_BUF definition is explained, for the cases of ‘transmission’ and ‘reception’ double-buffered bulk endpoints.

Table 223. Double-buffering buffer flag definition

| Buffer flag | 'Transmission' endpoint | 'Reception' endpoint |
|-------------|---------------------------|----------------------------|
| DTOG | DTOGTX (USB_CHEPnR bit 6) | DTOGRX (USB_CHEPnR bit 14) |
| SW_BUF | USB_CHEPnR bit 14 | USB_CHEPnR bit 6 |

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 224. Bulk double-buffering memory buffers usage (Device mode)

| Endpoint type | DTOG | SW_BUF | Packet buffer used by USB peripheral | Packet buffer used by Application Software |
|---------------|------|--------|---|---|
| Transmit (IN) | 0 | 1 | USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. | USB_CHEP_RXRTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations |
| | 1 | 0 | USB_CHEP_RXRTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations | USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. |
| | 0 | 0 | None ⁽¹⁾ | USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. |
| | 1 | 1 | None ⁽¹⁾ | USB_CHEP_RXRTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. |
| Receive (OUT) | 0 | 1 | USB_CHEP_RXRTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. | USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |
| | 1 | 0 | USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations | USB_CHEP_RXRTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |
| | 0 | 0 | None ⁽¹⁾ | USB_CHEP_RXRTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |
| | 1 | 1 | None ⁽¹⁾ | USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by:

- Writing UTYPE bit field at 00 in its USB_CHEPnR register, to define the endpoint as a bulk, and
- Setting EPKIND bit at 1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains 11 (VALID). However, as the token packet of a new transaction is received, the actual endpoint status is masked as 10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see [Table 224 on page 1209](#)). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing 1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control takes place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from 11 (VALID) into the STAT bit pair of the related USB_CHEPnR register. In this case, the USB peripheral always uses the programmed endpoint status, regardless of the buffer usage condition.

36.5.4 Double buffered channels: usage in Host mode

In Host mode the underlying transmit and receive methods for double buffered channels are the same as those described for Device mode.

Similar to the Device mode table, a new table below [Table 225: Bulk double-buffering memory buffers usage \(Host mode\)](#) shows the programming settings for OUT and IN tokens.

Table 225. Bulk double-buffering memory buffers usage (Host mode)

| Endpoint type | DTOG | SW_BUF | Packet buffer used by USB peripheral | Packet buffer used by Application Software |
|----------------------|-------------|---------------|--|--|
| Transmit (OUT) | 0 | 1 | USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. | USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations |
| | 1 | 0 | USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations | USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. |
| | 0 | 0 | None ⁽¹⁾ | USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. |
| | 1 | 1 | None ⁽¹⁾ | USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. |
| Receive (IN) | 0 | 1 | USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. | USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |
| | 1 | 0 | USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations | USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |
| | 0 | 0 | None ⁽¹⁾ | USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |
| | 1 | 1 | None ⁽¹⁾ | USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |

1. Endpoint in NAK Status.

36.5.5 Isochronous transfers in Device mode

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID), any other value produces results not compliant to USB standard. Isochronous endpoints implement double-buffering

to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOGRX for ‘reception’ isochronous endpoints, DTOGTX for ‘transmission’ isochronous endpoints, both in the related USB_CHEPnR register) according to [Table 226](#).

Table 226. Isochronous memory buffers usage

| Endpoint Type | DTOG bit value | Packet buffer used by the USB peripheral | Packet buffer used by the application software |
|---------------|----------------|--|--|
| Transmit (IN) | 0 | USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. | USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations |
| | 1 | USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations | USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations. |
| Receive (OUT) | 0 | USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. | USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |
| | 1 | USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations | USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations. |

As it happens with double-buffered bulk endpoints, the USB_CHEPnR registers used to implement isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have isochronous endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for isochronous transfers due to the lack of handshake phase, the endpoint remains always 11 (VALID). CRC errors or buffer-overrun conditions occurring during isochronous OUT transfers are anyway considered as correct transactions and they always trigger a VTRX event. However, CRC errors set the ERR bit in the USB_ISTR register anyway, in order to notify the software of the possible data corruption.

36.5.6 Isochronous transfers in Host mode

From the host point of view isochronous packets are issued or requested one by frame by the host frame scheduler. There is no NAK/ACK protocol and no resend of data or token.

The mechanism is based on a table very similar to that for Device mode. See [Table 227](#) below to understand the relationship between the DTOG bit buffers and the buffer usage.

Table 227. Isochronous memory buffers usage

| Endpoint Type | DTOG bit value | Packet buffer used by the USB peripheral | Packet buffer used by the application software |
|----------------|----------------|--|--|
| Transmit (OUT) | 0 | USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations. | USB_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations |
| | 1 | USB_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations | USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations. |
| Receive (IN) | 0 | USB_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations. | USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations. |
| | 1 | USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations | USB_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations. |

The isochronous behavior for an endpoint is selected by setting the UTYPEn bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID),

Just as in Device mode, the mechanism allows automatic toggle of the DTOG bit. Note that in Host mode, at the same time as this toggle, the STATTX or STATRX of the completed buffer is automatically set to DISABLED, permitting the future buffer to be accessed before re-enabling it by setting it to 11 (VALID).

36.5.7 Suspend/resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to 1 in USBISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the SUSPEN bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set SUSPRDY bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to insure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (see “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the SUSPRDY bit in USB_CNTR register asynchronously. Even if this event can trigger a WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure must address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear SUSPEN bit of USB_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to [Table 228](#), which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the idle bus state; moreover at the end of a reset sequence the RST_DCON bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which must be handled as usual.

Table 228. Resume event detection

| [RXDP,RXDM] status | Wake-up event | Required resume software action |
|--------------------|----------------------------|---------------------------------|
| "00" | Root reset | None |
| "10" | None (noise on bus) | Go back in Suspend mode |
| "01" | Root resume | None |
| "11" | Not allowed (noise on bus) | Go back in Suspend mode |

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (for example a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the L2RES bit in the USB_CNTR register to 1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the L2RES bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

Note: *The L2RES bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the SUSPEN bit in USB_CNTR register to 1.*

Suspend and resume in Host mode

The basics of the suspend and resume mechanism has been described in the previous section.

From the host stand-point, suspend is entered by writing the SUSPEN bit in USB_CNTR. When suspend entry is confirmed, SUSPRDY (also in USB_CNTR) is set.

Once in suspend, and when the application want to resume the bus, this can be done by setting the L2RES bit in USB_CNTR to 1.

Below in [Table 229](#), the different actions recommended after a wake-up event are indicated. According to the different line states after a wake-up event, the interpretation of the event and the suggested behavior are shown. Note that, this table here is somewhat expanded when compared to the previously shown device table, as the host may encounter both full speed and low speed devices which use different line states for both suspend and resume.

Table 229. Resume event detection for host

| [RXDP,RXDM] status | Wake-up event | Required resume software action |
|--------------------|--|---------------------------------|
| “00” | Not allowed (noise on bus) | Go back in Suspend mode |
| “10” | Full speed capable device: Not allowed (noise on bus) Low speed device: Device remote wake-up resume | None |
| “01” | Full speed capable device: Device remote wake-up resume Low speed device: Not allowed (noise on bus) | None |
| “11” | Not allowed (noise on bus) | Go back in Suspend mode |

36.6 USB and USB SRAM registers

The USB peripheral registers can be divided into the following groups:

- Common registers: interrupt and control registers
- Endpoint/channel registers: endpoint/channel configuration and status

The USB SRAM registers cover:

- Buffer descriptor table: location of packet memory used to locate data buffers (see [Section 2.2: Memory organization](#) to find USB SRAM base address).

All register addresses are expressed as offsets with respect to the USB peripheral registers base address, except the buffer descriptor table locations, which starts at the USB SRAM base address.

Refer to [Section 1.2 on page 51](#) for a list of abbreviations used in register descriptions.

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

36.6.1 Common registers

These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.

USB control register (USB_CNTR)

Address offset: 0x40

Reset value: 0x0000 0003

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|----------|------|--------|--------|-------------|------|--------|---------|------|--------|--------|---------|----------|---------|----------|
| HOST | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DDISC M | THR 512M |
| rw | | | | | | | | | | | | | | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTRM | PMA OVRM | ERRM | WKUP M | SUSP M | RST_D CONNM | SOFM | ESOF M | L1REQ M | Res. | L1RE S | L2RE S | SUS PEN | SUSP RDY | PDWN | USB RST |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | r | rw | rw |

Bit 31 **HOST**: HOST mode

HOST bit selects between host or device USB mode of operation. It must be set before enabling the USB peripheral by the function enable bit.

0: USB Device function

1: USB host function (Reserved, host function is not available in these products, see [Section 36.3: USB implementation](#))

Bits 30:18 Reserved, must be kept at reset value.

Bit 17 **DDISCM**: Device disconnection mask

– Host mode

0: Device disconnection interrupt disabled

1: Device disconnection interrupt enabled

Bit 16 **THR512M**: 512 byte threshold interrupt mask

0: 512 byte threshold interrupt disabled

1: 512 byte threshold interrupt enabled

- Bit 15 **CTRM:** Correct transfer interrupt mask
 0: Correct transfer (CTR) interrupt disabled.
 1: CTR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 14 **PMAOVRM:** Packet memory area over / underrun interrupt mask
 0: PMAOVR interrupt disabled.
 1: PMAOVR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 13 **ERRM:** Error interrupt mask
 0: ERR interrupt disabled.
 1: ERR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 12 **WKUPM:** Wake-up interrupt mask
 0: WKUP interrupt disabled.
 1: WKUP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 11 **SUSPM:** Suspend mode interrupt mask
 0: Suspend mode request (SUSP) interrupt disabled.
 1: SUSP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 10 **RST_DCONM:** USB reset request (Device mode) or device connect/disconnect (Host mode) interrupt mask
 0: RESET interrupt disabled.
 1: RESET interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 9 **SOFM:** Start of frame interrupt mask
 0: SOF interrupt disabled.
 1: SOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 8 **ESOFM:** Expected start of frame interrupt mask
 0: Expected start of frame (ESOF) interrupt disabled.
 1: ESOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 7 **L1REQM:** LPM L1 state request interrupt mask
 0: LPM L1 state request (L1REQ) interrupt disabled.
 1: L1REQ interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **L1RES:** L1 remote wake-up / resume driver
 – Device mode
 Software sets this bit to send a LPM L1 50 µs remote wake-up signaling to the host. After the signaling ends, this bit is cleared by hardware.
 0: No effect
 1: Send 50 µs remote-wake-up signaling to host

Bit 4 L2RES: L2 remote wake-up / resume driver

– Device mode

The microcontroller can set this bit to send remote wake-up signaling to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the host PC is ready to drive the resume sequence up to its end.

– Host mode

Software sets this bit to send resume signaling to the device.

Software clears this bit to send end of resume to device and restart SOF generation.

In the context of remote wake up, this bit is to be set following the WAKEUP interrupt.

0: No effect

1: Send L2 resume signaling to device

Bit 3 SUSPEN: Suspend state enable

– Condition: Device mode

Software can set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms. Software can also set this bit when the L1REQ interrupt is received with positive acknowledge sent.

As soon as the suspend state is propagated internally all device activity is stopped, USB clock is gated, USB transceiver is set into low power mode and the SUSPRDY bit is set by hardware. In the case that device application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the microcontroller to stop mode, as in the case of bus powered device application, it must first wait few cycles to see the SUSPRDY = 1 acknowledge the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

– Condition: Host mode

Software can set this bit when host application has nothing scheduled for the next frames and wants to enter long term power saving. When set, it stops immediately SOF generation and any other host activity, gates the USB clock and sets the transceiver in low power mode. If any USB transaction is on-going at the time SUSPEN is set, suspend is entered at the end of the current transaction.

As soon as suspend state is propagated internally and gets effective the SUSPRDY bit is set. In the case that host application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the micro-controller to STOP mode, it must first wait few cycles to see SUSPRDY=1 acknowledge to the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

Bit 2 SUSPRDY: Suspend state effective

This bit is set by hardware as soon as the suspend state entered through the SUSPEN control gets internally effective. In this state USB activity is suspended, USB clock is gated, transceiver is set in low power mode by disabling the differential receiver. Only asynchronous wake-up logic and single ended receiver is kept alive to detect remote wake-up or resume events.

Software must poll this bit to confirm it to be set before any STOP mode entry.

This bit is cleared by hardware simultaneously to the WAKEUP flag being set.

0: Normal operation

1: Suspend state

Bit 1 PDWN: Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.

- 0: Exit power down
- 1: Enter power down mode

Bit 0 USBRST: USB Reset

- Condition: Device mode

Software can set this bit to reset the USB core, exactly as it happens when receiving a RESET signaling on the USB. The USB peripheral, in response to a RESET, resets its internal protocol state machine. Reception and transmission are disabled until the RST_DCON bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RST_DCON interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.

- 0: No effect
 - 1: USB core is under reset
- Condition: Host mode
- Software sets this bit to drive USB reset state on the bus and initialize the device. USB reset terminates as soon as this bit is cleared by software.

- 0: No effect
- 1: USB reset driven

USB interrupt status register (USB_ISTR)

Address offset: 0x44

Reset value: 0x0000 0000

This register contains the status of all the interrupt sources permitting application software to determine which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, performs all necessary actions, and finally it clears the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line is kept high again. If several bits are set simultaneously, only a single interrupt is generated.

Endpoint/channel transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint/channel successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint/channel dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_CHEPnR register (the CTR bit is actually a read only bit). For endpoint/channel-related interrupts, the software can use the direction of transaction (DIR) and IDN read-only bits to identify which endpoint/channel made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with 0 (these bits can only be cleared by software). Read-modify-write cycles must be avoided because between the read and the write operations another bit can be set by the hardware and the next write clears it before the device has the time to service the event.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|---------|-----------|-------|-------|----------|-------|-------|-------|------|------|------|----------|------|-------|---------|
| Res. | LS_DCON | DCON_STAT | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | DDISC | THR 512 |
| r | r | | | | | | | | | | | | | rc_w0 | rc_w0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTR | PMA_OVR | ERR | WKUP | SUSP | RST_DCON | SOF | ESOF | L1REQ | Res. | Res. | DIR | IDN[3:0] | | | |
| r | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | r | r | r | r | r |

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LS_DCON**: Low speed device connected

- Host mode:

This bit is set by hardware when an LS device connection is detected. Device connection is signaled after LS J-state is sampled for 22 consecutive cycles of the USB clock (48 MHz) from the unconnected state.

Bit 29 **DCON_STAT**: Device connection status

- Host mode:

This bit contains information about device connection status. It is set by hardware when a LS/FS device is attached to the host while it is reset when the device is disconnected.

0: No device connected

1: FS or LS device connected to the host

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **DDISC**: Device connection

- Host mode

This bit is set when a device connection is detected. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 16 **THR512**: 512 byte threshold interrupt

This bit is set to 1 by the hardware when 512 bytes have been transmitted or received during isochronous transfers. This bit is read/write but only 0 can be written and writing 1 has no effect. Note that no information is available to indicate the associated channel/endpoint, however in practice only one ISO endpoint/channel with such large packets can be supported, so that channel.

Bit 15 **CTR**: Completed transfer in host mode

This bit is set by the hardware to indicate that an endpoint/channel has successfully completed a transaction; using DIR and IDN bits software can determine which endpoint/channel requested the interrupt. This bit is read-only.

Bit 14 PMAOVR: Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host retries the transaction. The PMAOVR interrupt must never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 13 ERR: Error

This flag is set whenever one of the errors listed below has occurred:

NANS: No ANSwer. The timeout for a host response has expired.

CRC: Cyclic redundancy check error. One of the received CRCs, either in the token or in the data, was wrong.

BST: Bit stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

FVIO: Framing format violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (for example loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 12 WKUP: Wake-up

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the SUSPRDY bit in the CTLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (for example wake-up unit) about the start of the resume process. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 11 SUSP: Suspend mode request

– Device mode

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (SUSPEN=1) until the end of resume sequence. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 10 RST_DCON: USB reset request (Device mode) or device connect/disconnect (Host mode)

– Device mode

This bit is set by hardware when an USB reset is released by the host and the bus returns to idle. USB reset state is internally detected after the sampling of 60 consecutive SE0 cycles.

– Host mode

This bit is set by hardware when device connection or device disconnection is detected. Device connection is signaled after J state is sampled for 22 cycles consecutively from unconnected state. Device disconnection is signaled after SE0 state is seen for 22 bit times consecutively from connected state.

Bit 9 SOF: Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine may monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this can be useful for isochronous applications). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 8 ESOF: Expected start of frame

- Device mode

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the suspend timer issues this interrupt. If three consecutive ESOF interrupts are generated (for example three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the suspend timer is not yet locked. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 7 L1REQ: LPM L1 state request

- Device mode

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 DIR: Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.

If DIR bit = 0, VTTX bit is set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).

If DIR bit = 1, VTRX bit or both VTTX/VTRX are set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed. This information can be used by the application software to access the USB_CHEPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **IDN[3:0]:** Device Endpoint / host channel identification number

These bits are written by the hardware according to the host channel or device endpoint number, which generated the interrupt request. If several endpoint/channel transactions are pending, the hardware writes the identification number related to the endpoint/channel having the highest priority defined in the following way: two levels are defined, in order of priority: isochronous and double-buffered bulk channels/endpoints are considered first and then the others are examined. If more than one endpoint/channel from the same set is requesting an interrupt, the IDN bits in USB_ISTR register are assigned according to the lowest requesting register, CHEP0R having the highest priority followed by CHEP1R and so on. The application software can assign a register to each endpoint/channel according to this priority scheme, so as to order the concurring endpoint/channel requests in a suitable way. These bits are read only.

USB frame number register (USB_FNR)

Address offset: 0x48

Reset value: 0x0000 0XXX (where X is undefined)

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

Bit 15 **RXDP:** Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 14 **RXDM:** Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 13 **LCK:** Locked

– Device mode

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]:** Lost SOF

– Device mode

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]:** Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

USB Device address (USB_DADDR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

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

Bit 7 EF: Enable function

This bit is set by the software to enable the USB Device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at 0 no transactions are handled, irrespective of the settings of USB_CHEPnR registers.

Bits 6:0 ADD[6:0]: Device address

- Device mode

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the endpoint/channel address (EA) field in the associated USB_CHEPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

- Host mode

These bits contain the address transmitted with the LPM transaction

LPM control and status register (USB_LPMCSR)

Address offset: 0x54

Reset value: 0x0000 0000

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

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

Bits 7:4 BESL[3:0]: BESL value

- Device mode

These bits contain the BESL value received with last ACKed LPM Token

Bit 3 REMWAKE: bRemoteWake value

- Device mode

This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved, must be kept at reset value.

Bit 1 LPMACK: LPM token acknowledge enable

- Device mode:

0: the valid LPM token is NYET.

1: the valid LPM token is ACK.

The NYET/ACK is returned only on a successful LPM transaction:

No errors in both the EXT token and the LPM token (else ERROR)

A valid bLinkState = 0001B (L1) is received (else STALL)

Bit 0 LPMEN: LPM support enable

- Device mode

This bit is set by the software to enable the LPM support within the USB Device. If this bit is at 0 no LPM transactions are handled.

Battery charging detector (USB_BCDR)

Address offset: 0x58

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------|------|------|------|------|------|------|---------|------|------|------|------|------|------|--------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DPPU-DPD | Res. | Res. | Res. | Res. | Res. | Res. | PS2 DET | SDET | PDET | Res. | SDEN | PDEN | Res. | BCD EN | |
| rw | | | | | | | r | r | r | | rw | rw | | rw | |

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

Bit 15 **DPPU_DPD:** DP pull-up / DPDM pull-down

- Device mode

This bit is set by software to enable the embedded pull-up on DP line. Clearing it to 0 can be used to signal disconnect to the host when needed by the user software.

- Host mode

This bit is set by software to enable the embedded pull-down on DP and DM lines.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET:** DM pull-up detection status

- Device mode

This bit is active only during PD and gives the result of comparison between DM voltage level and V_{LGC} threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, ACA, CDP or DCP).

1: PS2 port or proprietary charger detected.

Bit 6 **SDET:** Secondary detection (SD) status

- Device mode

This bit gives the result of SD.

0: CDP detected.

1: DCP detected.

Bit 5 **PDET:** Primary detection (PD) status

- Device mode

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 Reserved, must be kept at reset value.

Bit 3 **SDEN:** Secondary detection (SD) mode enable

- Device mode

This bit is set by the software to put the BCD into SD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly.

Bit 2 **PDEN**: Primary detection (PD) mode enable

- Device mode

This bit is set by the software to put the BCD into PD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCDEN**: Battery charging detector (BCD) enable

- Device mode

This bit is set by the software to enable the BCD support within the USB Device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD must be placed in OFF mode by clearing this bit to 0 in order to allow the normal USB operation.

Host channel-specific/device endpoint-specific registers

The USB peripheral supports up to 8 bidirectional endpoints or host channels. Each USB Device must support a control endpoint/channel whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint/channel number value. For each endpoint, an **USB_CHEPnR** register is available to store the endpoint/channel specific information.

USB endpoint/channel n register (**USB_CHEPnR**)

Address offset: $0x00 + 0x4 * n$, ($n = 0$ to 7)

Reset value: $0x0000\ 0000$

They are also reset when an USB reset is received from the USB bus or forced through bit **USBRST** in the **CTLR** register, except the **VTRX** and **VTTX** bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint/channel has its **USB_CHEPnR** register where n is the endpoint/channel identifier.

Read-modify-write cycles on these registers must be avoided because between the read and the write operations some bits can be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------------|--------------------------|--------------------|--------------------------|--------------|-------------------|---------------|----------------|-------------|---------------------|--------------------|----|----------------|----|----|----|
| Res. | THREE_ERR_RX[1:0] | | THREE_ERR_TX[1:0] | | ERR_RX | ERR_TX | LS_EP | NAK | DEVADDR[6:0] | | | | | | |
| | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rw | rc_w0 | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VTRX | DTOG_RX | STATRX[1:0] | | SETUP | UTYPE[1:0] | | EP_KIND | VTTX | DTOG_TX | STATTX[1:0] | | EA[3:0] | | | |
| rc_w0 | t | t | t | r | rw | rw | rw | rc_w0 | t | t | t | rw | rw | rw | rw |

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **THREE_ERR_RX[1:0]**: Three errors for an IN transaction

- Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an IN transaction. THREE_ERR_RX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bits 28:27 **THREE_ERR_TX[1:0]**: Three errors for an OUT or SETUP transaction

- Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an OUT transaction. THREE_ERR_TX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bit 26 **ERR_RX**: Received error for an IN transaction

- Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an IN transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 25 **ERR_TX**: Received error for an OUT/SETUP transaction

- Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an OUT or SETUP transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 24 **LS_EP**: Low speed endpoint – host with HUB only

- Host mode

This bit is set by the software to send an LS transaction to the corresponding endpoint.

0: Full speed endpoint

1: Low speed endpoint

Bit 23 **NAK**:

- Host mode

This bit is set by the hardware when a device responds with a NAK. Software can use this bit to monitor the number of NAKs received from a device.

Bits 22:16 **DEVADDR[6:0]**:

- Host mode

Device address assigned to the endpoint during the enumeration process.

Bit 15 **VTRX**: USB valid transaction received

– Device mode

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only 0 can be written, writing 1 has no effect.

– Host mode

This bit is set by the hardware when an IN transaction is successfully completed on this channel. The software can only clear this bit. If the CTRM bit in USB_CNTR register is set a generic interrupt condition is generated together with the channel related flag, which is always activated.

- A transaction ended with a NAK sets this bit and NAK answer is reported to application reading the NAK state from the STATRX field of this register. One NAKed transaction keeps pending and is automatically retried by the host at the next frame, or the host can immediately retry by resetting STATRX state to VALID.

- A transaction ended by STALL handshake sets this bit and the STALL answer is reported to application reading the STALL state from the STATRX field of this register. Host application must consequently disable the channel and re-enumerate.

- A transaction ended with ACK handshake sets this bit

If double buffering is disabled, ACK answer is reported by application reading the DISABLE state from the STATRX field of this register. Host application must read received data from USBRAM and re-arm the channel by writing VALID to the STATRX field of this register.

If double buffering is enabled, ACK answer is reported by application reading VALID state from the STATRX field of this register. Host application must read received data from USBRAM and toggle the DTOGTX bit of this register.

- A transaction ended with error sets this bit.

Errors can be seen via the bits ERR_RX (host mode only).

This bit is read/write but only 0 can be written, writing 1 has no effect.

Bit 14 DTOGRX: Data Toggle, for reception transfers

If the endpoint/channel is not isochronous, this bit contains the expected value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID received from host (in device mode), while it sets this bit to 1 when SETUP transaction is acknowledged by device (in host mode).

If the endpoint/channel is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to [Section 36.5.3: Double-buffered endpoints and usage in Device mode](#)).

If the endpoint/channel is isochronous, this bit is used only to support packet buffer swapping for data transmission since no data toggling is used for this kind of channels/endpoints and only DATA0 packet are transmitted (Refer to [Section 36.5.5: Isochronous transfers in Device mode](#)). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGRX remains unchanged, while writing 1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STATRX[1:0]**: Status bits, for reception transfers

– Device mode

These bits contain information about the endpoint status, which are listed in [Table 230: Reception status encoding on page 1234](#). These bits can be toggled by software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATRX bits to NAK when a correct transfer has occurred ($VTRX = 1$) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledges a new transaction.

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to [Section 36.5.3: Double-buffered endpoints and usage in Device mode](#)).

If the endpoint is defined as isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STATRX bits to ‘STALL’ or ‘NAK’ for an isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

These bits are the host application controls to start, retry, or abort host transactions driven by the channel.

These bits also contain information about the device answer to the last IN channel transaction and report the current status of the channel according to the following STATRX table of states:

- DISABLE

DISABLE value is reported in case of ACK acknowledge is received on a single-buffer channel. When in DISABLE state the channel is unused or not active waiting for application to restart it by writing VALID. Application can reset a VALID channel to DISABLE to abort a transaction. In this case the transaction is immediately removed from the host execution list. If the aborted transaction was already under execution it is regularly terminated on the USB but the relative VTRX interrupt is not generated.

- VALID

A host channel is actively trying to submit USB transaction to device only when in VALID state. VALID state can be set by software or automatically by hardware on a NAKED channel at the start of a new frame. When set to VALID, an host channel enters the host execution queue and waits permission from the host frame scheduler to submit its configured transaction.

VALID value is also reported in case of ACK acknowledge is received on a double-buffered channel. In this case the channel remains active on the alternate buffer while application needs to read the current buffer and toggle DTOGTX. In case software is late in reading and the alternate buffer is not ready, the host channel is automatically suspended transparently to the application. The suspended double buffered channel is re-activated as soon as delay is recovered and DTOGTX is toggled.

- NAK

NAK value is reported in case of NAK acknowledge received. When in NAK state the channel is suspended and does not try to transmit. NAK state is moved to VALID by hardware at the start of the next frame, or software can change it to immediately retry transmission by writing it to VALID, or can disable it and abort the transaction by writing DISABLE

- STALL

STALL value is reported in case of STALL acknowledge received. When in STALL state the channel behaves as disabled. Application must not retry transmission but reset the USB and re-enumerate.

Bit 11 **SETUP:** Setup transaction completed

- Device mode

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (VTRX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while VTRX bit is at 1; its state changes when VTRX is at 0. This bit is read-only.

- Host mode

This bit is set by the software to send a SETUP transaction on a control endpoint. This bit changes its value only for control endpoints. It is cleared by hardware when the SETUP transaction is acknowledged and VTTX interrupt generated.

Bits 10:9 **UTYPE[1:0]:** USB type of transaction

These bits configure the behavior of this endpoint/channel as described in [Table 231: Endpoint/channel type encoding](#). Channel0/Endpoint0 must always be a control endpoint/channel and each USB function must have at least one control endpoint/channel which has address 0, but there may be other control channels/endpoints if required. Only control channels/endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint/channel is defined as NAK, the USB peripheral does not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint/channel is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint/channel is a control one. Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EPKIND configuration bit.

The usage of isochronous channels/endpoints is explained in [Section 36.5.5: Isochronous transfers in Device mode](#)

Bit 8 **EPKIND:** endpoint/channel kind

The meaning of this bit depends on the endpoint/channel type configured by the UTYPE bits. [Table 232](#) summarizes the different meanings.

DBL_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in [Section 36.5.3: Double-buffered endpoints and usage in Device mode](#).

STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered 'STALL' instead of 'ACK'. This bit may be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **VTTX:** Valid USB transaction transmitted

- Device mode

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only 0 can be written.

- Host mode

Same as VTRX behavior but for USB OUT and SETUP transactions.

Bit 6 DTOGTX: Data toggle, for transmission transfers

If the endpoint/channel is non-isochronous, this bit contains the required value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint/channel is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint (in device mode) or when a SETUP transaction is acknowledged by the device (in host mode).

If the endpoint/channel is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to [Section 36.5.3: Double-buffered endpoints and usage in Device mode](#)).

If the endpoint/channel is isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (refer to [Section 36.5.5: Isochronous transfers in Device mode](#)). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint/channel is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGTX remains unchanged, while writing 1 makes the bit value to toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 STATTX[1:0]: Status bits, for transmission transfers

- Device mode

These bits contain the information about the endpoint status, listed in [Table 233](#). These bits can be toggled by the software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATTX bits to NAK, when a correct transfer has occurred ($VTTX = 1$) corresponding to a IN or SETUP (control only) transaction addressed to this channel/endpoint. It then waits for the software to prepare the next set of data to be transmitted.

Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to [Section 36.5.3: Double-buffered endpoints and usage in Device mode](#)).

If the endpoint is defined as isochronous, its status can only be “VALID” or “DISABLED”. Therefore, the hardware cannot change the status of the channel/endpoint/channel after a successful transaction. If the software sets the STATTX bits to ‘STALL’ or ‘NAK’ for an isochronous channel/endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

- Host mode

The STATTX bits contain the information about the channel status. Refer to [Table 233](#) for the full descriptions (“Host mode” descriptions). Whereas in Device mode, these bits contain the status that are given out on the following transaction, in Host mode they capture the status last received from the device. If a NAK is received, STATTX contains the value indicating NAK.

Bits 3:0 EA[3:0]: endpoint/channel address

- Device mode

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

- Host mode

Software must write in this field the 4-bit address used to identify the channel addressed by the host transaction.

Table 230. Reception status encoding

| STATRX[1:0] | Meaning |
|-------------|---|
| 00 | DISABLED: all reception requests addressed to this endpoint/channel are ignored. |
| 01 | STALL: Device mode: the endpoint is stalled and all reception requests result in a STALL handshake. Host mode: this indicates that the device has STALLED the channel. |
| 10 | NAK: Device mode: the endpoint is NAKed and all reception requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the reception request. |
| 11 | VALID: this endpoint/channel is enabled for reception. |

Table 231. Endpoint/channel type encoding

| UTYPE[1:0] | Meaning |
|------------|-----------|
| 00 | BULK |
| 01 | CONTROL |
| 10 | ISO |
| 11 | INTERRUPT |

Table 232. Endpoint/channel kind meaning

| UTYPE[1:0] | | EPKIND meaning |
|------------|-----------|---|
| 00 | BULK | DBL_BUF |
| 01 | CONTROL | STATUS_OUT |
| 10 | ISO | SBUF_ISO: This bit is set by the software to enable the single-buffering feature for isochronous endpoint |
| 11 | INTERRUPT | Not used |

Table 233. Transmission status encoding

| STATTX[1:0] | Meaning |
|-------------|--|
| 00 | DISABLED: all transmission requests addressed to this endpoint/channel are ignored. |
| 01 | STALL: Device mode: the endpoint is stalled and all transmission requests result in a STALL handshake. Host mode: this indicates that the device has STALLED the channel. |

Table 233. Transmission status encoding (continued)

| STATTX[1:0] | Meaning |
|-------------|---|
| 10 | NAK: Device mode: the endpoint is NAKed and all transmission requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the transmission request. |
| 11 | VALID: this endpoint/channel is enabled for transmission. |

36.6.2 USBSRAM registers

Note: *The buffer descriptor table is located inside the packet buffer memory in the separate "USB SRAM" address space.*

Although the buffer descriptor table is located inside the packet buffer memory ("USB SRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USB SRAM base address. The buffer descriptor table entry associated with the USB_CHEPnR registers is described below. The memory must be addressed using Word (32-bit) accesses.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in [Structure and usage of packet buffers on page 1201](#).

Channel/endpoint transmit buffer descriptor n (USB_CHEP_TXRXBD_n)

Address offset: $0x0 + 0x8 * n$, ($n = 0$ to 7)

Reset value: 0XXXXX XXXX

This register description applies when corresponding CHEPnR register does not program the use of double buffering working in receive mode (otherwise refer to following register description)

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]**: Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]**: Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as "00" since packet memory is word wide and all packet buffers must be word aligned.

Channel/endpoint receive buffer descriptor n [alternate] (USB_CHEP_TXRXBD_n)

Address offset: $0x0 + 0x8 * n$, ($n = 0$ to 7)

Reset value: 0XXXXX XXXX

This register description applies when corresponding CHEPnR register programs the use of double buffering and activates receive buffers (otherwise refer to previous register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---------------|----------------|----|----|----|----|----|---------------|----|----|----|----|----|----|----|----|----|
| BLSIZE | NUM_BLOCK[4:0] | | | | | | COUNT_RX[9:0] | | | | | | | | | |
| rw | 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 | |
| ADDR_RX[15:0] | | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **BLSIZE**: Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]**: Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in [Table 234](#).

Bits 25:16 **COUNT_RX[9:0]**: Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]**: Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

Table 234. Definition of allocated buffer memory

| Value of NUM_BLOCK[4:0] | Memory allocated when BLSIZE=0 | Memory allocated when BLSIZE=1 |
|------------------------------------|---|---|
| 0 (00000) | Not allowed | 32 bytes |
| 1 (00001) | 2 bytes | 64 bytes |
| 2 (00010) | 4 bytes | 96 bytes |
| 3 (00011) | 6 bytes | 128 bytes |
| ... | ... | ... |
| 14 (01110) | 28 bytes | 480 bytes |
| 15 (01111) | 30 bytes | |
| 16 (10000) | 32 bytes | |
| ... | ... | ... |
| 29 (11101) | 58 bytes | |
| 30 (11110) | 60 bytes | 992 bytes |
| 31 (11111) | 62 bytes | 1023 bytes |

Channel/endpoint receive buffer descriptor n (USB_CHEP_RXTXBD_n)

Address offset: $0x4 + 0x8 * n$, ($n = 0$ to 7)

Reset value: 0XXXXX XXXXX

This register description applies when corresponding CHEPnR register does not program use of double buffering in the transmit mode (otherwise refer to following register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

| | | | | | | | | | | | | | | | |
|----------------------|-----------------------|-----------|-----------|-----------|-----------|----------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BLSIZE | NUM_BLOCK[4:0] | | | | | COUNT_RX[9:0] | | | | | | | | | |
| 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 |
| ADDR_RX[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in [Table 234](#).

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as "00" since packet memory is word wide and all packet buffers must be word aligned.

**Channel/endpoint transmit buffer descriptor n [alternate]
(USB_CHEP_RXTXBD_n)**

Address offset: $0x4 + 0x8 * n$, ($n = 0$ to 7)

Reset value: 0xFFFF XXXX

This register description applies when corresponding CHEPnR register programs use of double buffering and activates transmit buffers (otherwise refer to previous register description).

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as "00" since packet memory is word wide and all packet buffers must be word aligned.

36.6.3 USB register map

The table below provides the USB register map and reset values.

Table 235. USB register map and reset values

Table 235. USB register map and reset values (continued)

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|-----------|-------------------|-------------|------|---------|------|------|-----------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|--|
| 0x20-0x3F | | HOST | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x40 | USB_CNTR | Res. | 0 | LS_DCON | Res. | 0 | DCON_STAT | Res. | |
| 0x44 | USB_ISTR | Reset value | 0 | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | |
| 0x48 | USB_FNR | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| 0x4C | USB_DADDR | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| 0x54 | USB_LPMCSR | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | |
| 0x58 | USB_BCDR | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | 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](#) for the register boundary addresses.

37 Debug support (DBG)

37.1 Introduction

A limited set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core

The debug features can be controlled via a serial-wire debug access port, using industry standard debugging tools. The debug features are based on Arm® CoreSight™ components:

- SWJ-DP: JTAG/Serial-wire debug port
- AHB-AP: AHB access port
- APB-AP: APB access port
- ROM table
- System control space (SCS)
- Breakpoint unit (BPU)
- Data watchpoint and trace unit (DWT)

The debug features are accessible by the debugger via the access ports.

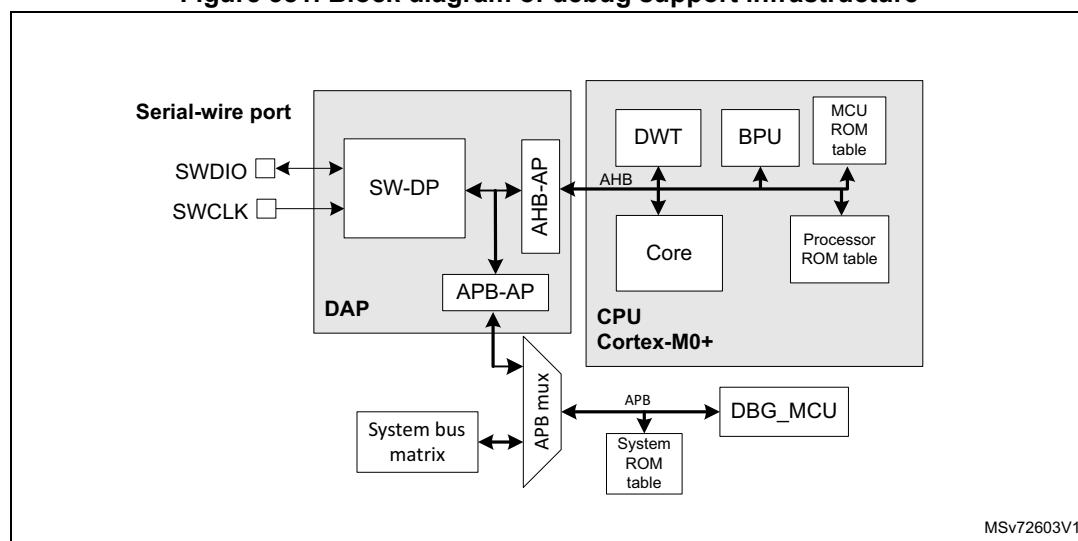
Additional information can be found in the Arm documents referenced in [Section 37.10: Reference documents](#).

Note: Only nonsecure memory access is supported in this product.

37.2 DBG functional description

37.2.1 DBG block diagram

Figure 381. Block diagram of debug support infrastructure



37.2.2 DBG pins and internal signals

SWD port pins

Two pins are used as outputs for the SW-DP as alternate functions of general purpose I/Os. These pins are available on all packages.

Table 236. SW debug port pins

| SW-DP pin name | SW debug port | | Pin assignment |
|----------------|---------------|-------------------------------|----------------|
| | Type | Debug assignment | |
| SWDIO | I/O | Serial wire data input/output | PA13 |
| SWCLK | I | Serial wire clock | PA14 |

SW-DP pin assignment

After reset (SYSRESETn or PORESETn), the pins used for the SW-DP are assigned as dedicated pins, which are immediately usable by the debugger host.

However, the MCU offers the possibility to disable the SWD port, and can then release the associated pins for general-purpose I/O (GPIO) usage. For more details on how to disable SW-DP port pins, refer to [Section 7.3.2: I/O pin alternate function multiplexer and mapping on page 228](#).

Internal pull-up and pull-down on SWD pins

Once the SWD port is released by the user software, the GPIO controller takes control of these pins. The reset state of the GPIO control registers puts the I/Os in the following states:

- SWDIO: input pull-up
- SWCLK: input pull-down

Note: Having embedded pull-up and pull-down resistors removes the need to add external resistors.

37.2.3 ID codes and locking mechanism

There are several ID codes inside the MCU. ST strongly recommends tool manufacturers to lock their debugger using the MCU device ID located in the DBGMCU (see [Section : DBGMCU device ID code register \(DBGMCU_IDCODE\)](#)).

Only the DEV_ID[11:0] field should be used for identification by the debugger/programmer tools (the REV_ID[15:0] field must not be taken into account).

37.2.4 DBG reset and clocks

The debug port (SW-DP) is reset by a power-on reset and when waking up from standby mode.

The debugger supplies the clock for the debug port via the debug interface pin SWCLK. This clock is used to register the serial input data in, as well as to operate the state machines and internal logic of the debug port. This clock 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 SW-DP contains an asynchronous interface to the system clock domain that covers the rest of the SW-DP and the access port.

The debug clock domain is enabled by the debugger using the CDBGPWRUPREQ bit in the [DP control and status register \(DP_CTRLSTAT\)](#). 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 DP_CTRLSTAT. The debug clock is disabled at power-up, and should be disabled before the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor are clocked with the processor clock.

37.2.5 DBG power domains

The debug components are located in the core power domain. This means that the debugger connection is not possible in standby low-power mode. To avoid losing the connection when the device enters standby mode, the power can be maintained to the core by setting a bit in the [DBGMCU configuration register \(DBGMCU_CR\)](#). This also keeps the processor clocks active, and holds off the reset, so that the debug session is maintained.

37.2.6 Debug in 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 if the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power-saving mode emulation is implemented. If the 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.

The emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information, refer to [Section 37.9: Microcontroller debug unit \(DBGMCU\)](#).

37.2.7 Security

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled or limited in scope. Debugger access is disabled while the processor is booting from system flash memory.

The following authentication signal is used by the system to determine which debug features are enabled or disabled:

- **dbgen**: global enable for all debug features
 - 0: All debug features are disabled.
 - 1: Debug features are enabled.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component chapter or to the relevant Arm® technical documentation.

The state of the signals is set according to the debug state as shown in [Table 237](#).

Table 237. Authentication signal states

| Debug state | Authentication signal state | Description |
|-------------|-----------------------------|--|
| OPEN | dbgen = 1 | Debug is enabled. All memory and resources are accessible to the debugger. |
| CLOSED | dbgen = 0 | Debug is disabled. |

The debug state depends on the flash memory readout protection state (see [Section 3.5.1: FLASH read protection \(RDP\)](#)).

Table 238. Life cycle state and debug states

| Product readout protection state (RDP) ⁽¹⁾ | Debug state |
|---|-------------|
| Level 0 | OPEN |
| Level 1 | CLOSED |
| Level 2 | CLOSED |

1. On new products, the readout protection mechanism is named “product life cycle”.

37.3 Serial-wire debug port (SW-DP)

The SW-DP is a CoreSight™ component that implements a two-pin (clock + data) serial-wire debug port for connecting debugging equipment.

A debugger must first select the SW-DP by transmitting the following serial data sequence on SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

SWCLK must be cycled for each data bit.

37.3.1 Serial-wire debug port

The serial-wire debug protocol uses the following pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional serial data

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 239](#)).
2. Acknowledge response (3 bits) transmitted by the target (see [Table 240](#)).
3. Data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read) (see [Table 241](#)).

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 239. Packet request

| Bit field | Name | Description |
|-----------|--------|--|
| 0 | Start | Must be 1 |
| 1 | APnDP | – 0: DP register access (see Section 37.3.2: Debug port registers) – 1: AP register access (see Section 37.4: Access ports) |
| 2 | RnW | – 0: write request – 1: read request |
| 4:3 | A(3:2) | Address field of the DP or AP register (refer to Table 243 or Table 245) |
| 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 240. ACK response

| Bit field | Name | Description |
|-----------|------|--|
| 2:0 | ACK | – 000: FAULT – 010: WAIT – 100: OK |

Table 241. 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.

37.3.2 Debug port registers

The SW-DP accesses the debug port (DP) registers listed in [Table 243](#).

The debugger can access the DP registers as follows:

1. 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 239](#)). Program the A(3:2) field with the register address (two most significant bits, the two LSBs are implicitly zero). Program the RnW bit to

- select a read or write. In the case of a write, program the data field with the write data. The write data are sent in the data phase.
2. To access one of the banked DP registers at address 0x4, the register number must first be written to the DPBANKSEL[3:0] field of the DP_SELECT register at address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the contents of DPBANKSEL.

Table 242. Debug port registers

| Address | A(3:2) value | R/W | Description |
|---------|--------------|-----|--|
| 0x0 | 00 | R | <i>DP debug port identification register (DP_DPIDR)</i> contains the IDCODE for the debug port. |
| | | W | <i>DP abort register (DP_ABORT)</i> aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRLSTATR register. |
| 0x4 | 01 | R/W | If DP_SELECTR.DPBANKSEL[3:0] = 0x0, <i>DP control and status register (DP_CTRLSTAT)</i> controls the DP and provides status information. |
| | | | If DP_SELECTR.DPBANKSEL[3:0] = 0x1, <i>DP data link control register (DP_DLCSR)</i> controls the operating mode of the SWD data link. |
| | | | If DP_SELECTR.DPBANKSEL[3:0] = 0x2, <i>DP target identification register (DP_TARGETID)</i> provides target identification information. |
| | | | If DP_SELECTR.DPBANKSEL[3:0] = 0x3, <i>DP data link protocol identification register (DP_DLPIDR)</i> provides the SWD protocol version. |
| 0x8 | 10 | R | <i>DP event status register (DP_RESEND)</i> returns the value that was returned by the last AP read or DP_RDBUFF read. Used in the event of a corrupted read transfer. |
| | | W | <i>DP access port select register (DP_SELECT)</i> selects the access port, access port register bank, and DP register at address 0x4. |
| 0xC | 11 | R | <i>DP read buffer register (DP_RDBUFF)</i> contains the result of the preceding AP read access, allowing a new AP access to be avoided. |

37.3.3 DEBUG port registers

DP debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x6BA0 2477

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---------------|----|----|----|----------------|----|----|----|----|----|----|----|----|----|------|------|
| REVISION[3:0] | | | | PARTNO[7:0] | | | | | | | | | | Res. | Res. |
| 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 |
| VERSION[3:0] | | | | DESIGNER[10:0] | | | | | | | | | | Res. | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:28 **REVISION[3:0]**: revision code

0x6: Rev 6

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

Bits 15:12 **VERSION[3:0]**: debug port 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.

DP abort register (DP_ABORT)

Address offset: 0x0

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | ORUNERRCLR | WDERRCLR | STKERRCLR | Res. | DAPABORT |
| | | | | | | | | | | | w | w | w | | w |

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

Bit 4 **ORUNERRCLR**: overrun error clear

0: no effect

1: STICKYORUN bit cleared in DP_CTRLSTAT register

Bit 3 **WDERRCLR**: write data error clear

0: no effect

1: WDATAERR bit cleared in DP_CTRLSTAT register

Bit 2 **STKERRCLR**: sticky error clear

0: no effect

1: STICKYERR bit cleared in DP_CTRLSTAT register

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DAPABORT**: current AP transaction aborted if excessive number of WAIT responses returned

This bit indicates that the transaction is stalled.

0: no effect

1: transaction aborted

DP control and status register (DP_CTRLSTAT)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECT.DPBANKSEL[3:0] = 0x0.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|-------------|-------------|------|------|------|------|----------|--------|-----------|------|------|------|------------|------------|
| Res. | Res. | CDBGWRUPACK | CDBGWRUPREQ | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | 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. | WDATAERR | READOK | STICKYERR | Res. | Res. | Res. | STICKYORUN | ORUNDETECT |
| | | | | | | | | r | r | r | | | | r | r |

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

Bit 29 **CDBGWRUPACK**: debug power-up acknowledgeSee description in [Section 37.2.6: Debug in low-power modes](#).

0: debug clock gated

1: debug clock enabled

Bit 28 **CDBGWRUPREQ**: debug power-up request

This bit controls the debug clock enable.

0: requests debug clock gating

1: requests debug clock enable

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **WDATAERR**: write data error (read-only)

This bit indicates that there is a parity or framing error on the data phase of a write, or a write accepted by the DP is then discarded without being submitted to the AP.

This bit is reset by writing 1 to the DP_ABORT.WDERRCLR bit.

0: no error

1: an error occurred

Bit 6 **READOK**: AP read response (read-only)

This bit indicates the response to the last AP read access.

0: read not OK

1: read OK

Bit 5 **STICKYERR**: transaction error

This bit indicates that an error occurred in an AP transaction. It is reset by writing 1 to the DP_ABORT.STKERRCLR bit

0: no error

1: an error occurred

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STICKYORUN**: overrun (read-only).

This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set. It is reset by writing 1 to the DP_ABORT.ORUNERRCLR bit.

0: no overrun

1: an overrun occurred

Bit 0 **ORUNDETECT**: overrun detection mode enable.

0: disabled

1: enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

DP data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECT.DPBANKSEL[3:0] = 0x1.

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

Bits 7:6 **WIREMODE[1:0]**: SW-DP mode

0x0: synchronous mode

Bits 5:0 Reserved, must be kept at reset value.

DP target identification register (DP_TARGETID)

Address offset: 0x4

Reset value: 0xXXXX 0041

This register is accessible when DP_SELECT.DPBANKSEL[3:0] = 0x2.

| 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] | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | Res. |

Bits 31:28 **TREVISION[3:0]**: target revision

Bits 27:12 **TPARTNO[15:0]**: target part number

0x4890: STM32U073/083x

0x4590: STM32U031x

Bits 11:1 **TDESIGNER[10:0]**: target designer JEDEC code

0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x3.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----------------|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|--------------|
| TINSTANCE[3:0] | | | | 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 | Res | Res | Res | PROTSVN[3:0] |
| | | | | | | | | | | | | r | r | r | r |

Bits 31:28 **TINSTANCE[3:0]**: target instance number

This field defines the instance number for the 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

DP event status register (DP_RESEND)

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[31:0]**: value returned by the last AP read or DP_RDBUFF read

This register is used in the event of a corrupted read transfer.

DP access port select register (DP_SELECT)

Address offset: 0x8

Reset value: 0xFFFF XXXX

| 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. |
| 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

This field selects the access port for the next transaction.

0x00: AP0 - System debug access port (APB-AP)

0x01: AP1 - Cortex®-M0+ debug access port (AHB-AP)

others: reserved, must not be used

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select

This field selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select

This field selects the register at address 0x4 of the debug port.

0x0: DP_CTRLSTAT register

0x1: DP_DLCSR register

0x2: DP_TARGETID register

0x3: DP_DLPIDR register

others: reserved, must not be used

DP read buffer register (DP_RDBUFF)

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]**: value 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 that 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.

37.3.4 Debug port register map and reset values

These registers are not on the CPU memory bus. They are only accessed through the SW-DP debug interface.

The debug port address offset is four-bit wide, where the two most significant bits are defined by the SW-DP packet request A[3:2] field. The two least significant bits are 00.

Table 243. Debug port register map and reset values

1. DP_SELECT.DPBANKSEL[3:0] = 0x0.

2. DP_SELECT.DPBANKSEL[3:0] = 0x1.
3. DP_SELECT.DPBANKSEL[3:0] = 0x2.
4. DP_SELECT.DPBANKSEL[3:0] = 0x3.

37.4 Access ports

There are two access ports (AP) attached to the DP:

- System debug access port (AP1): it enables the access to the DBGMCU and the system ROM table via an APB.
- Cortex®-M0+ debug access port (AP0): it enables the access to the debug and trace features integrated in the Cortex-M0+ processor core via a dedicated AHB.

37.4.1 Access port registers

The access ports are of type MEM-AP: the debug and trace component registers are mapped in the address space of the AHB or APB. 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 245](#).

The address of the AP registers is composed of the following fields:

- bits [7:4]: content of the APBANKSEL[3:0] field in the [DP access port select register \(DP_SELECT\)](#)
- bits [3:2]: content of the A(3:2) field of the SW-DP packet request (see [Table 239](#)), depending on the debug interface used
- bits [1:0]: always cleared

The content of the APSEL[3:0] field of the DP_SELECT register defines which MEM-AP is being accessed. Some register fields differ between the APB and AHB access ports.

Table 244. MEM-AP registers

| Address | APBANKSEL | A(3:2) | Name | Description |
|---------|-----------|--------|------|--|
| 0x00 | 0x0 | 0 | CSW | Control/status word register |
| 0x04 | 0x0 | 1 | TAR | Transfer address register Target address for the bus transaction. |
| 0x08 | - | - | - | Reserved |
| 0x0C | 0x0 | 3 | DRW | Data read/write register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:0] |
| 0x10 | 0x1 | 0 | BD0 | Banked data 0 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x0. |
| 0x14 | 0x1 | 1 | BD1 | Banked data 1 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x4. |

Table 244. MEM-AP registers (continued)

| Address | APBANKSEL | A(3:2) | Name | Description |
|--------------|-----------|--------|------|--|
| 0x18 | 0x1 | 2 | BD2 | Banked data 2 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x8. |
| 0x1C | 0x1 | 3 | BD3 | Banked data 3 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0xC. |
| 0x20 | - | - | - | Reserved |
| 0x24 to 0xEC | - | - | - | Reserved |
| 0xF0 | - | - | - | Reserved |
| 0xF4 | 0xF | 1 | CFG | Configuration register (read only) |
| 0xF8 | 0xF | 2 | BASE | Debug base address register (read only) Base address of the ROM table |
| 0xFC | 0xF | 3 | IDR | Identification register (read only) |

The debugger can access the AP registers as follows:

1. Program the APSEL[3:0] field in the [DP access port select register \(DP_SELECT\)](#) to choose the AP, and the APBANKSEL[3:0] field in DP_SELECT to select the register bank to be accessed.
2. 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 239](#)). Program the A(3:2) field 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. The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the [APx transfer address register \(APx_TAR\)](#) ($x = 0, 1$).
2. Program the [AP0 control/status word register \(AP0_CS0\)](#), if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the [APx data read/write register \(APx_DRW\)](#) ($x = 0, 1$) to initiate a bus transaction at the address held in AP_TAR. Alternatively, a read or write to the [APx banked data n register \(APx_BDn\)](#) ($x = 0, 1$) triggers access to the TAR[31:4] + n address, allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register.

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification [\[1\]](#).

AP1 control/status word register (AP1_CSW)

Address offset: 0x0

Reset value: 0x8000 0042

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---------|------|------|------|-----------|------|------|------|--------------|--------------|--------------|------|-----------|------|------|------|
| DBGSWEN | 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. | MODE[3:0] | | | | TRINP ROG | DEVIC EEN | ADDRINC[1:0] | Res. | SIZE[2:0] | | | |
| | | | | rw | rw | rw | rw | r | r | rw | rw | | r | r | r |

Bit 31 **DBGSWEN**: software access enable

Enables or disables software access to the APB bus.

0: disable software access

1: enable software access

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: mode of operation

0b0000: normal download or upload

other: reserved, must not be used

Bit 7 **TRINPROG**: transfer in progress (read only)

This field indicates whether a transfer is currently in progress on the APB master port.

0: no APB transfer in progress

1: APB transfer in progress

Bit 6 **DEVICEEN**: device enable status (read only)

1: APB transfers always enabled

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode

This field defines whether the TAR address is automatically incremented after a transaction.

0x0: no auto-increment

0x1: address incremented by the size in bytes of the transaction (SIZE field)

other: reserved, must not be used

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction

0x2: word (32-bit)

AP0 control/status word register (AP0_CSW)

Address offset: 0x0

Reset value: 0x43X0 00X2

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|-------|------|------|-----------|------|------|------|-----------|-----------|--------------|------|------|-----------|------|------|
| Res. | SPROT | Res. | Res. | PROT[3:0] | | | | SPISTATUS | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | rw | | | rw | rw | rw | rw | 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. | TRINPROG | DBGSTATUS | ADDRINC[1:0] | | Res. | SIZE[2:0] | | |
| | | | | | | | | r | r | rw | rw | | rw | rw | rw |

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: secure transfer request

This field sets the protection attribute HPROT[6] of the bus transfer.
0: reserved, must not be used

1: nonsecure transfer, HPROT[6] = high

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

Bits 27:24 **PROT[3:0]**: bus transfer protection

This field sets the protection attributes HPROT[3:0] of the bus transfer.

- bit 0 = 0: instruction access
- bit 0 = 1: data access
- bit 1 = 0: user mode
- bit 1 = 1: privilege mode
- bit 2 = 0: non-bufferable
- bit 2 = 1: bufferable
- bit 3 = 0: non-shareable, no look-up, non-modifiable
- bit 3 = 1: shareable, look-up, modifiable

Bit 23 **SPISTATUS**: secure debug authentication status

This field indicates the state of the SPIDEN signal

0: No secure AHB transfers allowed

Bits 22:8 Reserved, must be kept at reset value.

Bit 7 **TRINPROG**: transfer in progress (read only)

This field indicates whether a transfer is currently in progress on the APB master port.

- 0: No AHB transfer in progress
- 1: AHB transfer in progress

Bit 6 **DBGSTATUS**: debug enable (DBGEN) status

- 0: AHB transfers blocked
- 1: AHB transfers enabled

Bits 5:4 ADDRINC[1:0]: auto-increment mode

This field defines whether the TAR address is automatically incremented after a transaction.

0x0: no auto-increment

0x1: address incremented by the size in bytes of the transaction (SIZE field). Single transfer.

0x2: address incremented by the size in bytes of the transaction (SIZE field). Packs four 8-bit transfers or two 16-bit transfers into a 32-bit DAP transfer. Multiple transactions are carried out on the AHB interface.

other: reserved, must not be used

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)

others: reserved, must not be used

APx transfer address register (APx_TAR) (x = 0, 1)

Address offset: 0x04

Reset value: 0xXXXX XXXX

| 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**. In AP1, TA[1:0] are fixed at 0.

APx data read/write register (APx_DRW) (x = 0, 1)

Address offset: 0x0C

Reset value: 0xXXXX XXXX

| 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**

APx banked data n register (APx_BDn) (x = 0, 1)

Address offset: 0x10 + 0x4 * n, (n = 0 to 3)

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: banked data of current transfer to address TA [31:4] + 4 * n.

Auto address incrementing is not performed on APx_BD0-3. Banked transfers are only supported for word transfers.

APx base address register (APx_BASE) (x = 0, 1)

Address offset: 0xF8

Reset value: AP 0: 0xF000 0003

Reset value: AP 1: 0xF000 0003

| | | | | | | | | | | | | | | | |
|-----------------|----|----|----|------|------|------|------|------|------|------|------|------|------|--------|--------------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BASEADDR[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 |
| BASEADDR[15:12] | | | | | | | | | | | | | | | |
| | | | | Res. | FORMAT | ENTRYPRESENT |
| r | r | r | r | | | | | | | | | | | r | r |

Bits 31:12 **BASEADDR[31:12]**: base address (bits 31 to 12) of the first ROM table

The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.

0xF0000: AP0

0xF0000: AP1

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**: debug components presence

This bit indicates that debug components are present on the access port bus.

1: debug components present

APx identification register (APx_IDR) (x = 0, 1)

Address offset: 0xFC

Reset value: AP 1: 0x5477 0002

Reset value: AP 0: 0x8477 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | CLASS[3] | |
|---------------|----|----|------|-----------------|------|------|------|-----------------|----|----|----|----|----|----|----|----------|--|
| REVISION[3:0] | | | | JEDEC BANK[3:0] | | | | JEDEC CODE[6:0] | | | | | | | | | |
| r | 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 | | |
| CLASS[2:0] | | | Res. | Res. | Res. | Res. | Res. | IDENTITY[7:0] | | | | | | | | | |
| r | r | r | | | | | | r | r | r | r | r | r | r | r | r | |

Bits 31:28 **REVISION[3:0]**: revision number

0x5: r1p0

0x8: r0p9

Bits 27:24 **JEDEC BANK[3:0]**: JEDEC bank

0x4: Arm®

Bits 23:17 **JEDEC CODE[6:0]**: JEDEC code

0x3B: Arm®

Bits 16:13 **CLASS[3:0]**:

0x1: MEM-AP

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**:

0x1: AHB-AP

0x2: APB-AP

37.4.2 Access port register map

These registers are not on the CPU memory bus. They are only accessed through SW-DP debug interface.

The access port address is 8-bit wide, defined by DP_SELECT.APBANKSEL[3:0] field and by the SW-DP packet request A[3:2] field. The two least significant bits are 00.

Table 245. 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 | AP1_CSW | DBGSWEN | Res. | | |
| | | Reset value | 1 | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Table 245. Access port register map and reset values (continued)

37.5 ROM tables

The ROM table is a CoreSight™ component that contains the base addresses of the CoreSight debug components accessible via the access port to which it is attached. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There is one top level ROM table behind each access port, APn. The base address of this ROM table can be obtained by reading the APn_BASE register of the access port. The top level ROM table may point in turn to other ROM tables.

The system ROM table is pointed to by the AP1 base register, AP1_BASE. It contains the base address pointer for the DBGMCU.

The system ROM table occupies a 4-Kbyte, 32-bit wide chunk of the AP1 address space, from 0xF000 0000 to 0xF000 0FFC. It can be accessed by the debugger.

Table 246. System ROM table

| Address offset in ROM table | Component name | Component base address | Component address offset | Size (Kbytes) | Entry |
|-----------------------------|---------------------|------------------------|--------------------------|---------------|-------------------------------|
| 0x000 | DBGMCU | 0xF000 1000 (debugger) | 0x0000 1000 | 4 | 0x0000 1003 |
| 0x004 | Top of table | - | - | - | 0x0000 0000 |
| 0x008 to 0xFC8 | Reserved | - | - | - | 0x0000 0000 |
| 0xFFC to 0xFFC | ROM table registers | - | - | - | See Table 249 |

There are two ROM tables in the CPU subsystem. The MCU ROM table is pointed to by the AP0 base register, AP0_BASE. It contains the base-address pointer for the processor ROM table.

The MCU ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of the AP0 address space, from 0xF000 0000 to 0xF000 0FFC. It is not accessible by software.

Table 247. MCU ROM table

| Address offset in ROM table | Component name | Component base address | Component address offset | Size (Kbytes) | Entry |
|-----------------------------|---------------------|------------------------|--------------------------|---------------|-------------------------------|
| 0x000 | Processor ROM table | 0xE00F F000 | 0x0000 1000 | 4 | 0xF00F F003 |
| 0x004 | Reserved | - | - | - | 0x0020 0002 |
| 0x008 | Reserved | - | - | - | 0x1000 0002 |
| 0x00C | Reserved | - | - | - | 0x1000 0002 |
| 0x010 | Top of table | - | - | - | 0x0000 0000 |
| 0x014 to 0xFC8 | Reserved | - | - | - | 0x0000 0000 |
| 0xFFC to 0xFFC | ROM table registers | - | - | - | See Table 250 |

The processor ROM table contains the base-address pointer for the system control space (SCS) registers that allow the debugger to identify the CPU core, as well as for the BPU, DWT.

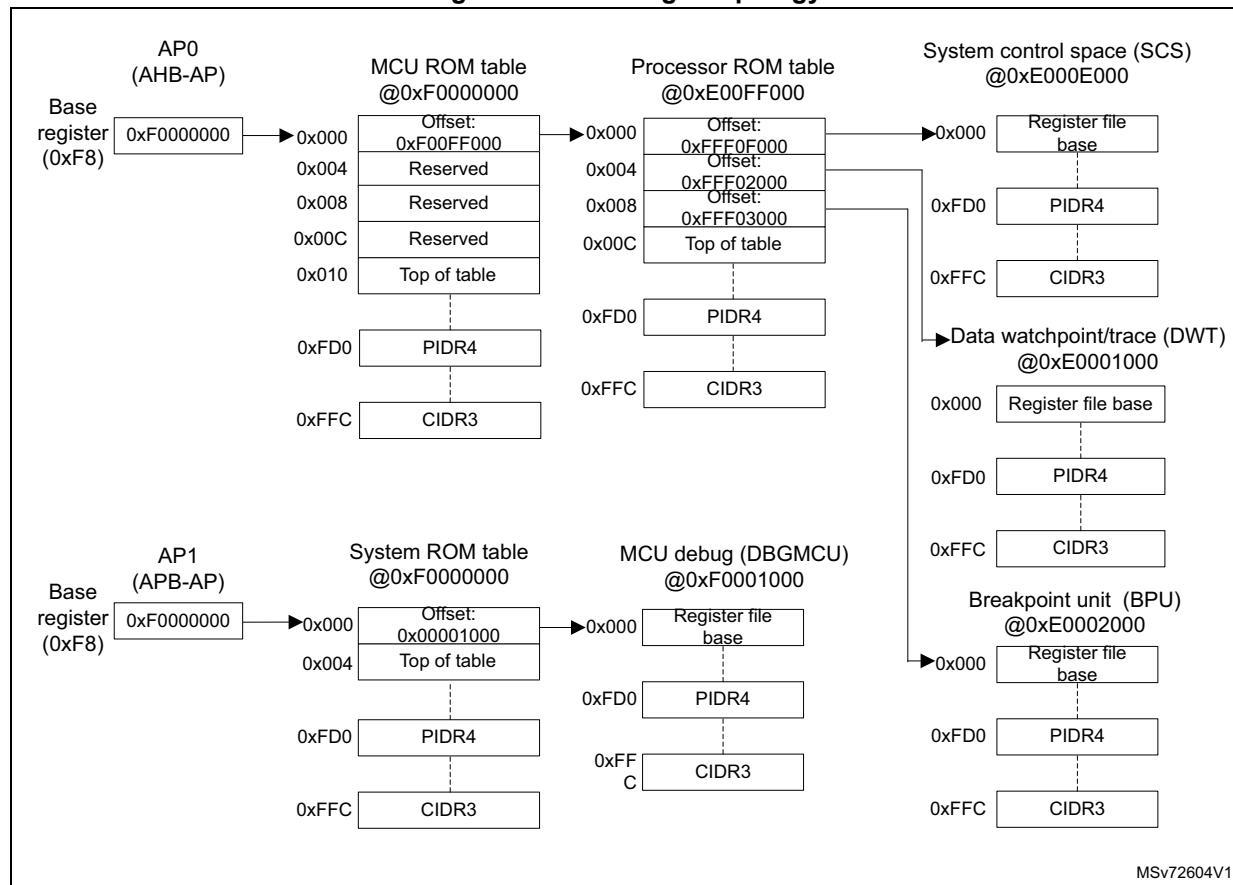
The processor ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of AP0 address space, from 0xE00F F000 to 0xE00F FFFC. It is not accessible by software.

Table 248. Processor ROM table

| Address in ROM table | Component name | Component base address | Component address offset | Size (Kbytes) | Entry |
|----------------------------|---------------------|------------------------|--------------------------|---------------|-------------------------------|
| 0xE00F F000 | SCS | 0xE000 E000 | 0xFFFF0 F000 | 4 | 0xFFFF0 F003 |
| 0xE00F F004 | DWT | 0xE000 1000 | 0xFFFF0 2000 | 4 | 0xFFFF0 2003 |
| 0xE00F F008 | BPU | 0xE000 2000 | 0xFFFF0 3000 | 4 | 0xFFFF0 3003 |
| 0xE00F F00C | Top of table | - | - | - | 0x0000 0000 |
| 0xE00F F010 to 0xE00F FFC8 | Reserved | - | - | - | 0x0000 0000 |
| 0xE00F FFCC to 0xE00F FFFC | ROM table registers | - | - | - | See Table 251 |

The topology for the CoreSight components is shown in [Figure 382](#).

Figure 382. CoreSight topology



37.5.1 System ROM table registers

System ROM memory type register (SYSROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SYSMEM |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **SYSMEM**: system memory

0x1: system memory present on this bus

System ROM CoreSight peripheral identity register 4 (SYSROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SIZE[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 **SIZE[3:0]**: register file size

0x0: The register file occupies a single 4-Kbyte region.

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

0x0: STMicroelectronics JEDEC continuation code

System ROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

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

0x00: STM32U0 series

System ROM CoreSight peripheral identity register 1(SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0000

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

Bits 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]

0x0: STM32U0 series

System ROM CoreSight peripheral identity register 2 (SYSROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | | | | | | | REVISION[3:0] |
| | | | | | | | | | r | r | r | r | r | r | JEDEC |
| | | | | | | | | | r | r | r | r | r | r | JEP106ID[6:4] |

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: designer identification specified by JEDEC
Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

System ROM CoreSight peripheral identity register 3 (SYSROM_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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | r | r | r | r | r | r | r | r |
| REVAND[3:0] | | | | | | | | | | | | CMOD[3:0] | | | |
| | | | | | | | | | | | | r | r | r | r |

Bits 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

System ROM CoreSight component identity register 0 (SYSROM_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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | r | r | r | r | r | r | r | r |
| PREAMBLE[7:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | r | r | r | r |

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

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: Common identification value

System ROM CoreSight peripheral identity register 1 (SYSROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class

0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]

0x0: Common identification value

System ROM CoreSight component identity register 2 (SYSROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. |
| | | | | | | | | 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 identification bits [23:16]

0x05: common identification value

System ROM CoreSight component identity register 3 (SYSROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. |
| | | | | | | | | 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 identification bits [31:24]

0xB1: Common identification value

37.5.2 System ROM table register map

Table 249. System ROM table register map and reset values

Refer to [Table 246: System ROM table](#) for register boundary addresses.

37.5.3 MCU ROM table registers

MCU ROM memory type register (MCUROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SYSMEM |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **SYSMEM**: system memory

0x1: system memory present on this bus

MCU ROM CoreSight peripheral identity register 4 (MCUROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SIZE[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 **SIZE[3:0]**: register file size

0x0: The register file occupies a single 4-Kbyte region.

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

0x0: STMicroelectronics JEDEC continuation code

MCU ROM CoreSight peripheral identity register 0 (MCUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0089

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

0x89: STM32U0 series

MCU ROM CoreSight peripheral identity register 1(MCUROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | | | | | | | | PARTNUM[11:8] |
| | | | | | | | | r | r | r | r | r | r | r | r |

Bits 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: STM32U0 series

MCU ROM CoreSight peripheral identity register 2 (MCUROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | | | | | | | | REVISION[3:0] JEDEC JEP106ID[6:4] |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: designer identification specified by JEDEC
Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

MCU ROM CoreSight peripheral identity register 3 (MCUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

Bits 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

MCU ROM CoreSight component identity register 0 (MCUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 identification bits [7:0]
0xD: Common identification value

MCU ROM CoreSight peripheral identity register 1 (MCUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class

0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]

0x0: Common identification value

MCU ROM CoreSight component identity register 2 (MCUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. |
| | | | | | | | | 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 identification bits [23:16]

0x05: common identification value

MCU ROM CoreSight component identity register 3 (MCUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. |
| | | | | | | | | 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 identification bits [31:24]

0xB1: Common identification value

37.5.4 MCU ROM table register map

Table 250. MCU ROM table register map and reset values

Refer to [Table 247: MCU ROM table](#) for register boundary addresses.

37.5.5 Processor ROM table registers

CPU ROM memory type register (CPUROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SYSMEM |
| | | | | | | | | | | | | | | | r |

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

Bit 0 **SYSMEM**: system memory

1: system memory present on this bus

CPU ROM CoreSight peripheral identity register 4 (CPUROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | SIZE[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 **SIZE[3:0]**: register file size

0x0: The register file occupies a single 4-Kbyte region.

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

0x4: ARM JEDEC continuation code

CPU ROM CoreSight peripheral identity register 0 (CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C0

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

CPU ROM CoreSight peripheral identity register 1 (CPUROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | | | | | | | | PARTNUM[11:8] |
| | | | | | | | | r | r | r | r | r | r | r | r |

Bits 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+

CPU ROM CoreSight peripheral identity register 2 (CPUROM_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | | | | | | | | REVISION[3:0] |
| | | | | | | | | r | r | r | r | r | r | r | JEDEC |
| | | | | | | | | | | | | | | | JEP106ID[6:4] |

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC
Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CPU ROM CoreSight peripheral identity register 3 (CPUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | REVAND[3:0] | | | | CMOD[3:0] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

Bits 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

CPU ROM CoreSight component identity register 0 (CPUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 identification bits [7:0]
0xD: Common identification value

CPU ROM CoreSight peripheral identity register 1 (CPUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | | CLASS[3:0] | | PREAMBLE[11:8] | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class

0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]

0x0: Common identification value

CPU ROM CoreSight component identity register 2 (CPUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 identification bits [23:16]

0x05: common identification value

CPU ROM CoreSight component identity register 3 (CPUROM_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 identification bits [31:24]
0xB1: common identification value

37.5.6 Processor ROM table register map

Table 251. CPU 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 | CPUROM_MEMTYPEP | Res. | 1 | SYSMEM | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFD4 to FDC | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFD0 | CPUROM_PIDR4 | Res. | 0 | JEP106CON [3:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFE0 | CPUROM_PIDR0 | Res. | 1 | PARTNUM[7:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFE4 | CPUROM_PIDR1 | Res. | 0 | JEP106ID [3:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFE8 | CPUROM_PIDR2 | Res. | 1 | PARTNUM [11:8] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFEC | CPUROM_PIDR3 | Res. | 0 | JEDEC [6:4] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFF0 | CPUROM_CIDR0 | Res. | 0 | PREAMBLE[7:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFF4 | CPUROM_CIDR1 | Res. | 0 | CLASS[3:0] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFF8 | CPUROM_CIDR2 | Res. | 0 | PREAMBLE[19:12] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFFC | CPUROM_CIDR3 | Res. | 1 | PREAMBLE[27:20] | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Refer to [Table 248: Processor ROM table](#) for register boundary addresses.

37.6 Data watchpoint and trace unit (DWT)

The DWT provides two comparators that can be used as one of the following, according to the *DWT function register x (DWT_FUNCTIONx)*:

- data address matching and creation of a watchpoint event
- instruction address matching and creation of a PC watchpoint event.

A DWT comparator compares the value held in its *DWT comparator x register (DWT_COMPx)* with one of the following:

- a data address
- an instruction address

For address matching, the comparator can use a mask, so it matches a range of addresses, up to 2GB. The bits to be masked are programmed in *DWT comparator x register (DWT_MASKx)*.

On a successful match, the comparator generates a watchpoint debug event. A watchpoint debug event causes the processor to halt execution and enter debug state.

The DWT also provides a PC sampling register, *DWT program counter sample register (DWT_PCSR)* which allows the debugger to sample the program counter for code profiling.

The DWT features can only be used if the DWT enable (DWTENA) bit is set in the SCS_DEMCR register.

For more details on how to use the DWT, refer to the Arm v6-M Architecture Reference Manual [4].

37.6.1 DWT registers

The DWT registers are accessible to the debugger via access port AP0. They are not accessible by software running on the CPU.

DWT control register (DWT_CTRL)

Address offset: 0x000

Reset value: 0x2000 0000

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

Bits 31:28 **NUMCOMP[3:0]**: number of comparators implemented
0x4: four comparators

Bits 27:0 Reserved, must be kept at reset value.

DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

| 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.

DWT comparator x register (DWT_COMPx)

Address offset: 0x020 + 0x010 * x, (x = 0 to 1)

Reset value: 0xXXXX XXXX

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

Bits 31:0 **COMP[31:0]**: reference value for comparison**DWT comparator x register (DWT_MASKx)**

Address offset: 0x024 + 0x010 * x, (x = 0 to 1)

Reset value: 0xXXXX 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. | Res. | Res. | Res. | 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]**: size of the ignore mask applied to address range matching

Writing all ones to this field and reading it back can be used to determine the maximum mask size supported.

DWT function register x (DWT_FUNCTIONx)

Address offset: 0x028 + 0x010 * x, (x = 0 to 1)

Reset value: 0x5800 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|---------|------|------|------|------|------|------|------|---------------|
| Res. | MATCHED | 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. | FUNCTION[3:0] |
| | | | | | | | | | | | | | rW | rW | rW |

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

Bit 24 **MATCHED**: comparator match

This bit indicates if a comparator match has occurred since the register was last read.

0: no match

1: a match occurred

Bits 23:4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]**: Action on comparator match

0x0: Comparator disabled

0x4: PC watchpoint event on instruction address match

0x5: Watchpoint event on data read address match

0x6: Watchpoint event on data write address match

0x7: Watchpoint event on data read or write address match

others: reserved, must not be used

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res | SIZE[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 **SIZE[3:0]**: register file size

0x0: The register file occupies a single 4-Kbyte region.

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

0x4: Arm® JEDEC code

DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 000A

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

0xA: Cortex®-M0+ DWT part number

DWT CoreSight peripheral identity register 1 (DWT_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | PARTNUM[11:8] |
| | | | | | | | | r | r | r | r | r | r | r | r |

Bits 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: Cortex®-M0+ DWT part number

DWT CoreSight peripheral identity register 2 (DWT_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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | REVISION[3:0] JEDEC JEP106ID[6:4] |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC
Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

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

Bits 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

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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 identification bits [7:0]
0xD: Common identification value

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

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

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class

0xE: Generic IP component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]

0x0: common identification value

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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 identification bits [23:16]

0x05: common identification value

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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 identification bits [31:24]
0xB1: common identification value

37.6.2 DWT register map

Refer to *Table 248: Processor ROM table* for DWT register bank base address.

Table 252. 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_CTRL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x004 to 0x018 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x01C | DWT_PCSR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x020 | DWT_COMP0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x024 | DWT_MASK0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x028 | DWT_FUNCTION0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 0 | MATCHED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x02C | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x030 | DWT_COMP1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | 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 | | | | | | |
| 0x034 | DWT_MASK1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x038 | DWT_FUNCTION1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 0 | MATCHED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x03C to 0xFCC | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFD0 | DWT_PIDR4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xFD4 to 0xFD8 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 252. DWT 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 |
|--------|---------------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|---|---|
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| 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. | Res. | Res. | Res. | Res. | Res. | Res. | | | |
| | Reset value | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | | | |

37.7 Breakpoint unit (BPU)

The BPU allows the user to set hardware breakpoints. It contains four comparators that monitor the instruction fetch address. If a match occurs, the instruction comparators can be configured to generate a breakpoint event.

For more information on the breakpoint unit and how to use it, refer to [\[4\]](#).

37.7.1 BPU registers

The BPU registers are accessible to the debugger via access port AP0. They are not accessible by software running on the CPU.

BPU control register (BPU_CTRL)

Address offset: 0x000

Reset value: 0x0000 0040

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

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

Bits 7:4 **NUM_CODE[3:0]**: number of instruction address comparators supported

0x04: 8 instruction comparators supported

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY**: Write protect key.

This bit must be set when writing to BPU_CTRL register, otherwise the write is ignored.

Bit 0 **ENABLE**: BPU enable

0: disabled

1: enabled

BPU comparator x register (BPU_COMPx)

Address offset: 0x008 + 0x004 * x, (x = 0 to 3)

Reset value: 0xFFFF XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---------------|----|------|-------------|----|----|----|----|----|----|----|----|----|------|--------|----|
| BP_MATCH[1:0] | | Res. | COMP[28:16] | | | | | | | | | | | | |
| 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 |
| COMP[15:2] | | | | | | | | | | | | | Res. | ENABLE | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:30 **BP_MATCH[1:0]**: behavior when COMP field is matched to the instruction fetch address and ENABLE = 1.

00: no breakpoint

01: breakpoint if lower half-word of address matches (16-bit and 32-bit instructions)

10: breakpoint if upper half-word of address matches (16-bit instructions only)

11: breakpoint if both lower and upper half-word of address match (16 and 32-bit instructions)

Bit 29 Reserved, must be kept at reset value.

Bits 28:2 **COMP[28:2]**: Bits 28:2 of the comparison address.

Bits 31:29 and 1:0 of the comparison address are fixed to 0. The comparison address is compared with the instruction address fetched from code memory.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: breakpoint enable

0: disabled

1: enabled

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | | | SIZE[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 **SIZE[3:0]**: register file size

0x0: The register file occupies a single 4-Kbyte region.

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

0x4: Arm® JEDEC code

BPU CoreSight peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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]
0x0B: BPU part number

BPU CoreSight peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|---------------|------|------|------|---------------|------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | JEP106ID[3:0] | | | | PARTNUM[11:8] | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

Bits 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

BPU CoreSight peripheral identity register 2 (BPU_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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVISION[3:0]**: component revision number

0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value

0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]

0x3: Arm® JEDEC code

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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | r | r | r | r | r | r | r | r |
| REVAND[3:0] | | | | | | | | | | CMOD[3:0] | | | | | |
| | | | | | | | | | | | | | | | |

Bits 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

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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | r | r | r | r | r | r | r | r |
| PREAMBLE[7:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

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

Bits 7:0 PREAMBLE[7:0]: component identification bits [7:0]

0x0D: common identification value

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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | r | r | r | r | r | r | r | r |
| CLASS[3:0] | | | | | | | | | | PREAMBLE[11:8] | | | | | |
| | | | | | | | | | | | | | | | |

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

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0xE: generic IP

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 identification bits [23:16]
0x05: common identification value

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. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 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 identification bits [31:24]
0xB1: common identification value

37.7.2 BPU register map

Refer to [Table 248: Processor ROM table](#) for register boundary addresses.

Table 253. BPU register map and reset values

37.8 System control space (SCS)

The debug related registers in the Cortex®-M0+ core can be accessed by the debugger in the system control space. These registers can be used to stop and start the processor in debug mode and to step an instruction. They can also be used to access core registers.

For more detailed information on how to use the SCS registers, refer to the ARM architecture v6M reference manual [\[4\]](#).

37.8.1 SCS registers

The system control space registers are accessible to the debugger via access port AP0.

System handler control and state register (SCS_SHCSR)

Address offset: 0xD24

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 |
| SVCALLPENDED | Res. |
| rw | | | | | | | | | | | | | | | |

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

Bit 15 **SVCALLPENDED**: Supervisor call pending.

- 0: SVCall not pending
- 1: SVCall pending

Bits 14:0 Reserved, must be kept at reset value.

Debug fault status register (SCS_DFSR)

Address offset: 0xD30

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. |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | EXTER NAL | VCATC H | DWTTRAP | BKPT | HALTE D |
| | | | | | | | | | | | r | r | r | r | r |

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

Bit 4 **EXTERNAL**: Asynchronous external debug event (EDBGRQ)

- 0: no EDBGRQ debug event generated
- 1: EDBGRQ debug event generated

Bit 3 **VCATCH**: Vector catch debug event

- 0: no vector catch debug event generated
- 1: vector catch debug event generated

Bit 2 **DWTTRAP**: DWT generated debug event

- 0: no debug events generated by DWT
- 1: one or more debug event generated by DWT

Bit 1 **BKPT**: Breakpoint debug event

- 0: No breakpoint debug event
- 1: At least one breakpoint event generated by a BKPT instruction or a BPU comparator match

Bit 0 **HALTED**: Halt or step request debug event

- 0: No active halt request debug event
- 1: Halt request debug event generated by setting C_HALT or C_STEP in the DHCSR register

Debug halting control and status register (SCS_DHCSR)

Address offset: 0xDF0

Reset value: 0XXXXX XXXX

When writing:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------------|------|------|------|------|------|------|------|------|------|------|------|-------------|--------|--------|------------|
| DBGKEY[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. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | C_MASK_INTS | C_STEP | C_HALT | C_DEBUG_EN |
| | | | | | | | | | | | | rw | rw | rw | rw |

Bits 31:16 **DBGKEY[15:0]**: Debug key.

Software must write 0xA05F to this field to enable write accesses to bits 15:0, otherwise the processor ignores the write access.

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

Bit 3 **C_MASKINTS**: Mask interrupts.

This bit must be programmed while C_HALT = 1, before setting C_HALT = 0

- 0: do not mask interrupts

- 1: mask PendSV, SysTick and external configurable interrupts when restarting or stepping

Bit 2 **C_STEP**: Single stepping.

- 0: disable single stepping
- 1: enable single stepping.

Bit 1 **C_HALT**: Halt/run request.

- 0: request processor to exit debug and run (if C_STEP = 0) or single step (if CSTEP = 1)
- 1: request processor to halt if running or to remain in debug

Bit 0 **C_DEBUGEN**: Halting debug enable

- 0: disable halting debug. C_STEP and C_HALT values are ignored.
- 1: enable halting debug

Debug halting control and status register [alternate] (SCS_DHCSR)

Address offset: 0xDF0

Reset value: 0x0000 0000

When reading:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------------|-------------|------|------|------|------|-------------|---------|--------|-----------|
| Res. | Res. | Res. | Res. | Res. | Res. | S_RESET_ST | S_RETIRE_ST | Res. | Res. | Res. | Res. | S_LOCKUP | S_SLEEP | S_HALT | S_REG_RDY |
| | | | | | | rc_r | rc_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. | C_MASK_INTS | C_STEP | C_HALT | C_DEBUGEN |
| | | | | | | | | | | | | r | r | r | r |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **S_RESET_ST**: Sticky reset status.

This bit indicates whether the processor has been reset since last read of SCS_DHCSR.
It is cleared on reading SCS_DHCSR.

- 0: No reset
- 1: At least one reset has occurred

Bit 24 **S_RETIRE_ST**: Sticky instruction complete status.

This bit indicates whether the processor has completed execution of an instruction since last read of SCS_DHCSR. It is cleared on reading SCS_DHCSR

- 0: No instruction has completed
- 1: At least one instruction has completed

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **S_LOCKUP**: Lockup status.

This bit indicates whether the processor is locked up due to an unrecoverable exception. It is cleared on entering debug state.

- 0: No instruction has completed
- 1: At least one instruction has completed

Bit 18 **S_SLEEP**: Sleep status. Indicates whether the processor is sleeping.

- 0: Not sleeping
- 1: Sleeping

Bit 17 **S_HALT**: Halt status. Indicates whether the processor is in debug state.

- 0: Not in debug
- 1: In debug state

Bit 16 **S_REGRDY**: DCRDR register ready.

Handshake for transfers through the SCS_DCRDR register. This bit is cleared when a write occurs to the DCRSR and set when the transfer completes.

- 0: DCRDR transfer in progress
- 1: DCRDR transfer complete

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

Bit 3 **C_MASKINTS**: Mask interrupts.

The read value is unknown if C_DEBUGEN is cleared.

- 0: interrupts are not masked
- 1: PendSV, SysTick and external configurable interrupts are masked

Bit 2 **C_STEP**: Single stepping.

The read value is unknown if C_DEBUGEN is cleared.

- 0: single stepping disabled
- 1: single stepping enabled

Bit 1 **C_HALT**: Halt request.

The read value is unknown if C_DEBUGEN is cleared.

- 0: processor running
- 1: processor halted

Bit 0 **C_DEBUGEN**: Halting debug enable

- 0: halting debug disabled
- 1: halting debug enabled

Debug core register selector register (SCS_DCRSR)

Address offset: 0xDF4

Reset value: 0x0000 0000

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

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

Bit 16 **REGWNR**: Transfer direction.

- 0: read from register into SCS_DCRDR
- 1: write contents of SCS_DCRDR to register

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **REGSEL[4:0]**: Register select.

This field specifies the Cortex®-M0+ core register or special purpose register targeted for the transfer.

0b00000 - 0b01100: R{v}

0b01101: Current SP

0b01110: LR

0b01111: Debug return address

0b10000: xPSR

0b10001: MSP, main stack pointer

0b10010: PSP, process stack pointer

0b10100: bits 31:24 - CONTROL; bits 23:8 - reserved; bits 7:0 PRIMASK

Others: reserved, must not be used

Debug core register data register (SCS_DCRDR)

Address offset: 0xDF8

Reset value: 0x0000 0000

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

Bits 31:0 **DBGTMP[31:0]**: Temporary data register for reading and writing core registers.

Data written by the debugger to this register are transferred into the register specified in the DCRSR.REGSEL field, when the latter is written with REGWNR bit set.

When DCRCR.REGSEL is written with REGWNR bit cleared, the contents of the specified register are transferred into this register and can be read by the debugger.

In both cases the DHSCR.S_REGRDY bit indicates when the transfer is complete.

Debug exception and monitor control register (SCS_DEMCR)

Address offset: 0xDFC

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. | DWTE NA | Res. |
| | | | | | | | rw | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | Res. | Res. | Res. | Res. | VC HARDE RR | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | VC CORE RESET |
| | | | | | rw | | | | | | | | | | rw |

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

Bit 24 **DWTENA**: DWT enable.

- 0: DWT disabled
- 1: DWT enabled

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **VC_HARDERR**: Hard fault vector catch.

- 0: Hard fault debug trap disabled
- 1: Hard fault halts the processor in debug (if DHSCR.C_DEBUGEN = 1)

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **VC_COREREFRESH**: Reset vector catch.

- 0: Reset vector catch disabled
- 1: Local reset halts the processor in debug (if DHSCR.C_DEBUGEN = 1)

37.8.2 SCS register mapRefer to [Table 248: Processor ROM table](#) for SCS register file base address.**Table 254. SCS 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 | |
|----------------|------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|----|----|
| 0xD24 | SCS_SHCSR | Res. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD28 to 0xD2C | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD30 | SCS_DFSR | Res. | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD34 to 0xDEC | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 254. SCS 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 |
|--------|---------------------------------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|---|---|
| 0xDF0 | SCS_DHCSR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 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 | | |
| 0xDF0 | [alternate] SCS_DHCSR | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xDF4 | SCS_DCRSR | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xDF8 | SCS_DCRDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 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 | | |
| 0xDFC | SCS_DEMCR | Res. | Res. | Res. | Res. | Res. | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

37.9 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 software) to:

- maintain the clock and power to the processor cores when in low-power modes (sleep, stop or standby)
- maintain the clock and power to the system debug and trace components when in low-power modes
- stop the clock to certain peripherals (SMBUS timeout, watchdogs, timers, RTC) when either processor core is stopped in debug mode

The DBGMCU also supports the debug authentication feature for reopening debug on a closed device.

37.9.1 Device ID

The DBGMCU includes an identity code register, DBGMCU_IDCODE. This register contains the ID code for the device. Debug tools can locate this register via the CoreSight™ discovery procedure described in [Section 37.5: ROM tables](#).

37.9.2 Low-power mode emulation

When the device enters either stop mode (clocks are stopped) or standby mode (core power is switched off), the debugger can no longer access the debug access port and loses the connection with the device. To avoid this, the debugger (or software) can set the

DBG_STANDBY and/or DBG_STOP bits in the [DBGMCU configuration register \(DBGMCU_CR\)](#). These bits, when set, maintain the clock and power to the processor while the device is in the corresponding low-power mode. The processor remains in Sleep mode, and exits the low-power mode in the normal way. However, peripheral devices continue to operate, so the device behavior may not be identical to that of the actual low-power mode.

37.9.3 Peripheral clock freeze

The DBGMCU peripheral clock freeze registers allow the operation of certain peripherals to be suspended in debug mode. The peripheral units, which support this feature are listed in the table below.

Table 255. Peripheral clock freeze control bits

| Bus | Control register | Peripheral | Description |
|-----|------------------|------------|-------------------------|
| APB | DBGMCU_APB1FZR | TIM2 | General purpose timer 2 |
| | | TIM3 | General purpose timer 3 |
| | | TIM6 | General purpose timer 6 |
| | | TIM7 | General purpose timer 7 |
| | | RTC | Real time clock |
| | | WWDG | Window watchdog |
| | | IWDG | Independent watchdog |
| | | LPTIM1 | Low power timer 1 |
| | | LPTIM2 | Low power timer 2 |
| | | TIM1 | General purpose timer 1 |
| APB | DBGMCU_APB2FZR | TIM15 | General purpose timer 3 |
| | | TIM16 | General purpose timer 2 |
| | | LPTIM3 | Low power timer 3 |

Each peripheral unit or DMA channel has a corresponding control bit, DBG_xxx_STOP, where xxx is the acronym of the peripheral (or DMA channel). The control bits are organized in DBGMCU_zzzFZR registers, where zzz corresponds to the name of the bus. For example, DBGMCU_APB1FZR contains the control bits for peripherals on the APB bus.

The control bit, when set, causes the corresponding peripheral operation to be suspended when the CPU is stopped in debug (HALTED = 1), according to the table below:

Table 256. Peripheral behavior in debug mode

| HALTED | DBG_xxx_STOP | Peripheral behavior |
|--------|--------------|-----------------------------|
| 0 | X | The operation continues. |
| 1 | 0 | The operation continues. |
| 1 | 1 | The operation is suspended. |

37.9.4 DBGMCU registers

The DBGMCU registers are not reset by a system reset, only by a power-on reset. They are accessible to the debugger via the APB access port AP0 at base address 0xF000 1000, and to the software at base address 0x4001 5800.

DBGMCU device ID code register (DBGMCU_IDCODE)

Address offset: 0x000

Reset value: 0xFFFF 6XXX

Only 32-bit access supported.

This read-only register allows identification of the device and its die revision. It is always accessible by the debugger or the user software.

| 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 |
| DEV_ID[11:0] | | | | | | | | | | | | | | | |
| Res. | Res. | Res. | Res. | | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:16 **REV_ID[15:0]**: Revision identifier

This field indicates the revision of the device.

0x1000: Revision A for STM32U031/73/83xx

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

Upon read, these reserved bits return 0b0110.

Bits 11:0 **DEV_ID[11:0]**: Device identifier

This field indicates the device ID.

0x459: STM32U031xx

0x489: STM32U073xx/083xx

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x0000 0004

Reset value: 0x0000 0000 (power-on reset)

Only 32-bit access supported.

This register is only accessible when debug is open.

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

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

Bit 2 **DBG_STANDBY:** Debug Standby and Shutdown modes

Debug options in Standby or Shutdown mode.

0: Digital part powered. From software point of view, exiting Standby and Shutdown modes is identical as fetching reset vector (except for status bits indicating that the MCU exits Standby)

1: Digital part powered and FCLK and HCLK running, derived from the internal RC oscillator remaining active. The MCU generates a system reset so that exiting Standby and Shutdown has the same effect as starting from reset.

Bit 1 **DBG_STOP:** Debug Stop mode

Debug options in Stop mode.

0: All clocks disabled, including FCLK and HCLK. Upon Stop mode exit, the CPU is clocked by the HSI internal RC oscillator.

1: FCLK and HCLK running, derived from the internal RC oscillator remaining active. If SysTick is enabled, it may generate periodic interrupt and wake up events.Upon Stop mode exit, the software must re-establish the desired clock configuration.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB1 freeze register (DBGMCU_APB1FZR)

Address offset: 0x08

Reset value: 0x0000 0000 (power-on reset)

Only 32-bit access are supported.

This register is only accessible when debug is open

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-----------------|-----------------|------|---------------|---------------|--------------|------|------|------|------|---------------|---------------|------|------|---------------|---------------|
| DBG_LPTIM1_STOP | DBG_LPTIM2_STOP | Res. | Res. | Res. | Res. | 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 |
| Res. | Res. | Res. | DBG_IWDG_STOP | DBG_WWDG_STOP | DBG_RTC_STOP | Res. | Res. | Res. | Res. | DBG_TIM7_STOP | DBG_TIM6_STOP | Res. | Res. | DBG_TIM3_STOP | DBG_TIM2_STOP |
| | | | rw | rw | rw | | | | | rw | rw | | | rw | rw |

- Bit 31 **DBG_LPTIM1_STOP**: LPTIM1 stop in debug
0: normal operation. LPTIM1 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM1 is frozen while CPU is in debug mode.
- Bit 30 **DBG_LPTIM2_STOP**: LPTIM2 stop in debug
0: normal operation. LPTIM2 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM2 is frozen while CPU is in debug mode.
- Bits 29:13 Reserved, must be kept at reset value.
- Bit 12 **DBG_IWDG_STOP**: IWDG stop in debug
0: normal operation. IWDG continues to operate while CPU is in debug mode.
1: stop in debug. IWDG is frozen while CPU is in debug mode.
- Bit 11 **DBG_WWDG_STOP**: WWDG stop in debug
0: normal operation. WWDG continues to operate while CPU is in debug mode.
1: stop in debug. WWDG is frozen while CPU is in debug mode.
- Bit 10 **DBG_RTC_STOP**: RTC stop in debug
0: normal operation. RTC counter continues to operate while CPU is in debug mode.
1: stop in debug. RTC counter is frozen while CPU is in debug mode.
- Bits 9:6 Reserved, must be kept at reset value.
- Bit 5 **DBG_TIM7_STOP**: TIM7 stop in debug
0: normal operation. TIM7 continues to operate while CPU is in debug mode.
1: stop in debug. TIM7 is frozen while CPU is in debug mode.
- Bit 4 **DBG_TIM6_STOP**: TIM6 stop in debug
0: normal operation. TIM6 continues to operate while CPU is in debug mode.
1: stop in debug. TIM6 is frozen while CPU is in debug mode.
- Bits 3:2 Reserved, must be kept at reset value.
- Bit 1 **DBG_TIM3_STOP**: TIM3 stop in debug
0: normal operation. TIM3 continues to operate while CPU is in debug mode.
1: stop in debug. TIM3 is frozen while CPU is in debug mode.
- Bit 0 **DBG_TIM2_STOP**: TIM2 stop in debug
0: normal operation. TIM2 continues to operate while CPU is in debug mode.
1: stop in debug. TIM2 is frozen while CPU is in debug mode.

DBG APB2 freeze register (DBGMCU_APB2FZR)

Address offset: 0x0C

Reset value: 0x0000 0000 (power-on reset)

This register is only accessible when debug is open

| 31 | 30 | 29 | 28 | 27 | 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_LPTIM3_STOP | DBG_TIM16_STOP | DBG_TIM15_STOP |
| | | | | | | | | | | | | | rw | 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. |
| | | | | rw | | | | | | | | | | | |

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

Bit 18 **DBG_LPTIM3_STOP**: LPTIM3 stop in debug

- 0: normal operation. LPTIM3 continues to operate while CPU is in debug mode.
- 1: stop in debug. LPTIM3 is frozen while CPU is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in debug

- 0: normal operation. TIM16 continues to operate while CPU is in debug mode.
- 1: stop in debug. TIM16 is frozen while CPU is in debug mode.

Bit 16 **DBG_TIM15_STOP**: TIM15 stop in debug

- 0: normal operation. TIM15 continues to operate while CPU is in debug mode.
- 1: stop in debug. TIM15 is frozen while CPU is in debug mode.

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

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in debug

- 0: normal operation. TIM1 continues to operate while CPU is in debug mode.
- 1: stop in debug. TIM1 is frozen while CPU is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

DBGMCU status register (DBGMCU_SR)

Address offset: 0xFC

Reset value: 0x0001 0003

This register is always accessible.

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------------|-------------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Res. | AP0_ENABLED | AP1_ENABLED |
| | | | | | | | | | | | | | | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | AP0_PRESENT | AP1_PRESENT |
| | | | | | | | | | | | | | | r | r |

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

Bit 17 **AP0_ENABLED**: Identifies whether access port AP0 is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)

- 0: AP0 locked
- 1: AP0 enabled

Bit 16 **AP1_ENABLED**: Identifies whether access port AP1 is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)

- 0: AP1 locked
- 1: AP1 enabled

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

Bit 1 **AP0_PRESENT**: Identifies whether access port AP0 is present in device

- 1: AP0 present

Bit 0 **AP1_PRESENT**: Identifies whether access port AP1 is present in device

- 1: AP1 present

**DBGMCU debug authentication mailbox host register
(DBGMCU_DBG_AUTH_HOST)**

Address offset: 0x100

Reset value: 0xXXXX XXXX

This register is read only when accessed by the CPU, writes have no effect.

This register can be written and read by an external debugger.

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

Bits 31:0 **MESSAGE[31:0]**: Debug host to device mailbox message.

During debug authentication the debug host communicates with the device via this register.

DBGMCU debug authentication mailbox device register (DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104

Reset value: 0xXXXX XXXX

This register is read only when accessed via the debug port or the CPU, writes have no effect.

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

Bits 31:0 **MESSAGE[31:0]**: Device to debug host mailbox message.

During debug authentication the device communicates with the debug host via this register.

DBGMCU CoreSight peripheral identity register 4 (DBGMCU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

This register is always accessible.

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

0x0: The register file occupies a single 4-Kbyte region.

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

0x0: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 0 (DBGMCU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

This register is always accessible.

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

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

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]

0x00: DBGMCU part number

DBGMCU CoreSight peripheral identity register 1 (DBGMCU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0000

This register is always accessible.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---------------|------|------|------|------|------|------|------|------|------|---------------|------|------|------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 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. |
| | | | | | | | | r | r | r | r | r | r | r | r |
| JEP106ID[3:0] | | | | | | | | | | PARTNUM[11:8] | | | | | |
| | | | | | | | | | | | | | | | |

Bits 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]

0x0: DBGMCU part number

DBGMCU CoreSight peripheral identity register 2 (DBGMCU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

This register is always accessible.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------|------|------|------|------|------|------|------|---------------|------|------|------|-------|---------------|------|------|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res. | REVISION[3:0] | | | | JEDEC | JEP106ID[6:4] | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

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

Bits 7:4 **REVISION[3:0]**: component revision number

0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value

0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]

0x2: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 3 (DBGMCU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

This register is always accessible.

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

Bits 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

DBGMCU CoreSight component identity register 0 (DBGMCU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

This register is always accessible.

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

0xD: common identification value

DBGMCU CoreSight component identity register 1 (DBGMCU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

This register is always accessible.

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

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

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class

0xF: Non-CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]

0x0: common identification value

DBGMCU CoreSight component identity register 2 (DBGMCU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

This register is always accessible.

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

0x05: common identification value

DBGMCU CoreSight component identity register 3 (DBGMCU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

This register is always accessible.

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

0xB1: common identification value

37.9.5 DBGMCU register mapThe following table summarizes the DBGMCU registers. Refer to [Table 249: System ROM table register map and reset values](#) for the register file base address.**Table 257. DBGMCU 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 | DBGMCU_IDCODE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value ⁽¹⁾ | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

Table 257. DBGMCU register map and reset values (continued)

Table 257. DBGMCU 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 |
|--------|----------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------------------|----------------|-----------------|---|---|---|---|---|
| 0xFE8 | DBGMCU_PID_R2 | Res. | REVISION [3:0] | JEDEC [6:4] | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 0 0 0 1 0 1 0 | | | | | | | |
| 0xFEC | DBGMCU_PID_R3 | Res. | REVAND[3:0] | CMOD[3:0] | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 0 0 0 0 0 0 0 0 | | | | | | | |
| 0xFF0 | DBGMCU_CID_R0 | Res. | PREAMBLE[7:0] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 0 0 0 1 1 0 1 | | | | | | | |
| 0xFF4 | DBGMCU_CID_R1 | Res. | CLASS[3:0] | PREAMBLE [11:8] | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 1 1 1 0 0 0 0 0 | | | | | | | |
| 0xFF8 | DBGMCU_CID_R2 | Res. | PREAMBLE[19:12] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 0 0 0 0 0 0 1 0 1 | | | | | | | |
| 0xFFC | DBGMCU_CID_R3 | Res. | PREAMBLE[27:20] | | | | | | | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | 1 0 1 1 0 0 0 0 1 | | | | | | | |

1. The reset value is product dependent. For more information, refer to [Section : DBGMCU device ID code register \(DBGMCU_IDCODE\)](#).

37.10 Reference documents

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G, 16th March 2015
3. DDI 0484C (ID011713) - Arm® Cortex®-M0+ Processor r0p1 Technical Reference Manual, Issue C, 16 Dec 2012
4. DDI 0419C (ID092410) - Arm® v6-M Architecture Reference Manual, Issue C, September 2010

38 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 to the characteristics of the STM32U0 series microcontroller.

38.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order 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 internal flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0xFFFF 6E50 (STM32U073/83xx), 0xFFFF 3E50 (STM32U031xx)

Address offset: 0x00

Reset value: 0xFFFF XXXX 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

Reset value: 0xFFFF XXXX 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

Reset value: 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)

38.2 Flash memory size data register

Base address: 0x1FFF 6EA0 (STM32U073/83xx), 0x1FFF 3EA0 (STM32U031xx)

Address offset: 0x00

Reset value: 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**

This bitfield indicates the size of the device flash memory expressed in Kbytes.

As an example, 0x040 corresponds to 64 Kbytes.

38.3 Package data register

Base address: 0x1FFF 6D00 (STM32U073/83xx), 0x1FFF 3D00 (STM32U031xx)

Address offset: 0x00

Reset value: 0xXXXX where X is factory-programmed

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Res. |
| | | | | | | | | | | | | r | r | r | r |

Bits 15:4 Reserved

Bits 3:0 **PKG[3:0]**: Package type

Condition: STM32U031xx

0000 : Reserved

0001 : UFQFPN32

0011 : UFQFPN48

0100 : LQFP48

0101 : LQFP64

0110 : BGA64

1001 : TSSOP20

1010 : WL CSP29

1011 : LQFP32

Others: Reserved

Condition: STM32U0x3xx

0000 : Reserved

0001 : UFQFPN32

0010 : WL CSP42

0011 : UFQFPN48

0100 : LQFP48

0101 : LQFP64

0110 : BGA64

0111 : LQFP80

1000 : BGA81

Others : Reserved

Appendix A OEM key CRC calculation source code

Below is the source code that can be used to compare the result of the CRC computation performed on the loaded OEM1/2 keys (option byte loading or programming of new keys), with the CRC values of the OEM1/2 keys available in the OEM1KEYCRC/OEM2KEYCRC bitfields of the FLASH_OEMKEYSR register.

```
uint8_t getCRC(uint32_t * keyin)
{
    const uint8_t CRC7_POLY = 0x7;
    const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0};
    uint8_t i, j, k, crc = 0x0;
    uint32_t keyval;

    for (j = 0; j < 4; j++)
    {
        keyval = *(keyin+j);
        if (j == 0)
        {
            keyval ^= key_strobe[0];
        } else
        {
            keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8) | crc;
        }
        for (i = 0, crc = 0; i < 32; i++)
        {
            k = (((crc >> 7) ^ (keyval >> (31-i))&0xF) & 1;
            crc <= 1;
            if (k)
            {
                crc ^= CRC7_POLY;
            }
        }
        crc^=0x55;
    }
    return crc;
}
```

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.

Revision history

Table 258. Document revision history

| Date | Revision | Changes |
|-------------|----------|--|
| 14-Mar-2024 | 1 | Initial release. |
| 26-Mar-2024 | 2 | Added <i>Section 26: General-purpose timers (TIM15/TIM16)</i> . <i>Section 33: Universal synchronous/asynchronous receiver transmitter (USART/UART)</i> : Removed Table <i>USART interconnection (USART1/2/3/4 and LPUART1/2/3)</i> . |

Index

A

| | |
|-------------------|---------|
| ADC_AWD1TR | 374 |
| ADC_AWD2CR | 380 |
| ADC_AWD2TR | 375 |
| ADC_AWD3CR | 380 |
| ADC_AWD3TR | 379 |
| ADC_CALFACT | 381 |
| ADC_CCR | 381 |
| ADC_CFGR1 | 369 |
| ADC_CFGR2 | 372 |
| ADC_CHSELR | 376-377 |
| ADC_CR | 367 |
| ADC_DR | 379 |
| ADC_IER | 365 |
| ADC_ISR | 363 |
| ADC_SMPR | 373 |
| AES_CR | 538 |
| AES_DINR | 542 |
| AES_DOUTR | 542 |
| AES_IVR0 | 544 |
| AES_IVR1 | 545 |
| AES_IVR2 | 545 |
| AES_IVR3 | 545 |
| AES_KEYR0 | 543 |
| AES_KEYR1 | 543 |
| AES_KEYR2 | 544 |
| AES_KEYR3 | 544 |
| AES_KEYR4 | 546 |
| AES_KEYR5 | 546 |
| AES_KEYR6 | 546 |
| AES_KEYR7 | 547 |
| AES_SR | 540 |
| AES_SUSPxR | 547 |
| AP0_CSW | 1257 |
| AP1_CSW | 1256 |
| APx_BASE | 1259 |
| APx_BDn | 1259 |
| APx_DRW | 1258 |
| APx_IDR | 1260 |
| APx_TAR | 1258 |

B

| | |
|-----------------|------|
| BPU_CIDR0 | 1290 |
| BPU_CIDR1 | 1290 |
| BPU_CIDR2 | 1291 |
| BPU_CIDR3 | 1291 |
| BPU_COMPx | 1287 |

| | |
|-----------------|------|
| BPU_CTRL | 1287 |
| BPU_PIDR0 | 1288 |
| BPU_PIDR1 | 1289 |
| BPU_PIDR2 | 1289 |
| BPU_PIDR3 | 1290 |
| BPU_PIDR4 | 1288 |

C

| | |
|-----------------------|------------------|
| C1ROM_CIDR3 | 1268, 1273, 1278 |
| COMP1_CSR | 420 |
| COMP2_CSR | 421 |
| CPUROM_CIDR0 | 1276 |
| CPUROM_CIDR1 | 1277 |
| CPUROM_CIDR2 | 1277 |
| CPUROM_CIDR3 | 1277 |
| CPUROM_MEMTYPER | 1274 |
| CPUROM_PIDR0 | 1275 |
| CPUROM_PIDR1 | 1275 |
| CPUROM_PIDR2 | 1275 |
| CPUROM_PIDR3 | 1276 |
| CPUROM_PIDR4 | 1274 |
| CRC_CR | 323 |
| CRC_DR | 322 |
| CRC_IDR | 322 |
| CRC_INIT | 324 |
| CRC_POL | 324 |
| CRS_CFGR | 220 |
| CRS_CR | 219 |
| CRS_ICR | 223 |
| CRS_ISR | 221 |

D

| | |
|----------------------|------|
| DAC_CCR | 405 |
| DAC_CR | 400 |
| DAC_DHR12L1 | 403 |
| DAC_DHR12R1 | 402 |
| DAC_DHR8R1 | 403 |
| DAC_DOR1 | 404 |
| DAC_MCR | 405 |
| DAC_SHRR | 407 |
| DAC_SHRR | 407 |
| DAC_SHSR1 | 406 |
| DAC_SR | 404 |
| DAC_SWTRGR | 402 |
| DBGMCU_APB1FZR | 1302 |
| DBGMCU_APB2FZR | 1304 |
| DBGMCU_CIDR0 | 1309 |

| | | | |
|------------------------------|------|-----------------------|------|
| DBGMCU_CIDR1 | 1309 | EXTI_EMR2 | 316 |
| DBGMCU_CIDR2 | 1310 | EXTI_EXTICRx | 314 |
| DBGMCU_CIDR3 | 1310 | EXTI_FPR1 | 313 |
| DBGMCU_CR | 1301 | EXTI_FTSR1 | 312 |
| DBGMCU_DBG_AUTH_DEVICE | 1306 | EXTI_IMR1 | 315 |
| DBGMCU_DBG_AUTH_HOST | 1305 | EXTI_IMR2 | 316 |
| DBGMCU_IDCODE | 1301 | EXTI_RPR1 | 313 |
| DBGMCU_PIDR0 | 1307 | EXTI_RTSR1 | 311 |
| DBGMCU_PIDR1 | 1307 | EXTI_SWIER1 | 312 |
| DBGMCU_PIDR2 | 1308 | | |
| DBGMCU_PIDR3 | 1308 | | |
| DBGMCU_PIDR4 | 1306 | | |
| DBGMCU_SR | 1305 | F | |
| DMA_CCRx | 278 | FLASH_ACR | 86 |
| DMA_CMARx | 282 | FLASH_CR | 91 |
| DMA_CNDTRx | 281 | FLASH_ECCR | 92 |
| DMA_CPARx | 281 | FLASH_HDPCR | 101 |
| DMA_IFCR | 277 | FLASH_HDPEXTR | 101 |
| DMA_ISR | 274 | FLASH_KEYR | 88 |
| DMAMUX_CFR | 296 | FLASH_OEM1KEYR1 | 97 |
| DMAMUX_CSR | 296 | FLASH_OEM1KEYR2 | 97 |
| DMAMUX_CxCR | 295 | FLASH_OEM1KEYR3 | 98 |
| DMAMUX_RGCFR | 298 | FLASH_OEM1KEYR4 | 98 |
| DMAMUX_RGSR | 298 | FLASH_OEM2KEYR1 | 98 |
| DMAMUX_RGxCR | 297 | FLASH_OEM2KEYR2 | 99 |
| DP_ABORT | 1248 | FLASH_OEM2KEYR3 | 99 |
| DP_CTRLSTAT | 1249 | FLASH_OEM2KEYR4 | 100 |
| DP_DLCSR | 1250 | FLASH_OEMKEYSR | 100 |
| DP_DLPIDR | 1251 | FLASH_OPTKEYR | 88 |
| DP_DPIDR | 1247 | FLASH_OPTR | 93 |
| DP_RDBUFF | 1252 | FLASH_SECR | 96 |
| DP_RESEND | 1251 | FLASH_SR | 88 |
| DP_SELECT | 1252 | FLASH_WRP1AR | 95 |
| DP_TARGETID | 1250 | FLASH_WRP1BR | 96 |
| DWT_CIDR0 | 1283 | | |
| DWT_CIDR1 | 1284 | G | |
| DWT_CIDR2 | 1284 | GPIOx_AFRH | 239 |
| DWT_CIDR3 | 1284 | GPIOx_AFRL | 239 |
| DWT_COMPx | 1280 | GPIOx_BRR | 240 |
| DWT_CTRL | 1279 | GPIOx_BSRR | 237 |
| DWT_FUNCTIONx | 1281 | GPIOx_IDR | 236 |
| DWT_MASKx | 1280 | GPIOx_LCKR | 237 |
| DWT_PCSR | 1280 | GPIOx_MODER | 234 |
| DWT_PIDR0 | 1282 | GPIOx_ODR | 236 |
| DWT_PIDR1 | 1282 | GPIOx_OSPEEDR | 235 |
| DWT_PIDR2 | 1282 | GPIOx_OTYPER | 235 |
| DWT_PIDR3 | 1283 | GPIOx_PUPDR | 236 |
| DWT_PIDR4 | 1281 | | |
| E | | I | |
| EXTI_EMR1 | 315 | I2C_CR1 | 998 |
| | | I2C_CR2 | 1000 |
| | | I2C_ICR | 1007 |

| | |
|-------------------|------|
| I2C_ISR | 1005 |
| I2C_OAR1 | 1002 |
| I2C_OAR2 | 1003 |
| I2C_RXDR | 1007 |
| I2C_TIMINGR | 1004 |
| I2C_TXDR | 1008 |
| IWDG_EWCR | 890 |
| IWDG_KR | 887 |
| IWDG_PR | 887 |
| IWDG_RLR | 888 |
| IWDG_SR | 888 |
| IWDG_WINR | 890 |

L

| | |
|--------------------|------------|
| LCD_CLR | 458 |
| LCD_CR | 453 |
| LCD_FCR | 454 |
| LCD_RAMx | 458-459 |
| LCD_SR | 456 |
| LPTIM_ARR | 865 |
| LPTIM_CCMR1 | 867 |
| LPTIM_CCMR2 | 870 |
| LPTIM_CCR1 | 864 |
| LPTIM_CCR2 | 872 |
| LPTIM_CCR3 | 873 |
| LPTIM_CCR4 | 874 |
| LPTIM_CFGR | 860 |
| LPTIM_CFGR2 | 865 |
| LPTIM_CNT | 865 |
| LPTIM_CR | 863 |
| LPTIM_RCR | 867 |
| LPTIMx_DIER | 856, 858 |
| LPTIMx_ICR | 853, 855 |
| LPTIMx_ISR | 848, 850 |
| LPUART_BRR | 1147 |
| LPUART_CR1 | 1133, 1136 |
| LPUART_CR2 | 1139 |
| LPUART_CR3 | 1141, 1144 |
| LPUART_ICR | 1155 |
| LPUART_ISR | 1148, 1152 |
| LPUART_PRESC | 1157 |
| LPUART_RDR | 1156 |
| LPUART_RQR | 1147 |
| LPUART_TDR | 1157 |

M

| | |
|----------------------|------|
| MCUROM_CIDR0 | 1271 |
| MCUROM_CIDR1 | 1272 |
| MCUROM_CIDR2 | 1272 |
| MCUROM_CIDR3 | 1272 |
| MCUROM_MEMTYPE | 1269 |

| | |
|--------------------|------|
| MCUROM_PIDR0 | 1270 |
| MCUROM_PIDR1 | 1270 |
| MCUROM_PIDR2 | 1270 |
| MCUROM_PIDR3 | 1271 |
| MCUROM_PIDR4 | 1269 |

O

| | |
|--------------------|-----|
| OPAMP1_CSR | 433 |
| OPAMP1_LPOTR | 434 |
| OPAMP1_OTR | 434 |

P

| | |
|-----------------|-----|
| PWR_CR1 | 133 |
| PWR_CR2 | 134 |
| PWR_CR3 | 135 |
| PWR_CR4 | 137 |
| PWR_PDCRA | 142 |
| PWR_PDCRB | 144 |
| PWR_PDCRC | 145 |
| PWR_PDCRD | 146 |
| PWR_PDCRE | 147 |
| PWR_PDCRF | 148 |
| PWR_PUCRA | 142 |
| PWR_PUCRB | 143 |
| PWR_PUCRC | 144 |
| PWR_PUCRD | 145 |
| PWR_PUCRE | 146 |
| PWR_PUCRF | 148 |
| PWR_SCR | 141 |
| PWR_SR1 | 138 |
| PWR_SR2 | 139 |

R

| | |
|---------------------|-----|
| RCC_AHBENR | 189 |
| RCC_AHBRSTR | 182 |
| RCC_AHBSMENR | 196 |
| RCC_APBENR1 | 192 |
| RCC_APBENR2 | 195 |
| RCC_APBRSTR1 | 185 |
| RCC_APBRSTR2 | 188 |
| RCC_APBSMENR1 | 198 |
| RCC_APBSMENR2 | 201 |
| RCC_BDCR | 205 |
| RCC_CCIPR | 202 |
| RCC_CFGR | 173 |
| RCC_CICR | 181 |
| RCC_CIER | 178 |
| RCC_CIFR | 180 |
| RCC_CR | 169 |
| RCC_CRRCR | 209 |

| | | | |
|--------------------|------|-----------------------|------|
| RCC_CSR | 207 | SYSCFG_ITLINE0 | 248 |
| RCC_DBGCFGR | 191 | SYSCFG_ITLINE1 | 248 |
| RCC_ICSCR | 172 | SYSCFG_ITLINE10 | 252 |
| RCC_IOPENR | 190 | SYSCFG_ITLINE11 | 253 |
| RCC_IOPRSTR | 184 | SYSCFG_ITLINE12 | 253 |
| RCC_IOPSMENR | 197 | SYSCFG_ITLINE13 | 254 |
| RCC_PLLCFG | 176 | SYSCFG_ITLINE14 | 254 |
| RNG_CR | 496 | SYSCFG_ITLINE15 | 255 |
| RNG_DR | 499 | SYSCFG_ITLINE16 | 255 |
| RNG_HTCR | 500 | SYSCFG_ITLINE17 | 255 |
| RNG_NSQR | 499 | SYSCFG_ITLINE18 | 256 |
| RNG_SR | 498 | SYSCFG_ITLINE19 | 256 |
| RTC_ALRABINR | 939 | SYSCFG_ITLINE2 | 249 |
| RTC_ALRBINR | 939 | SYSCFG_ITLINE20 | 256 |
| RTC_ALRMAR | 932 | SYSCFG_ITLINE21 | 257 |
| RTC_ALRMASSR | 933 | SYSCFG_ITLINE22 | 257 |
| RTC_ALRMBR | 934 | SYSCFG_ITLINE23 | 257 |
| RTC_ALRMBSSR | 935 | SYSCFG_ITLINE24 | 258 |
| RTC_CALR | 928 | SYSCFG_ITLINE25 | 258 |
| RTC_CR | 924 | SYSCFG_ITLINE26 | 258 |
| RTC_DR | 920 | SYSCFG_ITLINE27 | 259 |
| RTC_ICSR | 921 | SYSCFG_ITLINE28 | 259 |
| RTC_MISR | 937 | SYSCFG_ITLINE29 | 260 |
| RTC_PRER | 923 | SYSCFG_ITLINE3 | 249 |
| RTC_SCR | 938 | SYSCFG_ITLINE30 | 260 |
| RTC_SHIFTR | 929 | SYSCFG_ITLINE31 | 260 |
| RTC_SR | 936 | SYSCFG_ITLINE4 | 250 |
| RTC_SSR | 921 | SYSCFG_ITLINE5 | 250 |
| RTC_TR | 919 | SYSCFG_ITLINE6 | 250 |
| RTC_TSDR | 931 | SYSCFG_ITLINE7 | 251 |
| RTC_TSSSR | 931 | SYSCFG_ITLINE8 | 251 |
| RTC_TSTR | 930 | SYSCFG_ITLINE9 | 252 |
| RTC_WPR | 928 | SYSCFG_SCSR | 246 |
| RTC_WUTR | 924 | SYSCFG_SKR | 247 |
| SYSCFG_TSCCR | 247 | SYSCFG_CIDR0 | 1266 |
| SYSROM_CIDR1 | 1267 | SYSCFG_CIDR1 | 1267 |
| SYSROM_CIDR2 | 1267 | SYSCFG_CIDR3 | 1267 |
| SYSROM_CIDR3 | 1267 | SYSCFG_MEMTYPER | 1264 |
| SYSROM_PIDR0 | 1265 | SYSCROM_PIDR0 | 1265 |
| SYSROM_PIDR1 | 1265 | SYSCROM_PIDR1 | 1265 |
| SYSROM_PIDR2 | 1265 | SYSCROM_PIDR2 | 1265 |
| SYSROM_PIDR3 | 1266 | SYSCROM_PIDR3 | 1266 |
| SYSROM_PIDR4 | 1264 | SYSCROM_PIDR4 | 1264 |
| T | | | |
| TAMP_BKPxR | 960 | | |
| TAMP_CFGR | 955 | | |
| TAMP_CR1 | 949 | | |
| TAMP_CR2 | 951 | | |

| | | | |
|-------------------|---------|-------------------|--------------|
| TAMP_CR3 | 953 | TIM16_AF1 | 821 |
| TAMP_FLTCR | 954 | TIM16_ARR | 815 |
| TAMP_IER | 955 | TIM16_BDTR | 817 |
| TAMP_MISR | 957 | TIM16_CCER | 812 |
| TAMP_SCR | 959 | TIM16_CCMR1 | 809-810 |
| TAMP_SR | 956 | TIM16_CCR1 | 816 |
| TIM1_AF1 | 644 | TIM16_CNT | 814 |
| TIM1_AF2 | 645 | TIM16_CR1 | 804 |
| TIM1_ARR | 632 | TIM16_CR2 | 805 |
| TIM1_BDTR | 635 | TIM16_DCR | 820 |
| TIM1_CCER | 629 | TIM16_DIER | 806 |
| TIM1_CCMR1 | 622-623 | TIM16_DMAR | 820 |
| TIM1_CCMR2 | 626-627 | TIM16_EGR | 808 |
| TIM1_CCMR3 | 641 | TIM16_PSC | 815 |
| TIM1_CCR1 | 633 | TIM16_RCR | 816 |
| TIM1_CCR2 | 634 | TIM16_SR | 807 |
| TIM1_CCR3 | 634 | TIM16_TISEL | 822 |
| TIM1_CCR4 | 635 | TIM2_AF1 | 719 |
| TIM1_CCR5 | 642 | TIM2_OR1 | 718 |
| TIM1_CCR6 | 643 | TIM2_TISEL | 720 |
| TIM1_CNT | 632 | TIM3_AF1 | 720 |
| TIM1_CR1 | 611 | TIM3_OR1 | 719 |
| TIM1_CR2 | 612 | TIM3_TISEL | 721 |
| TIM1_DCR | 639 | TIMx_ARR | 715, 737 |
| TIM1_DIER | 617 | TIMx_CCER | 712 |
| TIM1_DMAR | 640 | TIMx_CCMR1 | 706, 708 |
| TIM1_EGR | 621 | TIMx_CCMR2 | 710-711 |
| TIM1_OR1 | 641 | TIMx_CCR1 | 715 |
| TIM1_PSC | 632 | TIMx_CCR2 | 716 |
| TIM1_RCR | 633 | TIMx_CCR3 | 716 |
| TIM1_SMCR | 615 | TIMx_CCR4 | 717 |
| TIM1_SR | 619 | TIMx_CNT | 713-714, 736 |
| TIM1_TISEL | 647 | TIMx_CR1 | 696, 733 |
| TIM15_AF1 | 800 | TIMx_CR2 | 697, 735 |
| TIM15_ARR | 794 | TIMx_DCR | 718 |
| TIM15_BDTR | 796 | TIMx_DIER | 702, 735 |
| TIM15_CCER | 791 | TIMx_DMAR | 718 |
| TIM15_CCMR1 | 787-788 | TIMx_EGR | 705, 736 |
| TIM15_CCR1 | 795 | TIMx_PSC | 714, 737 |
| TIM15_CCR2 | 796 | TIMx_SMCR | 699 |
| TIM15_CNT | 794 | TIMx_SR | 703, 736 |
| TIM15_CR1 | 779 | TSC_CR | 474 |
| TIM15_CR2 | 780 | TSC_ICR | 478 |
| TIM15_DCR | 799 | TSC_IER | 477 |
| TIM15_DIER | 783 | TSC_IOASCR | 479 |
| TIM15_DMAR | 799 | TSC_IOCCR | 480 |
| TIM15_EGR | 786 | TSC_IOGCSR | 481 |
| TIM15_PSC | 794 | TSC_IOGxCR | 481 |
| TIM15_RCR | 795 | TSC_IOHCR | 479 |
| TIM15_SMCR | 782 | TSC_IOSCR | 480 |
| TIM15_SR | 784 | TSC_ISR | 478 |
| TIM15_TISEL | 801 | | |

U

| | |
|-------------------------|------------|
| USART_BRR | 1081 |
| USART_CR1 | 1062, 1066 |
| USART_CR2 | 1069 |
| USART_CR3 | 1073, 1078 |
| USART_GTPR | 1082 |
| USART_ICR | 1096 |
| USART_ISR | 1085, 1091 |
| USART_PRESC | 1099 |
| USART_RDR | 1098 |
| USART_RQR | 1084 |
| USART_RTOR | 1083 |
| USART_TDR | 1098 |
| USB_BCDR | 1226 |
| USB_CHEP_RXTXBD_n | 1238-1239 |
| USB_CHEP_TXRXBD_n | 1236 |
| USB_CHEPnR | 1227 |
| USB_CNTR | 1217 |
| USB_DADDR | 1224 |
| USB_FNR | 1224 |
| USB_ISTR | 1220 |
| USB_LPMCSR | 1225 |

V

| | |
|-------------------|-----|
| VREFBUF_CCR | 411 |
| VREFBUF_CSR | 411 |

W

| | |
|----------------|-----|
| WWDG_CFR | 898 |
| WWDG_CR | 897 |
| WWDG_SR | 898 |

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.

© 2024 STMicroelectronics – All rights reserved

