



---

**ARM®-based 32-bit Cortex®-M4 MCU with 32 to 64 KB Flash,  
sLib, CAN, OTGFS, 13 timers, ADC, 12 communication interfaces**

# **HT32F49041 User Manual**

Revision: V1.00    Date: January 20, 2025

[\*\*www.holtek.com\*\*](http://www.holtek.com)

## Table of Contents

|   |           |
|---|-----------|
| <b>1 System architecture .....</b>          | <b>21</b> |
| Feature .....                               | 22        |
| System overview .....                       | 23        |
| ARM Cortex®-M4 processor .....              | 23        |
| Bit band .....                              | 24        |
| Interrupt and exception vectors .....       | 27        |
| System Tick (SysTick) .....                 | 28        |
| Reset .....                                 | 28        |
| List of abbreviations for registers .....   | 30        |
| Device characteristics information .....    | 30        |
| Flash memory size register .....            | 30        |
| Device electronic signature .....           | 30        |
| <b>2 Memory resources .....</b>             | <b>31</b> |
| Internal memory address map .....           | 31        |
| Flash memory .....                          | 31        |
| SRAM memory .....                           | 32        |
| Peripheral address map .....                | 33        |
| <b>3 Power control (PWC) .....</b>          | <b>36</b> |
| Introduction .....                          | 36        |
| Main Features .....                         | 37        |
| POR/LVR .....                               | 37        |
| Power voltage monitor (PVM) .....           | 37        |
| Power domain .....                          | 38        |
| Power saving modes .....                    | 38        |
| PWC registers .....                         | 40        |
| Power control register (PWC_CTRL) .....     | 40        |
| <b>4 Clock and reset manage (CRM) .....</b> | <b>43</b> |
| Clock .....                                 | 43        |
| Clock sources .....                         | 44        |
| System clock .....                          | 45        |
| Peripheral clock .....                      | 45        |
| Clock fail detector .....                   | 46        |
| Clock output .....                          | 46        |
| Interrupts .....                            | 46        |
| Reset .....                                 | 46        |
| System reset .....                          | 46        |
| Battery powered domain reset .....          | 47        |

|   |           |
|---|-----------|
| CRM registers .....   | 47        |
| Clock control register (CRM_CTRL) .....                         | 48        |
| Clock configuration register (CRM_CFG) .....                    | 49        |
| Clock interrupt register (CRM_CLKINT) .....                     | 51        |
| APB2 peripheral reset register (CRM_APB2RST) .....              | 53        |
| APB1 peripheral reset register (CRM_APB1RST) .....              | 53        |
| APB peripheral clock enable register (CRM_AHBEN) .....          | 55        |
| APB2 peripheral clock enable register (CRM_AHB2EN) .....        | 56        |
| APB1 peripheral clock enable register (CRM_AHB1EN) .....        | 57        |
| Battery powered domain control register (CRM_BPDC) .....        | 58        |
| Control/status register (CRM_CTRLSTS) .....                     | 59        |
| APB peripheral reset register (CRM_APB1RST) .....               | 60        |
| PLL configuration register (CRM_PLL) .....                      | 60        |
| Additional register 1 (CRM_MISC1) .....                         | 61        |
| OTG_FS extended control register (CRM_OTG_EXTCTRL) .....        | 62        |
| Additional register 2 (CRM_MISC2) .....                         | 62        |
| <b>5 Flash memory controller (FLASH) .....</b>                  | <b>63</b> |
| FLASH introduction .....  | 63        |
| Flash memory operation .....                                    | 65        |
| Unlock/lock .....   | 65        |
| Erase operation .....   | 65        |
| Programming operation .....                                     | 68        |
| Read operation .....  | 69        |
| Main Flash memory extension area .....                          | 70        |
| User system data area operation .....                           | 70        |
| Unlock/lock .....   | 70        |
| Erase operation .....   | 70        |
| Programming operation .....                                     | 71        |
| Read operation .....  | 72        |
| Flash memory protection .....                                   | 73        |
| Access protection .....   | 73        |
| Erase/program protection .....                                  | 73        |
| Read access .....   | 74        |
| Special functions .....   | 74        |
| Security library settings .....                                 | 74        |
| Bootloader used as Flash memory extension .....                 | 75        |
| CRC verify .....  | 76        |
| Flash memory registers .....                                    | 76        |
| Flash performance select register (FLASH_PSR) .....             | 77        |
| Flash unlock register (FLASH_UNLOCK) .....                      | 77        |
| Flash user system data unlock register (FLASH_USD_UNLOCK) ..... | 77        |
| Flash status register (FLASH_STS) .....                         | 78        |
| Flash control register (FLASH_CTRL) .....                       | 78        |

|  |           |
|--|-----------|
| Flash address register (FLASH_ADDR) .....                                    | 79        |
| User system data register (FLASH_USD) .....                                  | 79        |
| Erase/program protection status register (FLASH_EPPS) .....                  | 80        |
| Flash security library status register 0 (SLIB_STS0) .....                   | 80        |
| Flash security library status register 1 (SLIB_STS1) .....                   | 81        |
| Security library password clear register (SLIB_PWD_CLR) .....                | 81        |
| Security library additional status register (SLIB_MISC_STS) .....            | 82        |
| Flash CRC address register (FLASH_CRC_ADDR) .....                            | 82        |
| Flash CRC control register (FLASH_CRC_CTRL) .....                            | 82        |
| Flash CRC check result register (FLASH_CRC_CHKCR) .....                      | 82        |
| Security library password setting register (SLIB_SET_PWD) .....              | 83        |
| Security library address setting register (SLIB_SET_RANGE) .....             | 83        |
| Flash extension memory security library setting register (EM_SLIB_SET) ..... | 84        |
| Boot memory mode setting register (BTM_MODE_SET) .....                       | 84        |
| Security library unlock register (FLASH_UNLOCK) .....                        | 84        |
| <b>6 GPIOs and IOMUX .....</b>   | <b>85</b> |
| Introduction .....   | 85        |
| Function overview .....  | 85        |
| GPIO structure .....   | 85        |
| GPIO reset status .....  | 86        |
| General-purpose input configuration .....                                    | 86        |
| Analog mode configuration .....  | 86        |
| General-purpose output configuration .....                                   | 86        |
| GPIO port write protection .....   | 87        |
| IOMUX structure .....  | 87        |
| Multiplexed function configuration .....                                     | 88        |
| IOMUX input/output .....   | 88        |
| Peripheral MUX function configuration .....                                  | 91        |
| IOMUX mapping priority .....   | 91        |
| External interrupt/wake-up lines .....                                       | 91        |
| GPIO registers .....   | 92        |
| GPIO configuration register (GPIOx_CFGR) (x = A/B/C/D/F) .....               | 92        |
| GPIO output mode register (GPIOx_OMODER) (x = A/B/C/D/F) .....               | 92        |
| GPIO drive capability register (GPIOx_ODRVR) (x = A/B/C/D/F) .....           | 93        |
| GPIO pull-up/pull-down register (GPIOx_PULL) (x = A/B/C/D/F) .....           | 93        |
| GPIO input register (GPIOx_IDT) (x = A/B/C/D/F) .....                        | 93        |
| GPIO output register (GPIOx_ODT) (x = A/B/C/D/F) .....                       | 93        |
| GPIO set/clear register (GPIOx_SCR) (x = A/B/C/D/F) .....                    | 94        |
| GPIO write protection register (GPIOx_WPR) (x = A/B/C/D/F) .....             | 94        |
| GPIO multiplexed function low register (GPIOx_MUXL) (x = A/B/C/D/F) .....    | 94        |
| GPIO multiplexed function high register (GPIOx_MUXH) (x = A/B/C/D/F) .....   | 95        |
| GPIO port bit clear register (GPIOx_CLR) (x = A/B/C/D/F) .....               | 95        |
| GPIO huge current control register (GPIOx_HDRV) (x = A/B/C/D/F) .....        | 95        |

|  |            |
|--|------------|
| <b>7 System configuration controller (SCFG) .....</b>                    | <b>96</b>  |
| Introduction .....   | 96         |
| SCFG registers .....   | 96         |
| SCFG configuration register 1 (SCFG_CFG1) .....                          | 96         |
| SCFG external interrupt configuration register 1 (SCFG_EXINTC1).....     | 97         |
| SCFG external interrupt configuration register 2 (SCFG_EXINTC2).....     | 98         |
| SCFG external interrupt configuration register 3 (SCFG_EXINTC3).....     | 99         |
| SCFG external interrupt configuration register 4 (SCFG_EXINTC4).....     | 100        |
| SCFG configuration register 2 (SCFG_CFG2) .....                          | 100        |
| <b>8 External interrupt/Event controller (EXINT) .....</b>               | <b>101</b> |
| EXINT introduction.....  | 101        |
| Function overview and configuration procedure .....                      | 101        |
| EXINT registers .....  | 102        |
| Interrupt enable register (EXINT_INTEN).....                             | 102        |
| Event enable register (EXINT_EVTEN).....                                 | 103        |
| Polarity configuration register 1 (EXINT_POLCFG1) .....                  | 103        |
| Polarity configuration register 2 (EXINT_POLCFG2) .....                  | 103        |
| Software trigger register (EXINT_SWTRG).....                             | 103        |
| Interrupt status register (EXINT_INTSTS).....                            | 104        |
| <b>9 DMA controller (DMA).....</b>                                       | <b>105</b> |
| Introduction .....   | 105        |
| Main features .....  | 105        |
| Function overview .....  | 106        |
| DMA configuration .....  | 106        |
| Handshake mechanism .....  | 106        |
| Arbiter .....  | 107        |
| Programmable data transfer width.....                                    | 107        |
| Errors .....   | 108        |
| Interrupts.....  | 108        |
| Flexible DMA request mapping.....  | 109        |
| DMA registers .....  | 109        |
| DMA interrupt status register (DMA_STS).....                             | 110        |
| DMA interrupt flag clear register (DMA_CLR).....                         | 112        |
| DMA channel-x configuration register (DMA_CxCTRL) (x = 1...7).....       | 114        |
| DMA channel-x number of data register (DMA_CxDTCNT) (x = 1...7) .....    | 115        |
| DMA channel-x peripheral address register (DMA_CxPADDR) (x = 1...7)..... | 115        |
| DMA channel-x memory address register (DMA_CxMADDR) (x = 1...7).....     | 116        |
| DMA channel source register 0 (DMA_SRC_SEL0).....                        | 116        |
| DMA channel source register 1 (DMA_SRC_SEL1) .....                       | 116        |

|   |            |
|---|------------|
| <b>10 CRC calculation unit (CRC)</b>                                      | <b>117</b> |
| CRC introduction  | 117        |
| CRC functional description  | 118        |
| CRC registers   | 119        |
| Data register (CRC_DT)  | 119        |
| Common data register (CRC_CDT)  | 119        |
| Control register (CRC_CTRL)   | 119        |
| Initialization register (CRC_IDT)   | 120        |
| Polynomial register (CRC_POLY)  | 120        |
| <b>11 I<sup>2</sup>C interface</b>  | <b>121</b> |
| I <sup>2</sup> C introduction   | 121        |
| I <sup>2</sup> C main features  | 121        |
| I <sup>2</sup> C function overview  | 121        |
| I <sup>2</sup> C interface  | 122        |
| I <sup>2</sup> C timing control   | 124        |
| Data transfer management  | 126        |
| I <sup>2</sup> C master communication flow                                | 128        |
| I <sup>2</sup> C slave communication flow                                 | 133        |
| SMBus   | 137        |
| SMBus master communication flow   | 140        |
| SMBus slave communication flow  | 144        |
| Data transfer using DMA   | 149        |
| Error management  | 150        |
| I <sup>2</sup> C interrupt requests                                       | 151        |
| I <sup>2</sup> C debug mode   | 152        |
| I <sup>2</sup> C registers  | 152        |
| Control register 1 (I2C_CTRL1)  | 152        |
| Control register 2 (I2C_CTRL2)  | 154        |
| Own address register 1 (I2C_OADDR1)                                       | 154        |
| Own address register 2 (I2C_OADDR2)                                       | 155        |
| Timing register (I2C_CLKCTRL)   | 155        |
| Timeout register (I2C_TIMEOUT)  | 155        |
| Status register (I2C_STS)   | 156        |
| Status clear register (I2C_CLR)   | 158        |
| PEC register (I2C_PEC)  | 158        |
| Receive data register (I2C_RXDT)  | 158        |
| Transmit data register (I2C_TXDT)   | 158        |
| <b>12 Universal synchronous/asynchronous receiver/transmitter (USART)</b> | <b>159</b> |
| USART introduction  | 159        |
| Full-duplex/half-duplex selector  | 161        |

|  |            |
|--|------------|
| Mode selector .....                                | 161        |
| Introduction .....                                 | 161        |
| Configuration procedure .....                      | 161        |
| USART frame format and configuration .....         | 165        |
| DMA transfer introduction .....                    | 167        |
| Transmission using DMA .....                       | 167        |
| Reception using DMA .....                          | 168        |
| Baud rate generation .....                         | 168        |
| Introduction .....                                 | 168        |
| Configuration .....                                | 168        |
| Transmitter .....                                  | 169        |
| Transmitter introduction .....                     | 169        |
| Transmitter configuration .....                    | 169        |
| Receiver .....                                     | 170        |
| Receiver introduction .....                        | 170        |
| Receiver configuration .....                       | 171        |
| Start bit and noise detection .....                | 172        |
| Tx/Rx swap .....                                   | 173        |
| Interrupt requests .....                           | 174        |
| I/O pin control .....                              | 175        |
| USART registers .....                              | 175        |
| Status register (USART_STS) .....                  | 176        |
| Data register (USART_DT) .....                     | 177        |
| Baud rate register (USART_BAUDR) .....             | 177        |
| Control register 1 (USART_CTRL1) .....             | 177        |
| Control register 2 (USART_CTRL2) .....             | 179        |
| Control register 3 (USART_CTRL3) .....             | 180        |
| Guard time and divider register (USART_GDIV) ..... | 182        |
| <b>13 Serial peripheral interface (SPI) .....</b>  | <b>183</b> |
| SPI introduction .....                             | 183        |
| Function overview .....                            | 183        |
| SPI description .....                              | 183        |
| Full-duplex/half-duplex selector .....             | 185        |
| Chip select controller .....                       | 188        |
| SPI_SCK controller .....                           | 188        |
| CRC .....  | 189        |
| DMA transfer .....                                 | 190        |
| TI mode .....                                      | 191        |
| Transmitter .....                                  | 191        |
| Receiver .....                                     | 192        |
| Motorola mode .....                                | 192        |
| TI mode .....                                      | 196        |

|   |            |
|---|------------|
| Interrupts.....   | 197        |
| IO pin control .....  | 197        |
| Precautions.....  | 197        |
| I <sup>2</sup> S functional description .....                               | 197        |
| I <sup>2</sup> S introduction .....   | 197        |
| I <sup>2</sup> S full-duplex .....  | 199        |
| Operating mode selector .....   | 199        |
| Audio protocol selector .....   | 201        |
| I2S_CLK controller.....   | 203        |
| DMA transfer.....   | 204        |
| Transmitter/Receiver .....  | 205        |
| I <sup>2</sup> S communication timings.....                                 | 206        |
| Interrupts .....  | 206        |
| IO pin control .....  | 207        |
| SPI registers .....   | 207        |
| SPI control register 1 (SPI_CTRL1) (not used in I <sup>2</sup> S mode)..... | 207        |
| SPI control register 2 (SPI_CTRL2) .....                                    | 209        |
| SPI status register (SPI_STS).....  | 210        |
| SPI data register (SPI_DT).....   | 210        |
| SPICRC register (SPI_CPOLY) (Not used in I <sup>2</sup> S mode).....        | 211        |
| SPIRxCRC register (SPI_RCRC) (not used in I <sup>2</sup> S mode).....       | 211        |
| SPITxCRC register (SPI_TCRC) .....  | 211        |
| SPI_I2S register (SPI_I2SCTRL) .....  | 211        |
| SPI_I2S prescaler register (SPI_I2SCLKP) .....                              | 212        |
| <b>14 Timer .....</b>   | <b>213</b> |
| Basic timer (TMR6 and TMR7) .....   | 214        |
| TMR6 and TMR7 introduction.....   | 214        |
| TMR6 and TMR7 main features .....   | 214        |
| TMR6 and TMR7 function overview .....                                       | 214        |
| TMR6 and TMR7 registers .....   | 216        |
| General-purpose timer (TMR2 and TMR3).....                                  | 219        |
| TMR2 and TMR3 introduction.....   | 219        |
| TMR2 and TMR3 main features .....   | 219        |
| TMR2 and TMR3 function overview .....                                       | 219        |
| TMR2 and TMR3 registers .....   | 238        |
| General-purpose timer (TMR9 to TMR14).....                                  | 251        |
| TMR13 and TMR14 introduction.....   | 251        |
| TMR13 and TMR14 main features .....   | 251        |
| TMR13 and TMR14 function overview .....                                     | 251        |
| TMR13 and TMR14 registers .....   | 258        |
| General-purpose timer (TMR15).....  | 263        |
| TMR15 introduction .....  | 263        |
| TMR15 main features .....   | 263        |



|   |            |
|---|------------|
| TMR15 function overview .....                   | 263        |
| TMR15 registers .....                           | 278        |
| General-purpose timers (TMR16 and TMR17) .....  | 291        |
| TMR16 and TMR17 introduction .....              | 291        |
| TMR16 and TMR17 main features .....             | 291        |
| TMR16 and TMR17 function overview .....         | 291        |
| TMR16 and TMR17 registers .....                 | 301        |
| Advanced-control timers (TMR1) .....            | 311        |
| TMR1 introduction .....                         | 311        |
| TMR1 main features .....                        | 311        |
| TMR1 function overview .....                    | 312        |
| TMR1 registers .....                            | 332        |
| <b>15 Window watchdog timer (WWDT) .....</b>    | <b>349</b> |
| WWDT introduction .....                         | 349        |
| WWDT main features .....                        | 349        |
| WWDT function overview .....                    | 349        |
| Debug mode .....                                | 350        |
| WWDT registers .....                            | 350        |
| Control register (WWDT_CTRL) .....              | 351        |
| Configuration register (WWDT_CFG) .....         | 351        |
| Status register (WWDT_STS) .....                | 351        |
| <b>16 Watchdog timer (WDT) .....</b>            | <b>352</b> |
| WDT introduction .....                          | 352        |
| WDT main features .....                         | 352        |
| WDT function overview .....                     | 352        |
| Debug mode .....                                | 353        |
| WDT registers .....                             | 354        |
| Command register (WDT_CMD) .....                | 354        |
| Divider register (WDT_DIV) .....                | 354        |
| Reload register (WDT_RLD) .....                 | 354        |
| Status register (WDT_STS) .....                 | 355        |
| Window register (WDT_WIN) .....                 | 355        |
| <b>17 Enhanced real-time clock (ERTC) .....</b> | <b>356</b> |
| ERTC introduction .....                         | 356        |
| ERTC main features .....                        | 356        |
| ERTC function overview .....                    | 357        |
| ERTC clock .....                                | 357        |
| ERTC initialization .....                       | 358        |
| Periodic automatic wakeup .....                 | 360        |
| ERTC calibration .....                          | 360        |
| Time stamp function .....                       | 361        |

|  |                |
|--|----------------|
| Tamper detection .....   | 361            |
| Multiplexed function output.....   | 362            |
| ERTC wakeup.....   | 362            |
| <b>ERTC registers .....</b>  | <b>363</b>     |
| ERTC time register (ERTC_TIME).....  | 364            |
| ERTC date register (ERTC_DATE).....  | 364            |
| ERTC control register (ERTC_CTRL).....                                     | 365            |
| ERTC initialization and status register (ERTC_STS).....                    | 366            |
| ERTC divider register (ERTC_DIV) .....                                     | 368            |
| ERTC wakeup timer register (ERTC_WAT).....                                 | 368            |
| ERTC alarm clock A register (ERTC_ALA).....                                | 368            |
| ERTC write protection register (ERTC_WP).....                              | 369            |
| ERTC subsecond register (ERTC_SBS) .....                                   | 369            |
| ERTC time adjustment register (ERTC_TADJ).....                             | 369            |
| ERTC time stamp time register (ERTC_TSTM).....                             | 369            |
| ERTC time stamp date register (ERTC_TSDT).....                             | 370            |
| ERTC time stamp subsecond register (ERTC_TSSBS) .....                      | 370            |
| ERTC smooth calibration register (ERTC_SCAL) .....                         | 370            |
| ERTC tamper configuration register (ERTC_TAMP) .....                       | 370            |
| ERTC alarm clock A subsecond register (ERTC_ALASBS) .....                  | 371            |
| ERTC battery powered domain data register (ERTC_BPRx) .....                | 372            |
| <br><b>18 Analog-to-digital converter (ADC).....</b>                       | <br><b>373</b> |
| ADC introduction.....  | 373            |
| ADC main features .....  | 373            |
| ADC structure .....  | 374            |
| ADC function overview .....  | 375            |
| Channel management .....   | 375            |
| ADC operation process.....   | 375            |
| Conversion sequence management.....  | 378            |
| Oversampling.....  | 380            |
| Data management .....  | 383            |
| Voltage monitoring .....   | 384            |
| Status flag and interrupts.....  | 384            |
| ADC registers .....  | 385            |
| ADC status register (ADC_STS).....   | 385            |
| ADC control register1 (ADC_CTRL1).....                                     | 386            |
| ADC control register2 (ADC_CTRL2).....                                     | 387            |
| ADC sampling time register 1 (ADC_SPT1).....                               | 389            |
| ADC sampling time register 2 (ADC_SPT2).....                               | 390            |
| ADC preempted channel data offset register x (ADC_PCDTOx) (x = 1..4) ..... | 392            |
| ADC voltage monitor high threshold register (ADC_VWHB) .....               | 392            |
| ADC voltage monitor low threshold register (ADC_VVLB).....                 | 392            |
| ADC ordinary sequence register 1 (ADC_OSQ1).....                           | 393            |

|  |            |
|--|------------|
| ADC ordinary sequence register 2 (ADC_ OSQ2).....                        | 393        |
| ADC ordinary sequence register 3 (ADC_ OSQ3).....                        | 393        |
| ADC preempted sequence register (ADC_ PSQ).....                          | 394        |
| ADC preempted data register x (ADC_ PDTx) (x = 1..4).....                | 394        |
| ADC ordinary data register (ADC_ ODT) .....                              | 394        |
| ADC oversampling register (ADC_ OVSP) .....                              | 395        |
| <b>19 Controller area network (CAN) .....</b>                            | <b>396</b> |
| CAN introduction.....  | 396        |
| CAN main features .....  | 396        |
| Baud rate .....  | 396        |
| Interrupt management .....   | 399        |
| Design tips .....  | 400        |
| Function overview .....  | 400        |
| General description.....   | 400        |
| Operating modes .....  | 401        |
| Test modes .....   | 402        |
| Message filtering.....   | 402        |
| Message transmission.....  | 405        |
| Message reception .....  | 406        |
| Error management.....  | 407        |
| CAN registers .....  | 408        |
| CAN control and status registers.....                                    | 409        |
| CAN mailbox registers.....   | 420        |
| CAN filter registers.....  | 423        |
| <b>20 Universal serial bus full-speed device interface (OTGFS) .....</b> | <b>425</b> |
| OTGFS structure .....  | 425        |
| OTGFS functional description.....  | 426        |
| OTGFS clock and pin configuration.....                                   | 426        |
| OTGFS clock configuration.....   | 426        |
| OTGFS pin configuration .....  | 427        |
| OTGFS interrupts .....   | 427        |
| OTGFS functional description.....  | 428        |
| OTGFS initialization.....  | 428        |
| OTGFS FIFO configuration.....  | 429        |
| OTGFS host mode.....   | 432        |
| OTGFS device mode .....  | 449        |
| OTGFS control and status registers .....                                 | 469        |
| CSR register map .....   | 469        |
| OTGFS register address map.....  | 470        |
| OTGFS global registers.....  | 474        |
| Host-mode registers .....  | 489        |

|  |            |
|--|------------|
| Device-mode registers.....                       | 497        |
| Power and clock control registers.....           | 514        |
| <b>21 HICK auto clock calibration (ACC).....</b> | <b>515</b> |
| ACC introduction.....                            | 515        |
| Main features .....                              | 515        |
| Interrupt requests.....                          | 515        |
| Functional description .....                     | 516        |
| Principle .....                                  | 517        |
| Register description .....                       | 519        |
| ACC register map .....                           | 519        |
| Status register (ACC_STS).....                   | 519        |
| Control register 1 (ACC_CTRL1).....              | 520        |
| Control register 2 (ACC_CTRL2).....              | 521        |
| Compare value 1 (ACC_C1).....                    | 521        |
| Compare value 2 (ACC_C2).....                    | 521        |
| Compare value 3 (ACC_C3).....                    | 522        |
| <b>22 Infrared timer (IRTMR).....</b>            | <b>523</b> |
| <b>23 Debug (DEBUG) .....</b>                    | <b>524</b> |
| Debug introduction.....                          | 524        |
| Debug and Trace .....                            | 524        |
| I/O pin control .....                            | 524        |
| DEGUB registers .....                            | 524        |
| DEBUG device ID (DEBUG_IDCODE).....              | 525        |
| DEBUG control register (DEBUG_CTRL).....         | 525        |
| DEBUG SERIES ID register (DEBUG_SER_ID).....     | 527        |

## List of Tables

|  |     |
|--|-----|
| Table 1. Bit-band address mapping in SRAM .....                                    | 26  |
| Table 2. Bit-band address mapping in the peripheral area .....                     | 26  |
| Table 3. HT32F49041 series vector table .....                                      | 27  |
| Table 4. List of abbreviations for registers .....                                 | 30  |
| Table 5. List of abbreviations for registers .....                                 | 30  |
| Table 6. Flash memory organization (64 KB).....                                    | 32  |
| Table 7. Flash memory organization (64 KB).....                                    | 32  |
| Table 8. Peripheral boundary address .....   | 33  |
| Table 9. PWC register map and reset values .....                                   | 40  |
| Table 10. CRM register map and reset values.....                                   | 47  |
| Table 11. Flash memory architecture (64 K) .....                                   | 63  |
| Table 12. Flash memory architecture (32 K).....                                    | 63  |
| Table 13. User system data area .....  | 64  |
| Table 14. Flash memory access limit.....   | 73  |
| Table 15. Flash memory register map and reset value .....                          | 76  |
| Table 16. Port A multiplexed function configuration with GPIOA_MUX* register ..... | 88  |
| Table 17. Port B multiplexed function configuration with GPIOB_MUX* register.....  | 89  |
| Table 18. Port C multiplexed function configuration with GPIOC_MUX* register ..... | 90  |
| Table 19. Port D multiplexed function configuration with GPIOD_MUX* register ..... | 90  |
| Table 20. Port F multiplexed function configuration with GPIOE_MUX* register ..... | 90  |
| Table 21. Pins owned by hardware .....   | 91  |
| Table 22. GPIO register map and reset values .....                                 | 92  |
| Table 23. SCFG register map and reset values .....                                 | 96  |
| Table 24. External interrupt/event controller register map and reset value.....    | 102 |
| Table 25. DMA error event .....  | 108 |
| Table 26. DMA interrupt requests .....   | 108 |
| Table 27. DMA flexible request sources.....  | 109 |
| Table 28. DMA register map and reset value.....                                    | 109 |
| Table 29. CRC register map and reset value .....                                   | 119 |
| Table 30. I <sup>2</sup> C timing specifications .....                             | 126 |
| Table 31. I <sup>2</sup> C configuration table .....                               | 128 |
| Table 32. SMBus timeout specification .....  | 139 |
| Table 33. SMBus timeout detection configuration.....                               | 139 |
| Table 34. SMBus mode configuration .....   | 139 |
| Table 35. I <sup>2</sup> C error events .....                                      | 150 |
| Table 36. I <sup>2</sup> C interrupt requests .....                                | 151 |
| Table 37. I <sup>2</sup> C register map and reset values .....                     | 152 |
| Table 38. Error calculation for programmed baud rate .....                         | 168 |
| Table 39. Data sampling over start bit and noise detection .....                   | 172 |
| Table 40. Data sampling over valid data and noise detection .....                  | 172 |
| Table 41. Maximum allowable deviation .....  | 173 |
| Table 42. USART interrupt request .....  | 174 |
| Table 43. USART register map and reset value.....                                  | 175 |

|   |     |
|---|-----|
| Table 44. Audio frequency precision using system clock.....   | 203 |
| Table 45. SPI register map and reset value.....   | 207 |
| Table 46. TMR functional comparison.....  | 213 |
| Table 47. TMR6 and TMR7 register table and reset value .....  | 216 |
| Table 48. TMRx internal trigger connection .....  | 223 |
| Table 49. Counting direction versus encoder signals .....   | 227 |
| Table 50. TMR2 and TMR3 register map and reset value .....  | 238 |
| Table 51. Standard CxOUT channel output control bit.....  | 248 |
| Table 52. TMR13 and TMR14 register map and reset value .....  | 258 |
| Table 53. Standard CxOUT channel output control bit.....  | 262 |
| Table 54. TMRx internal trigger connection .....  | 266 |
| Table 55. TMR15 register map and reset value .....  | 278 |
| Table 56. Complementary output channel CxOUT and CxCOUT control bits with brake function .....      | 287 |
| Table 57. TMR16 and TMR17 register map and reset value .....  | 301 |
| Table 58. Complementary output channel CxOUT and CxCOUT control bits with brake function .....      | 307 |
| Table 59. TMRx internal trigger connection .....  | 315 |
| Table 60. Counting direction versus encoder signals .....   | 321 |
| Table 61. TMR1 register map and reset value.....  | 332 |
| Table 62. Complementary output channel CxOUT and CxCOUT control bits with brake function .....      | 344 |
| Table 63. Minimum and maximum timeout value when PCLK1 = 72 MHz .....                               | 350 |
| Table 64. WWDTC register map and reset value .....  | 350 |
| Table 65. WDT timeout period (LICK = 40 kHz).....   | 353 |
| Table 66. WDT register and reset value.....   | 354 |
| Table 67. RTC register map and reset values.....  | 358 |
| Table 68. ERTC low-power mode wakeup.....   | 363 |
| Table 69. Interrupt control bits .....  | 363 |
| Table 70. ERTC register map and reset values .....  | 363 |
| Table 71. Trigger sources for ADC.....  | 377 |
| Table 72. Correlation between maximum cumulative data, oversampling multiple and shift digits ..... | 381 |
| Table 73. ADC register map and reset values .....   | 385 |
| Table 74. CAN register map and reset values .....   | 408 |
| Table 75. OTGFS input/output pins .....   | 427 |
| Table 76. OTGFS transmit FIFO SRAM allocation .....   | 430 |
| Table 77. OTGFS internal storage space allocation .....   | 431 |
| Table 78. OTGFS register map and reset values .....   | 470 |
| Table 79. Minimum duration for software disconnect.....   | 499 |
| Table 80. ACC interrupt requests.....   | 515 |
| Table 81. ACC register map and reset values .....   | 519 |
| Table 82. DEBUG register address and reset value .....  | 524 |

## List of Figures

|  |     |
|--|-----|
| Figure 1. HT32F49041 Series microcontrollers system architecture .....           | 21  |
| Figure 2. Internal block diagram of Cortex®-M4 .....                             | 24  |
| Figure 3. Comparison between bit-band region and its alias region: image A.....  | 24  |
| Figure 4. Comparison between bit-band region and its alias region: image B ..... | 25  |
| Figure 5. Reset process.....   | 28  |
| Figure 6. Example of MSP and PC initialization .....                             | 29  |
| Figure 7. HT32F49041 address mapping .....                                       | 31  |
| Figure 8. Block diagram of each power supply .....                               | 36  |
| Figure 9. Power-on reset/Low voltage reset waveform .....                        | 37  |
| Figure 10. PVM threshold and output .....  | 38  |
| Figure 11. HT32F49041 clock tree.....  | 43  |
| Figure 12. System reset circuit .....  | 47  |
| Figure 13. Flash memory sector erase process .....                               | 66  |
| Figure 14. Flash memory mass erase process.....                                  | 67  |
| Figure 15. Flash memory programming process .....                                | 69  |
| Figure 16. System data area erase process.....                                   | 71  |
| Figure 17. System data area programming process.....                             | 72  |
| Figure 18. GPIO basic structure .....  | 85  |
| Figure 19. IOMUX structure.....  | 87  |
| Figure 20. External interrupt/Event controller block diagram.....                | 101 |
| Figure 21. DMA block diagram .....   | 105 |
| Figure 22. Re-arbitrate after request/acknowledge .....                          | 107 |
| Figure 23. PWIDTH: byte, MWIDTH: half-word .....                                 | 107 |
| Figure 24. PWIDTH: half-word, MWIDTH: word .....                                 | 108 |
| Figure 25. PWIDTH: word, MWIDTH: byte .....                                      | 108 |
| Figure 26. CRC calculation unit block diagram.....                               | 117 |
| Figure 27. Diagram of byte reverse .....   | 118 |
| Figure 28. I <sup>2</sup> C bus protocol .....                                   | 122 |
| Figure 29. I <sup>2</sup> C function block diagram.....                          | 122 |
| Figure 30. Setup and hold time.....  | 125 |
| Figure 31. I <sup>2</sup> C master transmission flow .....                       | 130 |
| Figure 32. Transfer sequence of I <sup>2</sup> C master transmitter .....        | 131 |
| Figure 33. I <sup>2</sup> C master receive flow .....                            | 131 |
| Figure 34. Transfer sequence of I <sup>2</sup> C master receiver .....           | 132 |
| Figure 35. 10-bit address read access when READH10 = 1 .....                     | 132 |
| Figure 36. 10-bit address read access when READH10 = 0 .....                     | 133 |
| Figure 37. I <sup>2</sup> C slave transmission flow .....                        | 135 |
| Figure 38. I <sup>2</sup> C slave transmission timing .....                      | 136 |
| Figure 39. I <sup>2</sup> C slave receive flow .....                             | 136 |
| Figure 40. I <sup>2</sup> C slave receive timing .....                           | 137 |
| Figure 41. SMBus master transmission flow .....                                  | 142 |
| Figure 42. SMBus master transmission timing .....                                | 143 |
| Figure 43. SMBus master receive flow .....                                       | 143 |

|   |     |
|---|-----|
| Figure 44. SMBus master receive timing .....                                | 144 |
| Figure 45. SMBus slave transmission flow .....                              | 146 |
| Figure 46. SMBus slave transmission timing .....                            | 147 |
| Figure 47. SMBus slave receive flow .....                                   | 148 |
| Figure 48. SMBus slave receive timing .....                                 | 149 |
| Figure 49. USART block diagram .....  | 159 |
| Figure 50. BFF and FERR detection in LIN mode .....                         | 162 |
| Figure 51. Smartcard frame format .....                                     | 162 |
| Figure 52. IrDA DATA(3/16) – normal mode .....                              | 163 |
| Figure 53. Hardware flow control .....                                      | 163 |
| Figure 54. Mute mode using Idle line or Address mark detection .....        | 164 |
| Figure 55. 8-bit format USART synchronous mode .....                        | 165 |
| Figure 56. Word length .....  | 166 |
| Figure 57. Stop bit configuration .....                                     | 167 |
| Figure 58. TDC/TDBE behavior when transmitting .....                        | 170 |
| Figure 59. Data sampling for noise detection .....                          | 173 |
| Figure 60. Tx/Rx swap .....   | 174 |
| Figure 61. USART interrupt map diagram .....                                | 175 |
| Figure 62. SPI block diagram .....  | 183 |
| Figure 63. Data clock timing diagram .....                                  | 185 |
| Figure 64. SPI two-wire unidirectional full-duplex connection .....         | 186 |
| Figure 65. Single-wire unidirectional receive only in SPI master mode ..... | 186 |
| Figure 66. Single-wire unidirectional receive only in SPI slave mode .....  | 187 |
| Figure 67. Single-wire bidirectional half-duplex mode .....                 | 187 |
| Figure 68. Master full-duplex communications .....                          | 193 |
| Figure 69. Slave full-duplex communications .....                           | 193 |
| Figure 70. Master half-duplex transmit .....                                | 194 |
| Figure 71. Slave half-duplex receive .....                                  | 194 |
| Figure 72. Slave half-duplex transmit .....                                 | 195 |
| Figure 73. Master half-duplex receive .....                                 | 195 |
| Figure 74. TI mode continous transfer .....                                 | 196 |
| Figure 75. TI mode continous transfer with dummy CLK .....                  | 196 |
| Figure 76. TI mode continous transfer with dummy CLK .....                  | 196 |
| Figure 77. SPI interrupts .....   | 197 |
| Figure 78. I²S block diagram .....  | 198 |
| Figure 79. I²S full-duplex structure .....                                  | 199 |
| Figure 80. I²S slave device transmission .....                              | 200 |
| Figure 81. I²S slave device reception .....                                 | 200 |
| Figure 82. I²S master device transmission .....                             | 201 |
| Figure 83. I²S master device reception .....                                | 201 |
| Figure 84. CK & MCK source in master mode .....                             | 203 |
| Figure 85. Audio standard timings .....                                     | 206 |
| Figure 86. I²S interrupts .....   | 206 |
| Figure 87. Basic timer block diagram .....                                  | 214 |



|  |     |
|--|-----|
| Figure 88. Control circuit with CK_INT divided by 1 .....                                    | 214 |
| Figure 89. Basic structure of a counter .....  | 215 |
| Figure 90. Overflow event when PRBEN = 0.....  | 215 |
| Figure 91. Overflow event when PRBEN = 1.....  | 216 |
| Figure 92. Counting timing diagram when the prescaler division is 4.....                     | 216 |
| Figure 93. General-purpose timer block diagram .....   | 219 |
| Figure 94. Counting clock .....  | 220 |
| Figure 95. Control circuit with CK_INT, TMRx_DIV = 0x0 and TMRx_PR = 0x16 .....              | 220 |
| Figure 96. Block diagram of external clock mode A .....                                      | 221 |
| Figure 97. Counting in external clock mode A, PR = 0x32 and DIV = 0x0 .....                  | 222 |
| Figure 98. Block diagram of external clock mode B .....                                      | 222 |
| Figure 99. Counting in external clock mode B, PR = 0x32, DIV = 0x0 .....                     | 222 |
| Figure 100. Counter timing with prescaler value changing from 1 to 4 .....                   | 223 |
| Figure 101. Basic structure of a counter.....  | 224 |
| Figure 102. Overflow event when PRBEN = 0 .....  | 224 |
| Figure 103. Overflow event when PRBEN = 1 .....  | 224 |
| Figure 104. Counter timing diagram with internal clock divided by 4.....                     | 225 |
| Figure 105. Counter timing diagram with internal clock divided by 1 and TMRx_PR = 0x32 ..... | 225 |
| Figure 106. Encoder mode structure .....   | 226 |
| Figure 107. Example of counter behavior in encoder interface mode (encoder mode C) .....     | 227 |
| Figure 108. Input/output channel 1 main circuit.....   | 228 |
| Figure 109. Channel 1 input stage.....   | 228 |
| Figure 110. PWM input mode configuration example .....                                       | 229 |
| Figure 111. PWM input mode .....   | 230 |
| Figure 112. Capture/compare channel output stage (channel 1 to 4).....                       | 230 |
| Figure 113. C1ORAW toggles when counter value matches the C1DT value.....                    | 232 |
| Figure 114. Upcounting mode and PWM mode A.....  | 232 |
| Figure 115. Up/down counting mode and PWM mode A .....                                       | 233 |
| Figure 116. One-pulse mode .....   | 233 |
| Figure 117. Clearing CxORAW (PWM mode A) by EXT input.....                                   | 234 |
| Figure 118. Example of reset mode.....   | 235 |
| Figure 119. Example of suspend mode .....  | 235 |
| Figure 120. Example of trigger mode .....  | 236 |
| Figure 121. Master/slave timer connection.....   | 236 |
| Figure 122. Using master timer to start slave timer .....                                    | 237 |
| Figure 123. Starting master and slave timers synchronously by an external trigger .....      | 238 |
| Figure 124. Block diagram of general-purpose TMR13/14.....                                   | 251 |
| Figure 125. Counting clock .....   | 251 |
| Figure 126. Control circuit with CK_INT, TMRx_DIV = 0x0 and TMRx_PR = 0x16 .....             | 252 |
| Figure 127. Basic structure of a counter.....  | 253 |
| Figure 128. Overflow event when PRBEN = 0 .....  | 253 |
| Figure 129. Overflow event when PRBEN = 1.....   | 253 |
| Figure 130. Input/output channel 1 main circuit.....   | 254 |
| Figure 131. Channel 1 input stage.....   | 254 |

|   |     |
|---|-----|
| Figure 132. Capture/compare channel output stage .....                                  | 255 |
| Figure 133. C1ORAW toggles when counter value matches the C1DT value.....               | 256 |
| Figure 134. Upcounting mode and PWM mode A.....   | 257 |
| Figure 135. One-pulse mode .....  | 257 |
| Figure 136. TMR15 block diagram.....  | 263 |
| Figure 137. Basic structure of a counter .....  | 264 |
| Figure 138. Control circuit with CK_INT divided by 1, TMRx_DIV = 0x0 and PR = 0x16..... | 264 |
| Figure 139. Block diagram of external clock mode A .....                                | 265 |
| Figure 140. Counting in external clock mode A, PR = 0x32, DIV = 0x0 .....               | 265 |
| Figure 141. Counter timing with prescaler value changing from 1 to 4.....               | 266 |
| Figure 142. Basic structure of a counter.....   | 267 |
| Figure 143. Overflow event when PRBEN = 0.....  | 267 |
| Figure 144. Overflow event when PRBEN = 1.....  | 267 |
| Figure 145. OVFIIF when RPR = 2.....  | 268 |
| Figure 146. Input/output channel 1 main circuit.....                                    | 269 |
| Figure 147. Channel 1 input stage.....  | 269 |
| Figure 148. PWM input mode configuration example .....                                  | 270 |
| Figure 149. PWM input mode .....  | 271 |
| Figure 150. Channel 1 output stage .....  | 271 |
| Figure 151. Channel 2 output stage .....  | 271 |
| Figure 152. C1ORAW toggles when counter value matches the C1DT value.....               | 273 |
| Figure 153. Upcounting mode and PWM mode A.....   | 273 |
| Figure 154. One-pulse mode .....  | 274 |
| Figure 155. Complementary output with dead-time insertion .....                         | 275 |
| Figure 156. Example of TMR output control .....   | 276 |
| Figure 157. Example of TMR brake function .....   | 276 |
| Figure 158. Example of reset mode.....  | 277 |
| Figure 159. Example of suspend mode .....   | 277 |
| Figure 160. Example of trigger mode .....   | 278 |
| Figure 161. Block diagram of TMR16 and TMR17 timer .....                                | 291 |
| Figure 162. Counting clock .....  | 291 |
| Figure 163. Control circuit with CK_INT, TMRx_DIV = 0x0 and TMRx_PR = 0x16.....         | 292 |
| Figure 164. Basic structure of a counter.....   | 293 |
| Figure 165. Overflow event when PRBEN = 0.....  | 293 |
| Figure 166. Overflow event when PRBEN = 1.....  | 293 |
| Figure 167. OVFIIF when RPR = 2.....  | 294 |
| Figure 168. Input/output channel 1 main circuit.....                                    | 294 |
| Figure 169. Channel 1 input stage.....  | 295 |
| Figure 170. Channel output stage .....  | 295 |
| Figure 171. C1ORAW toggles when counter value matches the C1DT value.....               | 297 |
| Figure 172. Upcounting mode and PWM mode A.....   | 297 |
| Figure 173. One-pulse mode .....  | 298 |
| Figure 174. Complementary output with dead-time insertion .....                         | 299 |
| Figure 175. Example of TMR output control .....   | 300 |

|  |     |
|--|-----|
| Figure 176. Example of TMR brake function .....  | 300 |
| Figure 177. Block diagram of advanced-control timer .....                                    | 312 |
| Figure 178. Counting clock .....   | 312 |
| Figure 179. Control circuit with CK_INT, TMRx_DIV = 0x0 and TMRx_PR = 0x16 .....             | 313 |
| Figure 180. Block diagram of external clock mode A.....                                      | 314 |
| Figure 181. Counting in external clock mode A, PR = 0x32, DIV = 0x0 .....                    | 314 |
| Figure 182. Block diagram of external clock mode B .....                                     | 314 |
| Figure 183. Counting in external clock mode B, PR = 0x32, DIV = 0x0 .....                    | 315 |
| Figure 184. Counter timing with prescaler value changing from 1 to 4.....                    | 316 |
| Figure 185. Basic structure of a counter .....   | 316 |
| Figure 186. Overflow event when PRBEN = 0.....   | 317 |
| Figure 187. Overflow event when PRBEN = 1.....   | 317 |
| Figure 188. Counter timing diagram with internal clock divided by 4.....                     | 317 |
| Figure 189. Counter timing diagram with internal clock divided by 1 and TMRx_PR = 0x32 ..... | 318 |
| Figure 190. OVFIIF when RPR = 2.....   | 319 |
| Figure 191. Structure of encoder mode .....  | 320 |
| Figure 192. Example of encoder interface mode C .....  | 321 |
| Figure 193. Input/output channel 1 main circuit.....   | 322 |
| Figure 194. Channel 1 input stage.....   | 322 |
| Figure 195. PWM input mode configuration example.....  | 323 |
| Figure 196. PWM input mode .....   | 324 |
| Figure 197. Channel output stage (channel 1 to 3) .....                                      | 324 |
| Figure 198. Channel 4 output stage .....   | 324 |
| Figure 199. C1ORAW toggles when counter value matches the C1DT value.....                    | 326 |
| Figure 200. Upcounting mode and PWM mode A.....  | 326 |
| Figure 201. Up/down counting mode and PWM mode .....   | 327 |
| Figure 202. One-pulse mode .....   | 327 |
| Figure 203. Clearing CxORAW (PWM mode A) by EXT input.....                                   | 328 |
| Figure 204. Complementary output with dead-time insertion .....                              | 329 |
| Figure 205. Example of TMR output control .....  | 330 |
| Figure 206. Example of TMR brake function .....  | 330 |
| Figure 207. Example of reset mode.....   | 331 |
| Figure 208. Example of suspend mode .....  | 331 |
| Figure 209. Example of trigger mode .....  | 332 |
| Figure 210. Window watchdog block diagram .....  | 349 |
| Figure 211. Window watchdog timing diagram .....   | 350 |
| Figure 212. WDT block diagram .....  | 353 |
| Figure 213. ERTC block diagram .....   | 357 |
| Figure 214. ADC1 block diagram.....  | 374 |
| Figure 215. ADC basic operation process .....  | 376 |
| Figure 216. ADC power-on and calibration.....  | 377 |
| Figure 217. Sequence mode .....  | 378 |
| Figure 218. Preempted group auto conversion mode .....                                       | 379 |
| Figure 219. Repetition mode .....  | 379 |

|   |     |
|---|-----|
| Figure 220. Partition mod .....   | 380 |
| Figure 221. Ordinary oversampling restart mode selection .....                          | 382 |
| Figure 222. Ordinary oversampling trigger mode .....                                    | 383 |
| Figure 223. Oversampling of preempted group of channels.....                            | 383 |
| Figure 224. Data alignment .....  | 384 |
| Figure 225. Bit timing.....   | 397 |
| Figure 226. Frame type .....  | 398 |
| Figure 227. Transmit interrupt generation.....  | 399 |
| Figure 228. Receive interrupt 0 generation .....  | 399 |
| Figure 229. Receive interrupt 1 generation .....  | 399 |
| Figure 230. Status error interrupt generation.....                                      | 400 |
| Figure 231. CAN block diagram.....  | 401 |
| Figure 232. 32-bit identifier mask mode .....   | 403 |
| Figure 233. 32-bit identifier list mode.....  | 403 |
| Figure 234. 16-bit identifier mask mode .....   | 403 |
| Figure 235. 16-bit identifier list mode.....  | 403 |
| Figure 236. Transmit mailbox status .....   | 405 |
| Figure 237. Receive FIFO status .....   | 407 |
| Figure 238. Transmit and receive mailboxes .....  | 420 |
| Figure 239. Block diagram of OTGFS structure .....                                      | 425 |
| Figure 240. OTGFS interrupt hierarchy .....   | 428 |
| Figure 241. Writing the transmit FIFO.....  | 434 |
| Figure 242. Reading the receive FIFO .....  | 435 |
| Figure 243. HFIR behavior when HFIRRLDCTRL = 0x0 .....                                  | 436 |
| Figure 244. HFIR behavior when HFIRRLDCTRL = 0x1 .....                                  | 437 |
| Figure 245. Example of common Bulk/Control OUT/SETUP and Bulk/Control IN transfer ..... | 440 |
| Figure 246. Example of common interrupt OUT/IN transfers .....                          | 444 |
| Figure 247. Example of common synchronous OUT/IN transfers .....                        | 448 |
| Figure 248. Read receive FIFO .....   | 454 |
| Figure 249. SETUP data packet flowchart.....  | 456 |
| Figure 250. BULK OUT transfer block diagram .....                                       | 461 |
| Figure 251. CSR memory map .....  | 469 |
| Figure 252. ACC interrupt mapping diagram .....   | 515 |
| Figure 253. ACC block diagram.....  | 517 |
| Figure 254. Cross-return algorithm.....   | 517 |
| Figure 255. IRTMR block diagram.....  | 523 |

# 1 System architecture

The HT32F49041 series microcontrollers incorporates a 32-bit ARM® Cortex®-M4 processor core, multiple 16-bit and 32-bit timers, Infrared Transmitter (IRTMR), DMA controller, ERTC, communication interfaces such as SPI, I²C, USART/UART, CAN bus controller, USB2.0 full-speed interface, HICK with automatic clock calibration (ACC), 12-bit ADC, programmable voltage monitor (PVM) and other peripherals. Cortex®-M4 processor supports enhanced high-performance DSP instruction set, including extended single-cycle 16-bit/32-bit multiply accumulator (MAC), dual 16-bit MAC instructions, optimized 8-bit/16-bit SIMD operation and saturation operation instructions, as shown in Figure 1:

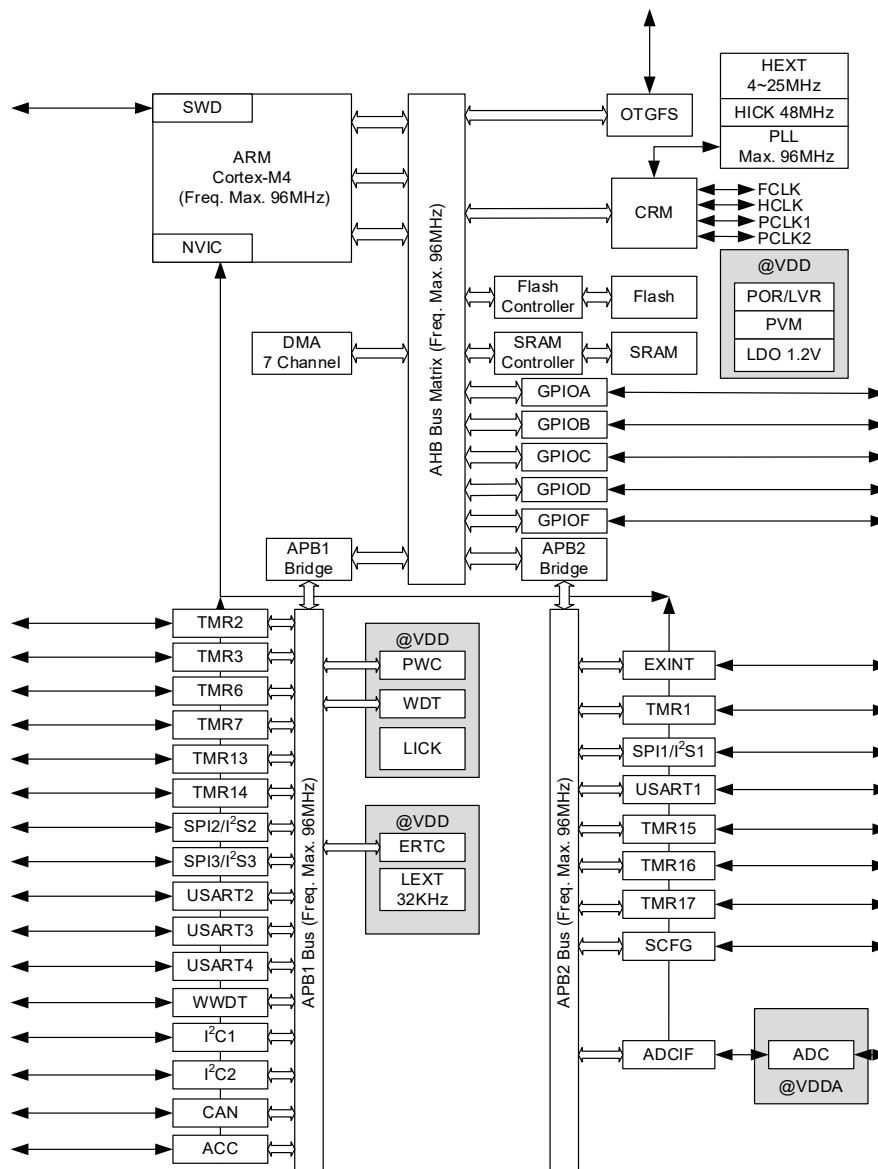


Figure 1. HT32F49041 Series microcontrollers system architecture

## Feature

- Core: ARM® 32-bit Cortex®-M4 CPU
  - 96 MHz maximum frequency, with a Memory Protection Unit (MPU), single-cycle multiplication and hardware division
  - DSP instructions
- Memories
  - 32 to 64 Kbytes of Flash memory
  - 4 Kbytes of boot memory used as a Bootloader or as a general instruction/data memory (one-time-configured)
  - sLib: configurable part of main Flash set as a library area with code executable but secured, non-readable
  - 20 Kbytes of SRAM
- Power control (PWC)
  - 2.4 V ~ 3.6 V application supply
  - Power-on reset (POR) / low voltage reset (LVR), and power voltage monitor (PVM)
  - Low power: Sleep, Deepsleep, and Standby modes, 6 WKUP pins can wake up Standby mode
  - 5 × 32-bit battery power registers (BPR)
- Clock and reset management (CRM)
  - 4 to 25 MHz crystal oscillator (HEXT)
  - Internal 48 MHz factory-trimmed clock (HICK), accuracy  $\pm 1\%$  at  $T_A = 25\text{ }^\circ\text{C}$ ,  $\pm 2.5\%$  at  $T_A = -40$  to  $+105\text{ }^\circ\text{C}$ , with automatic clock calibration (ACC)
  - PLL with configurable frequency multiplication and division factor
  - 32.768 kHz crystal oscillator (LEXT)
  - Internal 40 kHz RC oscillator (LICK)
- Analog
  - 1 × 12-bit 2 MSPS A/D converter, up to 16 input channels, hardware over-sampling up to equivalent 16-bit resolution
  - Internal reference voltage ( $V_{INTRV}$ )
- DMA
  - 1 × DMA controller for flexible mapping support
  - 7 channels in all
- Up to 55 Fast I/O Interfaces
  - All mappable to 16 external interrupt vectors
  - Almost 5 V-tolerant
- Up to 13 Timers (TMR)
  - 1 × 16-bit 7-channel advanced timer, 6-channel PWM output with dead-time generator and emergency brake

- Up to  $6 \times 16$ -bit and  $1 \times 32$ -bit general-purpose timers, each with 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
- Advanced and general-purpose timers provide up to 24-channel PWM
- $2 \times 16$ -bit basic timers
- $2 \times$  watchdog timers (WDT and WWDT)
- SysTick timer: 24-bit downcounter
- ERTC: enhanced RTC with auto wakeup, alarm, subsecond precision, hardware calendar and calibration feature
- Up to 12 communication interfaces
  - Up to  $2 \times$  I<sup>2</sup>C interfaces (SMBus/PMBus)
  - Up to  $4 \times$  USARTs, support master synchronous SPI and modem control, with ISO7816 interface, LIN, IrDA and RS485 driver enable; support TX/RX swap
  - Up to  $3 \times$  SPIs (36 Mbit/s), all with I<sup>2</sup>S interface multiplexed, any combination of two interfaces supports full-duplex
  - CAN interface (2.0B Active), with 256 bytes of dedicated SRAM
  - OTG full speed interface, with 1280 bytes of dedicated SRAM, supporting crystal-less in device mode
  - Infrared transmitter (IRTMR)
- CRC Calculation Unit
- 96-bit ID (UID)
- Debug mode
  - Serial wire debug (SWD) and serial wire output (SWO)
- Temperature range: -40 to 105 °C
- Packaging
  - LQFP64  $7 \times 7$  mm
  - LQFP48  $7 \times 7$  mm
  - QFN32  $4 \times 4$  mm
  - TSSOP20  $6.5 \times 4.4$  mm
- List of Models

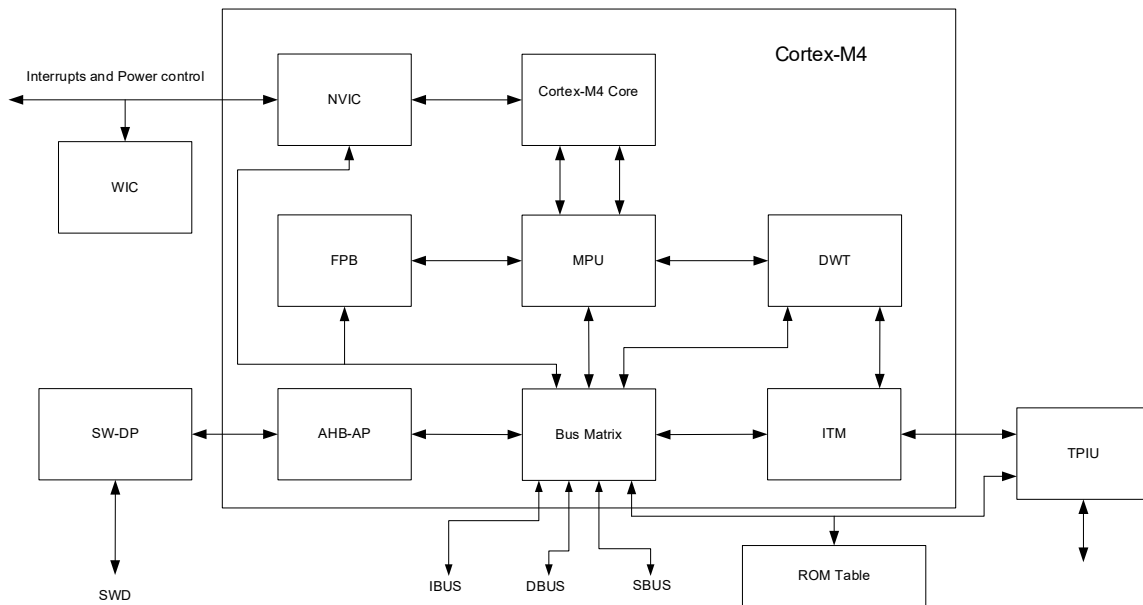
| Internal Flash | Part number |
|----------------|-------------|
| 64 Kbytes      | HT32F49041  |

## System overview

### ARM Cortex®-M4 processor

Cortex®-M4 processor is a low-power consumption processor featuring low gate count, low interrupt latency, and low-cost debug. It supports DSP instruction set and FPU, and is applicable to deeply-embedded applications that require quicker response to interruption. Cortex®-M4 processor is based on ARMv7-M architecture, supporting both Thumb instruction set and DSP instruction set.

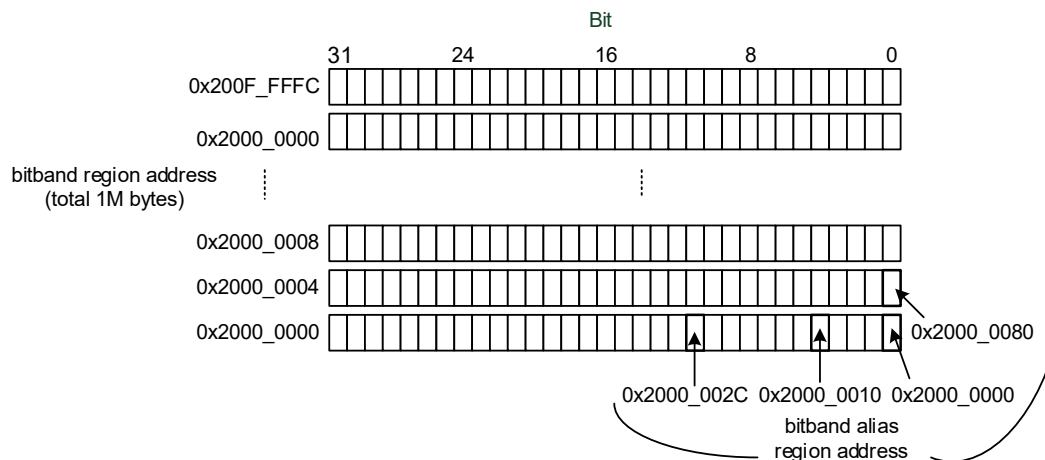
Figure 2 shows the internal block diagram of Cortex®-M4 processor. Please refer to ARM Cortex®-M4 Technical Reference Manual for more information.



**Figure 2. Internal block diagram of Cortex®-M4**

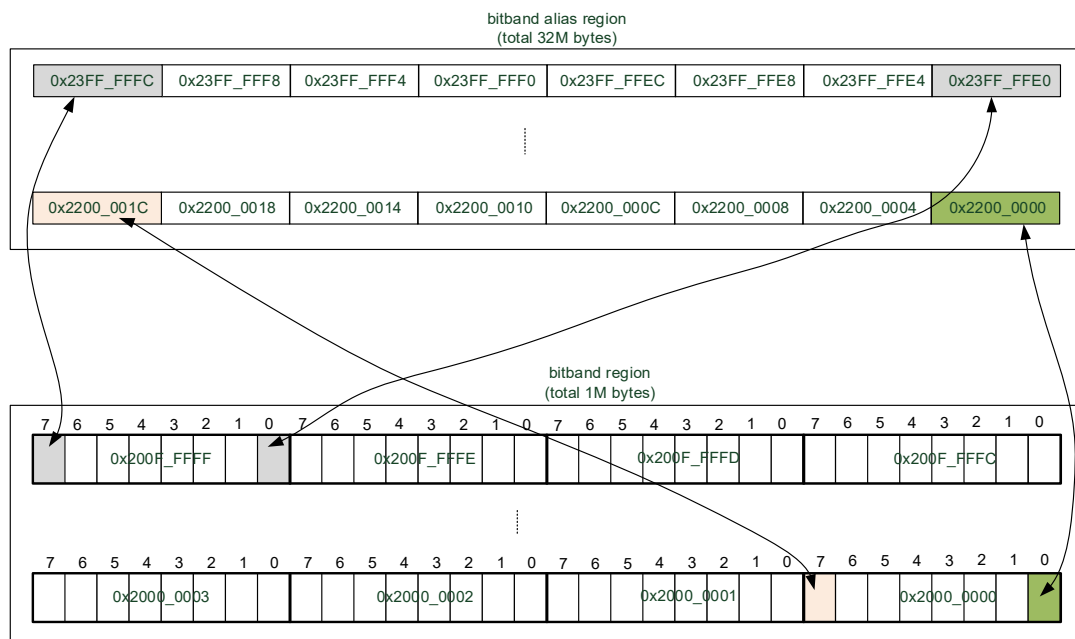
## Bit band

With the help of bit-band, read and write access to a single bit can be performed using common load/store operations. The Cortex®-M4 memory includes two bit-band regions: the least significant 1M bytes of SRAM and the least significant 1Mbytes of peripherals. In addition to access to bit-band addresses, their respective bit-band alias region can be used to access to any bit in these two bit-band regions. The bit-band alias region transforms each bit into a 32-bit word. Thus, accessing to a bit in an alias region has the same effect as read-modify-write operation on the corresponding bit in a bit-band region.



**Figure 3. Comparison between bit-band region and its alias region: image A**





**Figure 4. Comparison between bit-band region and its alias region: image B**

Bit-band region: address region for bit-band operations

Bit-band alias region: access to the alias region has the same effect as read-modify-write operation on the bit-band region

Each bit in a bit-band region is mapped into a word (LSB) in an alias region. When accessing to the address in a bit-band alias region, such address is transformed into a bit-band address first. For a read operation, read one word in the bit-band region, and then move the targeted bit to the right to LSB before returning LSB. For a write operation, first move the targeted bit to the left to the corresponding bit number, then perform a read-modify-write operation on bit level.

The address ranges of two memories supporting bit-band operations:

The lowest 1 Mbyte of the SRAM: 0x2000\_0000 ~ 0x200F\_FFFF

The lowest 1 Mbyte of peripherals: 0x4000\_0000 ~ 0x400F\_FFFF

For a bit in the SRAM bit-band region, if the byte address is A, the bit number is n ( $0 \leq n \leq 7$ ), then the alias address where the bit is:

$$\text{AliasAddr} = 0x2200\_0000 + (A - 0x2000\_0000) \times 32 + n \times 4$$

For a bit in the peripheral bit-band region, if the byte address is A, the bit number is n ( $0 \leq n \leq 7$ ), then the alias address where the bit is:

$$\text{AliasAddr} = 0x4200\_0000 + (A - 0x4000\_0000) \times 32 + n \times 4$$

Table 1 shows the mapping between bit-band region and alias region in SRAM:

**Table 1. Bit-band address mapping in SRAM**

| Bit-band region | Equivalent alias address |
|-----------------|--------------------------|
| 0x2000_0000.0   | 0x2200_0000.0            |
| 0x2000_0000.1   | 0x2200_0004.0            |
| 0x2000_0000.2   | 0x2200_0008.0            |
| ...             | ...                      |
| 0x2000_0000.31  | 0x2200_007C.0            |
| 0x2000_0004.0   | 0x2200_0080.0            |
| 0x2000_0004.1   | 0x2200_0084.0            |
| 0x2000_0004.2   | 0x2200_0088.0            |
| ...             | ...                      |
| 0x200F_FFFC.31  | 0x23FF_FFFC.0            |

Table 2 shows the mapping between bit-band region and alias region in the peripheral area:

**Table 2. Bit-band address mapping in the peripheral area**

| Bit-band region | Equivalent alias address |
|-----------------|--------------------------|
| 0x4000_0000.0   | 0x4200_0000.0            |
| 0x4000_0000.1   | 0x4200_0004.0            |
| 0x4000_0000.2   | 0x4200_0008.0            |
| ...             | ...                      |
| 0x4000_0000.31  | 0x4200_007C.0            |
| 0x4000_0004.0   | 0x4200_0080.0            |
| 0x4000_0004.1   | 0x4200_0084.0            |
| 0x4000_0004.2   | 0x4200_0088.0            |
| ...             | ...                      |
| 0x400F_FFFC.31  | 0x43FF_FFFC.0            |

In addition, bit-band operations can also simplify jump process. When jump operation is based on a bit level, the previous steps are:

- Read the whole register
- Mask the undesired bits
- Compare and jump

For now, you just need do:

- Read the bit status from the bit-band alias region
- Compare and jump

Apart from making code more concise, its important function is also reflected in multi-task environment. When it comes to multiple tasks, it turns the read-modify-write operations into a hardware-supported atomic operation to avoid the scenario where the read-modify-write operation is disrupted, resulting in disorder.

## Interrupt and exception vectors

**Table 3. HT32F49041 series vector table**

| Pos. | Priority | Priority Type | Name   | Description   | Address                      |
|------|----------|---------------|--|---|------------------------------|
|      | —        | —             | —  | Reserved  | 0x0000_0000                  |
|      | -3       | Fixed         | Reset  | Reset   | 0x0000_0004                  |
|      | -2       | Fixed         | NMI  | Non maskable interrupt<br>CRM clock fail detector (CFD) is linked to NMI vector | 0x0000_0008                  |
|      | -1       | Fixed         | HardFault                                    | All class of fault  | 0x0000_000C                  |
|      | 0        | Configurable  | MemoryManage                                 | Memory management   | 0x0000_0010                  |
|      | 1        | Configurable  | BusFault                                     | Pre-fetch fault, memory access fault  | 0x0000_0014                  |
|      | 2        | Configurable  | UsageFault                                   | Undefined instruction or illegal state  | 0x0000_0018                  |
|      | —        | —             | —  | Reserved  | 0x0000_001C<br>~ 0x0000_002B |
|      | 3        | Configurable  | SVCall                                       | System service call via SWI instruction   | 0x0000_002C                  |
|      | 4        | Configurable  | Debug Monitor                                | Debug monitor   | 0x0000_0030                  |
|      | —        | —             | —  | Reserved  | 0x0000_0034                  |
|      | 5        | Configurable  | PendSV                                       | Pendable request for system service   | 0x0000_0038                  |
|      | 6        | Configurable  | SysTick                                      | System tick timer   | 0x0000_003C                  |
| 0    | 7        | Configurable  | WWDT   | Window watchdog timer   | 0x0000_0040                  |
| 1    | 8        | Configurable  | PVM  | PVM from EXINT interrupt  | 0x0000_0044                  |
| 2    | 9        | Configurable  | TAMPER                                       | Tamper interrupt  | 0x0000_0048                  |
| 3    | 10       | Configurable  | FLASH  | Flash global interrupt  | 0x0000_004C                  |
| 4    | 11       | Configurable  | CRM  | Clock and Reset manage (CRM) interrupt  | 0x0000_0050                  |
| 5    | 12       | Configurable  | EXINT1_0                                     | EXINT line1_0 interrupt   | 0x0000_0054                  |
| 6    | 13       | Configurable  | EXINT3_2                                     | EXINT line3_2 interrupt   | 0x0000_0058                  |
| 7    | 14       | Configurable  | EXINT15_4                                    | EXINT line15_4 interrupt  | 0x0000_005C                  |
| 8    | 15       | Configurable  | ACC  | ACC interrupt   | 0x0000_0060                  |
| 9    | 16       | Configurable  | DMA channel 1                                | DMA channel 1 global interrupt  | 0x0000_0064                  |
| 10   | 17       | Configurable  | DMA channel 3_2                              | DMA channel 3_2 global interrupt  | 0x0000_0068                  |
| 11   | 18       | Configurable  | DMA channel 7_4                              | DMA channel 7_4 global interrupt  | 0x0000_006C                  |
| 12   | 19       | Configurable  | ADC  | ADC global interrupt  | 0x0000_0070                  |
| 13   | 20       | Configurable  | TMR1_BRK<br>TMR1_UP<br>TMR1_TRG<br>TMR1_HALL | TMR1 interrupt  | 0x0000_0074                  |
| 14   | 21       | Configurable  | TMR1_CH                                      | TMR1 capture compare interrupt  | 0x0000_0078                  |
| 15   | 22       | Configurable  | TMR2   | TMR2 global interrupt   | 0x0000_007C                  |
| 16   | 23       | Configurable  | TMR3   | TMR3 global interrupt   | 0x0000_0080                  |
| 17   | 24       | Configurable  | TMR6   | TMR6 global interrupt   | 0x0000_0084                  |
| 18   | 25       | Configurable  | TMR7   | TMR7 global interrupt   | 0x0000_0088                  |
| 19   | 26       | Configurable  | TMR14  | TMR14 global interrupt  | 0x0000_008C                  |
| 20   | 27       | Configurable  | TMR15  | TMR15 global interrupt  | 0x0000_0090                  |
| 21   | 28       | Configurable  | TMR16  | TMR16 global interrupt  | 0x0000_0094                  |
| 22   | 29       | Configurable  | TMR17  | TMR17 global interrupt  | 0x0000_0098                  |
| 23   | 30       | Configurable  | I2C1_EVT                                     | I2C1 event interrupt  | 0x0000_009C                  |

| Pos. | Priority | Priority Type | Name     | Description   | Address     |
|------|----------|---------------|----------|---|-------------|
| 24   | 31       | Configurable  | I2C2_EVT | I2C2 event interrupt  | 0x0000_00A0 |
| 25   | 32       | Configurable  | SPI1     | SPI1 global interrupt   | 0x0000_00A4 |
| 26   | 33       | Configurable  | SPI2     | SPI2 global interrupt   | 0x0000_00A8 |
| 27   | 34       | Configurable  | USART1   | USART1 global interrupt   | 0x0000_00AC |
| 28   | 35       | Configurable  | USART2   | USART2 global interrupt   | 0x0000_00B0 |
| 29   | 36       | Configurable  | USART3_4 | USART3_4 global interrupt   | 0x0000_00B4 |
| 30   | 37       | Configurable  | CAN      | CAN global interrupt  | 0x0000_00B8 |
| 31   | 38       | Configurable  | OTGFS    | OTGFS global interrupt and OTGFS wakeup interrupt through EXINT line 18 | 0x0000_00BC |
| 32   | 39       | Configurable  | I2C1_ERR | I <sup>2</sup> C1 error interrupt                                       | 0x0000_00C0 |
| 33   | 40       | Configurable  | SPI3     | SPI3 global interrupt   | 0x0000_00C4 |
| 34   | 41       | Configurable  | I2C2_ERR | I <sup>2</sup> C2 error interrupt                                       | 0x0000_00C8 |
| 35   | 42       | Configurable  | TMR13    | TMR1 channel interrupt  | 0x0000_00C8 |

## System Tick (SysTick)

The System Tick is a 24-bit downcounter. It will be reloaded with the initial value automatically when it is decremented to zero. It can generate periodic interrupts, so it is often used as multi-task scheduling counter for embedded operating system, and also to call the periodic tasks for non-embedded system.

The System Tick calibration value is fixed to 9000, which gives a reference time base of 1 ms when the System Tick clock is set to 9 MHz.

## Reset

The processor reads the first two words from the CODE memory after a system reset and before program execution.

- Get the initial value of the main stack pointer (MSP) from address 0x0000\_0000.
- Get the initial value of the program counter (PC) from address 0x0000\_0004. This value is a reset vector and LSB must be 1. Then take the instructions from the address corresponding to this value.

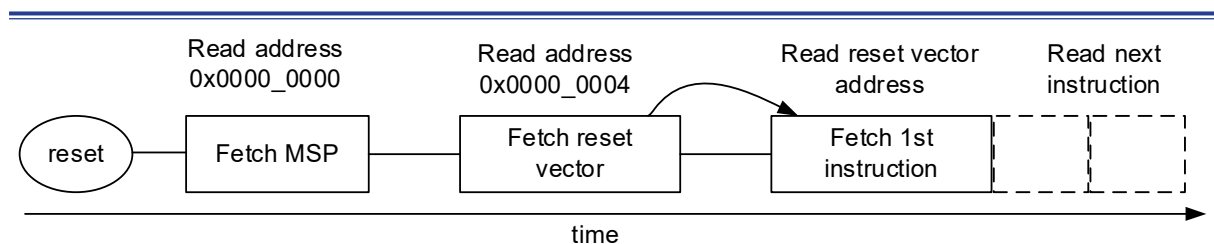
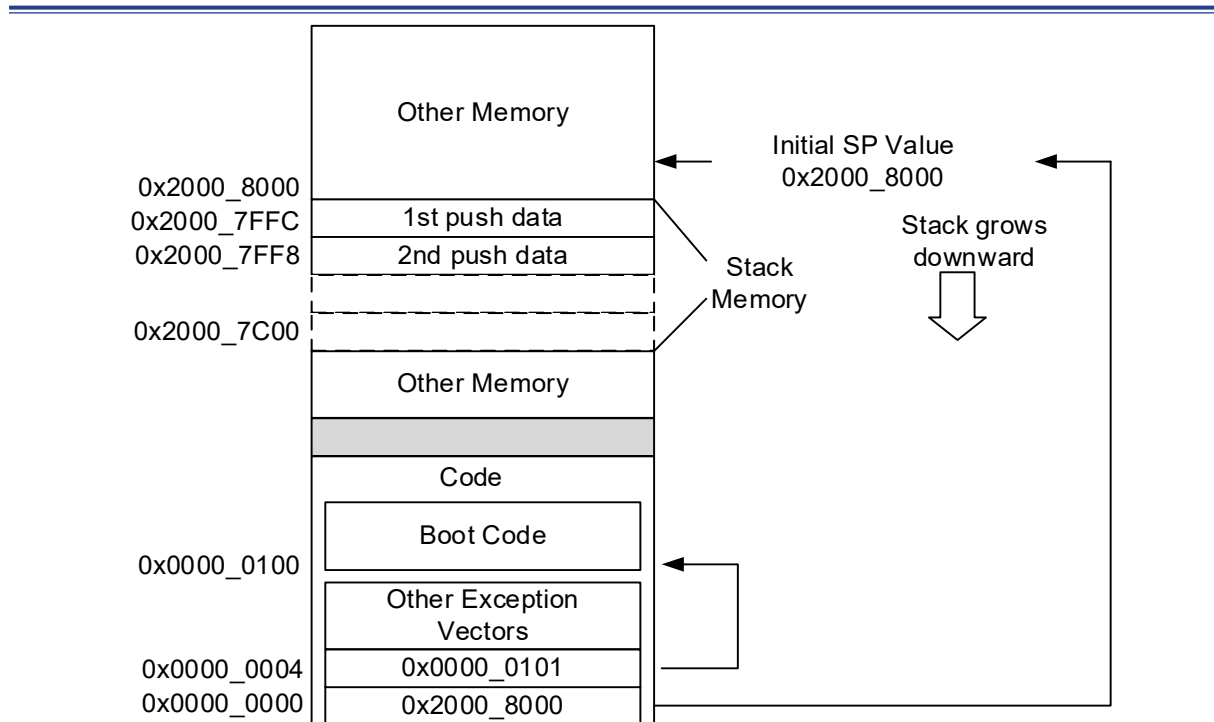


Figure 5. Reset process

Cortex®-M4 uses a full stack that increases downward, so the initial value of the main stack pointer (MSP) must be the end address of the stack memory plus 1. For example, if the stack area is set between 0x2000\_7C00 and 0x2000\_7FFF, then the initial value of MSP must be defined as 0x2000\_8000.

The vector table follows the initial value of MSP. Cortex®-M4 operates in Thumb state, and thus each value in the vector table must set the LSB to 1. In Figure 6, 0x0000\_0101 is used to represent 0x0000\_0100. After the instruction at 0x0000\_0100 is executed, the program starts running formally. Before that, it is a must for initializing MSP, because the first instruction may be interrupted by NMI or other faults before being executed. After the completion of MSP initialization, it is ready to prepare stack room for its service routines.



**Figure 6. Example of MSP and PC initialization**

In the HT32F49041 series, the main Flash memory, Boot code or SRAM can be remapped to the code area between 0x0000\_0000 and 0x07FF\_FFFF. nBOOT1 corresponds to the value of the bit nBOOT1 in the SSB of the User System Data (USD). nBOOT1 and BOOT0 are used to set the specific memory from which CODE starts.

{nBOOT1, BOOT0} = 00/10, CODE starts from the main Flash memory

{nBOOT1, BOOT0} = 11, CODE starts from boot memory

{nBOOT1, BOOT0} = 01, CODE starts from SRAM

After a system reset or when leaving from Standby mode, the pin values of both nBOOT1 and BOOT0 will be relatched. When the CODE starts from SRAM, the status of BOOT is latched, and it is impossible to load a new boot mode through a system reset. At this point, the power-on reset must be performed to reload a new boot code mode.

Boot code memory contains an embedded bootloader program that provides not only Flash programming function through USART1 or USART2, but also provides extra firmware including communication protocol stacks that can be called for use by software developer through API.

## List of abbreviations for registers

**Table 4. List of abbreviations for registers**

| Register type | Description   |
|---------------|---|
| rw            | Software can read and write to this bit.  |
| ro            | Software can only read this bit.  |
| wo            | Software can only write to the bit. Reading it returns its reset value.                     |
| rrc           | Software can read this bit. Reading this bit automatically clears it.                       |
| rw0c          | Software can read this bit and clear it by writing 0. Writing 1 has no effect on this bit.  |
| rw1c          | Software can read this bit and clear it by writing 1. Writing 0 has no effect on this bit.  |
| rw1s          | Software can read this bit and set it by writing 1. Writing 0 has no effect on this bit.    |
| tog           | Software can read this bit and toggle it by writing 1. Writing 0 has no effect on this bit. |
| rwt           | Software can read this bit. Writing any value will trigger an event.                        |
| resd          | Reserved.   |

## Device characteristics information

**Table 5. List of abbreviations for registers**

| Register abbr. | Base address | Reset value |
|----------------|--------------|-------------|
| F_SIZE         | 0x1FFF F7E0  | 0XXXXX      |
| UID[31:0]      | 0x1FFF F7E8  | 0XXXXX XXXX |
| UID[63:32]     | 0x1FFF F7EC  | 0XXXXX XXXX |
| UID[95:64]     | 0x1FFF F7F0  | 0XXXXX XXXX |

### Flash memory size register

This register contains the information about Flash memory size.

| Bit      | Abbr.  | Reset value | Type | Description  |
|----------|--------|-------------|------|--|
| Bit 15:0 | F_SIZE | 0XXXXX      | ro   | Flash size, in terms of Kbyte<br>For example: 0x0080 = 128 Kbyte |

### Device electronic signature

The device electronic signature contains the memory size and the unique device ID (96 bits). It is stored in the information block of the Flash memory. The 96-bit ID is unique for any device, and cannot be altered by users. It can be used for the following:

- Serial number: such as USB string serial number
- Part of security keys

| Bit      | Abbr.      | Reset value | Type | Description              |
|----------|------------|-------------|------|--------------------------|
| Bit 31:0 | UID[31:0]  | 0XXXXX XXXX | ro   | UID for bit 31 to bit 0  |
| Bit      | Abbr.      | Reset value | Type | Description              |
| Bit 31:0 | UID[63:32] | 0XXXXX XXXX | ro   | UID for bit 63 to bit 32 |
| Bit      | Abbr.      | Reset value | Type | Description              |
| Bit 31:0 | UID[95:64] | 0XXXXX XXXX | ro   | UID for bit 95 to bit 64 |

Note: UID[95:88] is series ID, which is 0x97 for HT32F49041.

## 2 Memory resources

### Internal memory address map

Internal memory contains program memory (Flash), data memory (SRAM), peripheral registers and core registers. Their respective address mapping are shown in Figure 7.

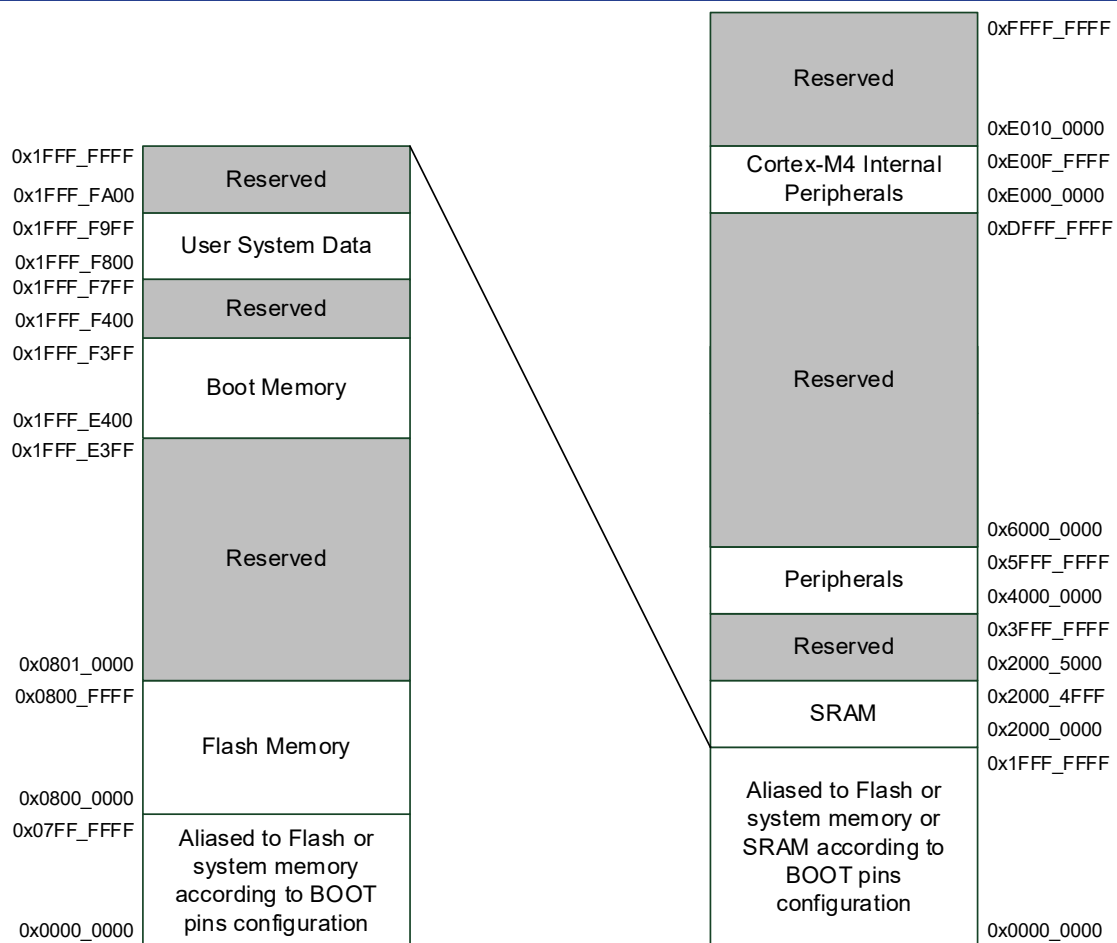


Figure 7. HT32F49041 address mapping

### Flash memory

HT32F49041 series provide up to 64 KB of on-chip Flash memory, supporting a single-cycle 32-bit read operation.

Refer to Chapter 5 for more details about Flash memory controller and register configuration.

#### Flash memory organization (64 KB)

The main memory contains only bank 1 (64 Kbytes), which is divided into 64 sectors of 1 Kbytes.

**Table 6. Flash memory organization (64 KB)**

| Bank  |                  | Name                      | Address range             |
|---|------------------|---------------------------|---------------------------|
| Main memory                                 | Bank1<br>(64 KB) | Sector 0                  | 0x0800 0000 – 0x0800 03FF |
|   |                  | Sector 1                  | 0x0800 0400 – 0x0800 07FF |
|   |                  | Sector 2                  | 0x0800 0800 – 0x0800 0BFF |
|   |                  | ...                       | ...                       |
|   |                  | Sector 63                 | 0x0800 FC00 – 0x0800 FFFF |
| Information block<br>512 B user system data |                  | 4 KB bootloader           | 0x1FFF E400 – 0x1FFF F3FF |
|   |                  | 0x1FFF F800 – 0x1FFF F9FF |                           |

### Flash memory organization (32 KB)

The main memory contains only bank 1 (32 Kbytes), which is divided into 32 sectors of 1 Kbytes.

**Table 7. Flash memory organization (64 KB)**

| Bank  |                  | Name                      | Address range             |
|---|------------------|---------------------------|---------------------------|
| Main memory                                 | Bank1<br>(32 KB) | Sector 0                  | 0x0800 0000 – 0x0800 03FF |
|   |                  | Sector 1                  | 0x0800 0400 – 0x0800 07FF |
|   |                  | Sector 2                  | 0x0800 0800 – 0x0800 0BFF |
|   |                  | ...                       | ...                       |
|   |                  | Sector 31                 | 0x0800 FC00 – 0x0800 FFFF |
| Information block<br>512 B user system data |                  | 4 KB bootloader           | 0x1FFF E400 – 0x1FFF F3FF |
|   |                  | 0x1FFF F800 – 0x1FFF F9FF |                           |

## SRAM memory

The HT32F49041 series contain a 20 KB on-chip SRAM which starts at the address 0x2000\_0000. It can be accessed by bytes, half-words (16-bit) or words (32-bit).



## Peripheral address map

Table 8. Peripheral boundary address

| Bus | Boundary address          | Peripherals                    |
|-----|---------------------------|--------------------------------|
| AHB | 0xA000 1000 - 0xFFFF FFFF | Reserved                       |
|     | 0x6000 0000 - 0xA000 0FFF | Reserved                       |
|     | 0x5004 0000 - 0x5FFF FFFF | Reserved                       |
|     | 0x5000 0000 - 0x5003 FFFF | OTGFS                          |
|     | 0x4800 1800 - 0x4FFF FFFF | Reserved                       |
|     | 0x4800 1400 - 0x4800 17FF | GPIOF                          |
|     | 0x4800 1000 - 0x4800 13FF | Reserved                       |
|     | 0x4800 0C00 - 0x4800 0FFF | GPIOD                          |
|     | 0x4800 0800 - 0x4800 0BFF | GPIOC                          |
|     | 0x4800 0400 - 0x4800 07FF | GPIOB                          |
|     | 0x4800 0000 - 0x4800 03FF | GPIOA                          |
|     | 0x4002 3400 - 0x47FF FFFF | Reserved                       |
|     | 0x4002 3000 - 0x4002 33FF | CRC                            |
|     | 0x4002 2000 - 0x4002 23FF | Flash memory interface (FLASH) |
|     | 0x4002 1400 - 0x4002 1FFF | Reserved                       |
|     | 0x4002 1000 - 0x4002 13FF | Clock and reset manage (CRM)   |
|     | 0x4002 0800 - 0x4002 0FFF | Reserved                       |
|     | 0x4002 0400 - 0x4002 07FF | Reserved                       |
|     | 0x4002 0000 - 0x4002 03FF | DMA                            |
|     | 0x4001 8400 - 0x4001 7FFF | Reserved                       |
|     | 0x4001 8000 - 0x4001 83FF | Reserved                       |

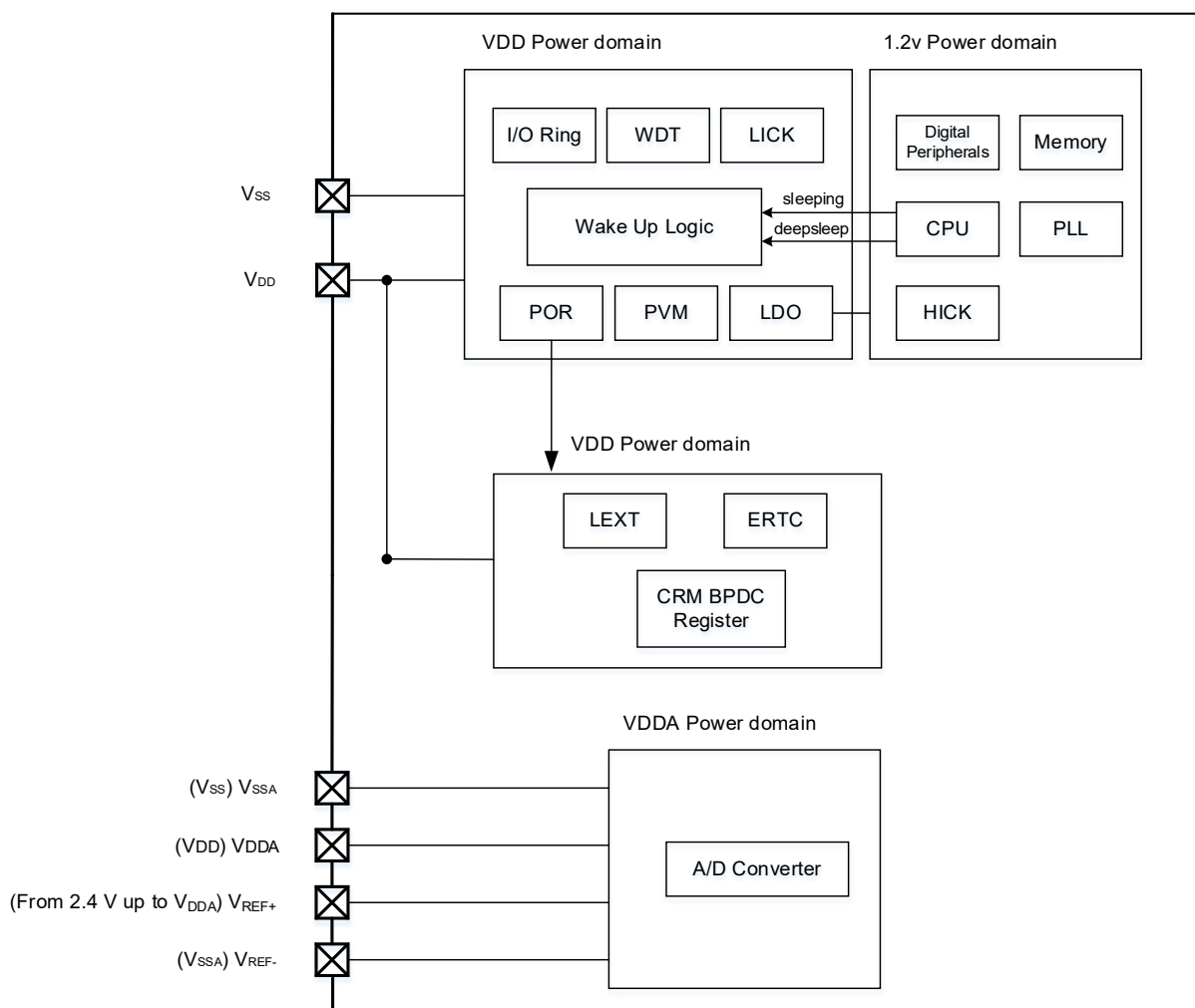
| Bus  | Boundary address          | Peripherals            |
|------|---------------------------|------------------------|
| APB2 | 0x4001 7C00 - 0x4001 7FFF | Reserved               |
|      | 0x4001 7800 - 0x4001 7BFF | Reserved               |
|      | 0x4001 7400 - 0x4001 77FF | Reserved               |
|      | 0x4001 7000 - 0x4001 73FF | Reserved               |
|      | 0x4001 6C00 - 0x4001 6FFF | Reserved               |
|      | 0x4001 6800 - 0x4001 6BFF | Reserved               |
|      | 0x4001 6400 - 0x4001 67FF | Reserved               |
|      | 0x4001 6000 - 0x4001 63FF | Reserved               |
|      | 0x4001 5C00 - 0x4001 5FFF | Reserved               |
|      | 0x4001 5800 - 0x4001 5BFF | Reserved               |
|      | 0x4001 5400 - 0x4001 57FF | Reserved               |
|      | 0x4001 5000 - 0x4001 53FF | Reserved               |
|      | 0x4001 4C00 - 0x4001 4FFF | Reserved               |
|      | 0x4001 4800 - 0x4001 4BFF | TMR17 timer            |
|      | 0x4001 4400 - 0x4001 47FF | TMR16 timer            |
|      | 0x4001 4000 - 0x4001 43FF | TMR15 timer            |
|      | 0x4001 3C00 - 0x4001 3FFF | Reserved               |
|      | 0x4001 3800 - 0x4001 3BFF | USART1                 |
|      | 0x4001 3400 - 0x4001 37FF | Reserved               |
|      | 0x4001 3000 - 0x4001 33FF | SPI1/I <sup>2</sup> S1 |
|      | 0x4001 2C00 - 0x4001 2FFF | TMR1 timer             |
|      | 0x4001 2800 - 0x4001 2BFF | Reserved               |
|      | 0x4001 2400 - 0x4001 27FF | ADC                    |
|      | 0x4001 2000 - 0x4001 23FF | Reserved               |
|      | 0x4001 1C00 - 0x4001 1FFF | Reserved               |
|      | 0x4001 1800 - 0x4001 1BFF | Reserved               |
|      | 0x4001 1400 - 0x4001 17FF | Reserved               |
|      | 0x4001 1000 - 0x4001 13FF | Reserved               |
|      | 0x4001 0C00 - 0x4001 0FFF | Reserved               |
|      | 0x4001 0800 - 0x4001 0BFF | Reserved               |
|      | 0x4001 0400 - 0x4001 07FF | EXINT                  |
|      | 0x4001 0000 - 0x4001 03FF | SCFG                   |

| Bus  | Boundary address          | Peripherals                  |
|------|---------------------------|------------------------------|
| APB1 | 0x4000 8000 - 0x4000 FFFF | Reserved                     |
|      | 0x4000 7C00 - 0x4000 7FFF | Reserved                     |
|      | 0x4000 7800 - 0x4000 7BFF | Reserved                     |
|      | 0x4000 7400 - 0x4000 77FF | Reserved                     |
|      | 0x4000 7000 - 0x4000 73FF | Power control (PWC)          |
|      | 0x4000 6C00 - 0x4000 6FFF | ACC                          |
|      | 0x4000 6800 - 0x4000 6BFF | Reserved                     |
|      | 0x4000 6400 - 0x4000 67FF | CAN                          |
|      | 0x4000 6000 - 0x4000 63FF | Reserved                     |
|      | 0x4000 5C00 - 0x4000 5FFF | Reserved                     |
|      | 0x4000 5800 - 0x4000 5BFF | I <sup>2</sup> C2            |
|      | 0x4000 5400 - 0x4000 57FF | I <sup>2</sup> C1            |
|      | 0x4000 5000 - 0x4000 53FF | Reserved                     |
|      | 0x4000 4C00 - 0x4000 4FFF | USART4                       |
|      | 0x4000 4800 - 0x4000 4BFF | USART3                       |
|      | 0x4000 4400 - 0x4000 47FF | USART2                       |
|      | 0x4000 4000 - 0x4000 43FF | Reserved                     |
|      | 0x4000 3C00 - 0x4000 3FFF | SPI3/I <sup>2</sup> S3       |
|      | 0x4000 3800 - 0x4000 3BFF | SPI2/I <sup>2</sup> S2       |
|      | 0x4000 3400 - 0x4000 37FF | Reserved                     |
|      | 0x4000 3000 - 0x4000 33FF | Watchdog timer (WDT)         |
|      | 0x4000 2C00 - 0x4000 2FFF | Window watchdog timer (WWDT) |
|      | 0x4000 2800 - 0x4000 2BFF | ERTC                         |
|      | 0x4000 2400 - 0x4000 27FF | Reserved                     |
|      | 0x4000 2000 - 0x4000 23FF | TMR14 timer                  |
|      | 0x4000 1C00 - 0x4000 1FFF | TMR13 timer                  |
|      | 0x4000 1800 - 0x4000 1BFF | Reserved                     |
|      | 0x4000 1400 - 0x4000 17FF | TMR7 timer                   |
|      | 0x4000 1000 - 0x4000 13FF | TMR6 timer                   |
|      | 0x4000 0C00 - 0x4000 0FFF | Reserved                     |
|      | 0x4000 0800 - 0x4000 0BFF | Reserved                     |
|      | 0x4000 0400 - 0x4000 07FF | TMR3 timer                   |
|      | 0x4000 0000 - 0x4000 03FF | TMR2 timer                   |

## 3 Power control (PWC)

### Introduction

The HT32F49041 operating voltage supply is 2.4V ~ 3.6V, with a temperature range of -40 ~ +105 °C. To reduce power consumption, this series provides three types of power saving modes, including Sleep, Deepsleep and Standby modes so as to achieve the best tradeoff among the conflicting demands of CPU operating time, speed and power consumption. The HT32F49041 series have two power domains - VDD/VDDA domain and 1.2V domain. The VDD/VDDA domain is supplied directly by external power, the 1.2V domain is powered by an embedded LDO in the VDD/VDDA domain.



**Figure 8. Block diagram of each power supply**

## Main Features

- Two power domains: VDD/VDDA domain and 1.2V domain
- Three types of power saving modes: Sleep mode, Deepsleep mode, and Standby mode
- Internal voltage regulator supplies 1.2V voltage source for the core domain
- Power voltage detector is provided to issue an interrupt when the supply voltage is lower or higher than a programmed threshold
- VDD/VDDA applies separated digital and analog module to reduce noise on external power

## POR/LVR

A POR analog module embedded in the VDD/VDDA domain is used to generate a power reset. The power reset signal is released at  $V_{POR}$  when the VDD is increased from 0 V to the operating voltage, or it is triggered at  $V_{LVR}$  when the VDD drops from the operating voltage to 0 V. During the power-on reset period, the reset signal has certain amount of time delay compared to VDD boost process. At the same time, hysteresis occurs in power-on reset (POR) and low voltage reset (LVR).

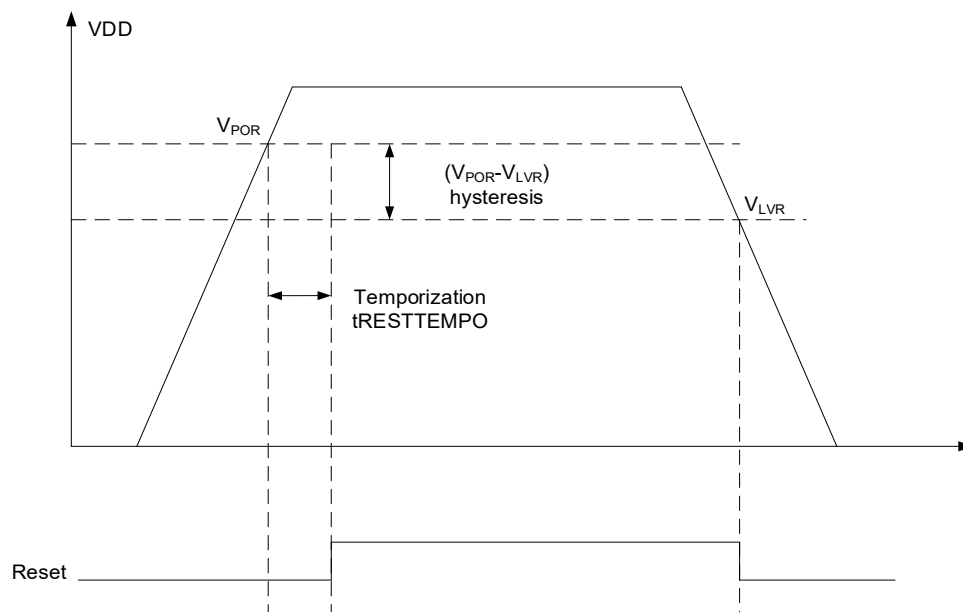


Figure 9. Power-on reset/Low voltage reset waveform

## Power voltage monitor (PVM)

The PVM is used to monitor the power supply variations. It is enabled by setting the PVMEN bit in the power control register (PWC\_CTRL), and the threshold value for voltage monitor is selected with the PVMSEL[2:0].

After PVM is enabled, the comparison result between VDD and the programmed threshold is indicated by the PVMOF bit in the PWC\_CTRLSTS register, with the hysteresis voltage  $V_{HYS\_P}$  being 100 mv. The PVM interrupt will be generated through the EXTI line 16 when VDD rises above the PVM threshold.

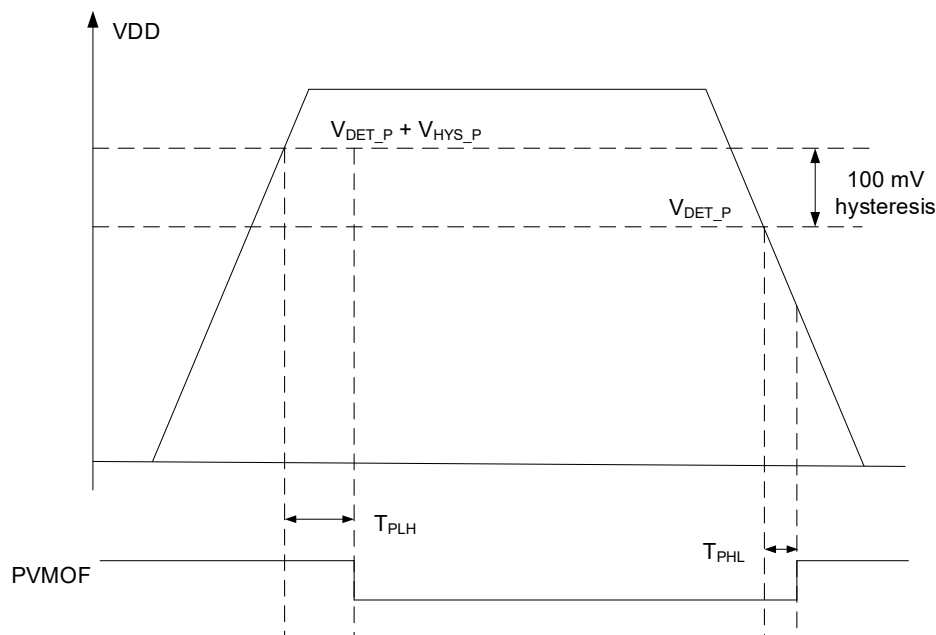


Figure 10. PVM threshold and output

## Power domain

### 1.2V domain

1.2V core domain includes a CPU core, SRAM, embedded digital peripherals and Phase Locked Loop (PLL). Such power domain is supplied by LDO (voltage regulator).

### VDD/VDDA domain

VDD/VDDA domain includes VDD domain and VDDA domain. The VDD domain contains I/O circuit, power-saving mode wakeup circuit, watchdog timer (WDT), power-on reset/low voltage reset (POR/LVR), LDO, ERTC circuit, LEXT oscillator and all PAD circuits. The VDDA domain contains an ADC (AD converters), and so on.

Typically, to ensure a better accuracy of ADC at a low voltage, the digital circuit is supplied by VDD while the analog circuit is powered by VDDA. The external reference voltage VREF+ and VREF- are connected to the VDDA pin and VSSA pin, respectively.

## Power saving modes

When the CPU does not need to be kept running, there are three types of low-power modes available (Sleep mode, Deepsleep mode and Standby mode) to save power. Users can select the mode that gives the best compromise according to the low-power consumption, short startup time, and available wakeup sources. In addition, the power consumption in Run mode can be reduced by slowing down the system clocks or gating the clocks to the APB and AHB peripherals when they are not used.

### Sleep mode

The Sleep mode is entered by executing WFI or WFE instruction. There are two options to select the Sleep mode entry mechanism through the SLEEPONEXIT bit in the Cortex®-M4 system control register.

#### SLEEP-NOW mode:

When SLEEPDEEP = 0 and SLEEPONEXIT = 0, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.

When SLEEPDEEP = 0 and SLEEPONEXIT = 1, the MCU enters Sleep mode as soon as the system exits the lowest-priority interrupt service routine by executing the WFI instruction.

In Sleep mode, all clocks and LDO work normally except CPU clocks (stopped), and all I/O pins keep the same state as in Run mode. The LDO provides a 1.2 V power (for CPU core, memory and embedded peripherals) as it is in normal power consumption mode. The LDO output voltage is configurable by the PWC\_LDOOV register.

- (1) If the WFI is executed to enter Sleep mode, any peripheral interrupt can wake up the device from Sleep mode.
- (2) If the WFE is executed to enter Sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated by the following:
  - Enabling a peripheral interrupt (it is not enabled in the NVIC) and enabling the SEVONPEND bit. When the MCU resumes, the peripheral interrupt pending bit and NVIC channel pending bit must be cleared.
  - Configuring an internal EXINT line as an event mode to generate a wakeup event.

The wakeup time required by a WFE instruction is the shortest, since no time is wasted on interrupt entry/exit.

#### Deepsleep Mode

Deepsleep mode is entered by setting the SLEEPDEEP bit in the Cortex®-M4 system control register and clearing the LPSEL bit in the power control register before WFI or WFE instructions.

The LDO status is selected by setting the VRSEL bit in the power control register (PWC\_CTRL). When VRSEL = 0, the LDO works in normal mode. When VRSEL = 1, the LDO is set in low-power consumption mode.

In Deepsleep mode, all clocks in 1.2V domain are stopped, and both HICK and HEXT oscillators are disabled. The LDO supplies power to the 1.2V domain in normal mode or low-power mode. All I/O pins keep the same state as in Run mode. SRAM and register contents are preserved.

- (1) When the Sleep mode is entered by executing a WFI instruction, the interrupt generated on any external interrupt line in Interrupt mode can wake up the system from Deepsleep mode.
- (2) When the Sleep mode is entered by executing a WFE instruction, the interrupt generated on any external interrupt line in Event mode can wake up the system from Deepsleep mode.

When the MCU exits the Deepsleep mode, the HICK RC oscillator is enabled and selected as a system clock after stabilization. When the LDO operates in low-power mode, an additional wakeup delay is incurred for the reason that the LDO must be stabilized before the system is waken from the Deepsleep mode.

#### Standby Mode

Standby mode can achieve the lowest power consumption for the device. In this mode, the LDO is disabled. The whole 1.2V domain, PLL, HICK and HEXT oscillators are also powered off except VDD/VDDA domain. SRAM and register contents are lost.

The Standby mode is entered by the following procedures:

- Set the SLEEPDEE bit in the Cortex®-M4 system control register
- Set the LPSEL bit in the power control register (PWC\_CTRL)

- Clear the SWEF bit in the power control/status register (PWC\_CTRLSTS)
- Execute a WFI/WFE instruction

In Standby mode, all I/O pins remain in a high-impedance state except reset pins, TAMPER pins that are set as anti-tamper or calibration output, and the wakeup pins enabled.

The MCU leaves the Standby mode when an external reset (NRST pin), an WDT reset, ERTC periodic wakeup, ERTC timestamp, ERTC tamper event and a rising edge on the WKUP pin or the rising edge of an ERTC alarm event occurs.

### Debug mode

By default, the debug connection is lost if the MCU enters Deepsleep mode or Standby mode while debugging. The reason is that the Cortex®-M4 core is no longer clocked. However, the software can be debugged even in the low-power mode by setting some configuration bits in the DEBUG register (DEBUG\_CTRL).

## PWC registers

The peripheral registers must be accessed by words (32 bits).

**Table 9. PWC register map and reset values**

| Register    | Offset | Reset value |
|-------------|--------|-------------|
| PWC_CTRL    | 0x00   | 0x0000 0000 |
| PWC_CTRLSTS | 0x04   | 0x0000 0000 |
| PWC_CTRL2   | 0x20   | 0x0000 0080 |

### Power control register (PWC\_CTRL)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31:9 | Reserved | 0x000000    | resd | Kept at its default value.   |
| Bit 8    | BPWEN    | 0x0         | rw   | Battery powered domain write enable<br>0: Disabled<br>1: Enabled<br>Note: After reset, ERTC is write protected. To write, this bit must be set.                                  |
| Bit 7:5  | PVMSEL   | 0x0         | rw   | Power voltage monitoring boundary select<br>000: Unused, not configurable<br>001: 2.3 V<br>010: 2.4 V<br>011: 2.5 V<br>100: 2.6 V<br>101: 2.7 V<br>110: 2.8 V<br>111: 2.9 V      |
| Bit 4    | PVMEN    | 0x0         | rw   | Power voltage monitoring enable<br>0: Disabled<br>1: Enabled   |
| Bit 3    | CLSEF    | 0x0         | wo   | Clear SEF flag<br>0: No effect<br>1: Clear the SEF flag<br>Note: This bit is cleared by hardware after clearing the SEF flag. Reading this bit at any time will return all zero. |



| Bit   | Name   | Reset value | Type | Description  |
|-------|--------|-------------|------|--|
| Bit 2 | CLSWEF | 0x0         | wo   | Clear SWEF flag<br>0: No effect<br>1: Clear the SWEF flag<br>Note: Clear the SWEF flag after two system clock cycles. This bit is cleared by hardware after clearing the SWEF flag. Reading this bit at any time will return all zero. |
| Bit 1 | LPSEL  | 0x0         | rw   | Low power mode select when Cortex®-M4 sleepdeep<br>0: Enter DEEPSLEEP mode<br>1: Enter Standby mode  |
| Bit 0 | VRSEL  | 0x0         | rw   | LDO state select in Deepsleep mode<br>0: Enabled<br>1: Low-power consumption mode  |

#### Power control/status register (PWC\_CTRLSTS)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:15 | Reserved | 0x000000    | resd | Kept at its default value.  |
| Bit 14    | SWPEN7   | 0x0         | rw   | Standby wake-up pin7 enable<br>0: Disabled (this pin is used for general-purpose I/O)<br>1: Enabled (this pin is forced in input pull-down mode, and no longer used for general-purpose I/O)<br>Note: This bit is cleared by hardware after system reset. |
| Bit 13    | SWPEN6   | 0x0         | rw   | Standby wake-up pin6 enable<br>0: Disabled (this pin is used for general-purpose I/O)<br>1: Enabled (this pin is forced in input pull-down mode, and no longer used for general-purpose I/O)<br>Note: This bit is cleared by hardware after system reset. |
| Bit 12    | SWPEN5   | 0x0         | rw   | Standby wake-up pin5 enable<br>0: Disabled (this pin is used for general-purpose I/O)<br>1: Enabled (this pin is forced in input pull-down mode, and no longer used for general-purpose I/O)<br>Note: This bit is cleared by hardware after system reset. |
| Bit 11    | SWPEN4   | 0x0         | rw   | Standby wake-up pin4 enable<br>0: Disabled (this pin is used for general-purpose I/O)<br>1: Enabled (this pin is forced in input pull-down mode, and no longer used for general-purpose I/O)<br>Note: This bit is cleared by hardware after system reset. |
| Bit 10    | Reserved | 0x00        | resd | Kept at its default value.  |
| Bit 9     | SWPEN2   | 0x0         | rw   | Standby wake-up pin2 enable<br>0: Disabled (this pin is used for general-purpose I/O)<br>1: Enabled (this pin is forced in input pull-down mode, and no longer used for general-purpose I/O)<br>Note: This bit is cleared by hardware after system reset. |
| Bit 8     | SWPEN1   | 0x0         | rw   | Standby wake-up pin1 enable<br>0: Disabled (this pin is used for general-purpose I/O)<br>1: Enabled (this pin is forced in input pull-down mode, and no longer used for general-purpose I/O)<br>Note: This bit is cleared by hardware after system reset. |
| Bit 7:3   | Reserved | 0x00        | resd | Kept at its default value.  |

| Bit   | Name  | Reset value | Type | Description  |
|-------|-------|-------------|------|--|
| Bit 2 | PVMOF | 0x0         | ro   | Power voltage monitoring output flag<br>0: Power voltage is higher than the threshold<br>1: Power voltage is lower than the threshold<br>Note: The power voltage monitor is stopped in Standby mode.   |
| Bit 1 | SEF   | 0x0         | ro   | Standby mode entry flag<br>0: Device is not in Standby mode<br>1: Device is in Standby mode<br>Note: This bit is set by hardware (enter Standby mode) and cleared by POR/LVR or by setting the CLSEF bit.  |
| Bit 0 | SWEF  | 0x0         | ro   | Standby wake-up event flag<br>0: No wakeup event occurred<br>1: A wakeup event occurred<br>Note: This bit is set by hardware (on a wakeup event), and cleared by POR/LVR or by setting the CLSWEF bit.<br>A wakeup event is generated by one of the following:<br>When the rising edge on the Standby wakeup pin occurs;<br>When the ERTC alarm event occurs;<br>If the Standby wakeup pin is enabled when the Standby wakeup pin level is high. |

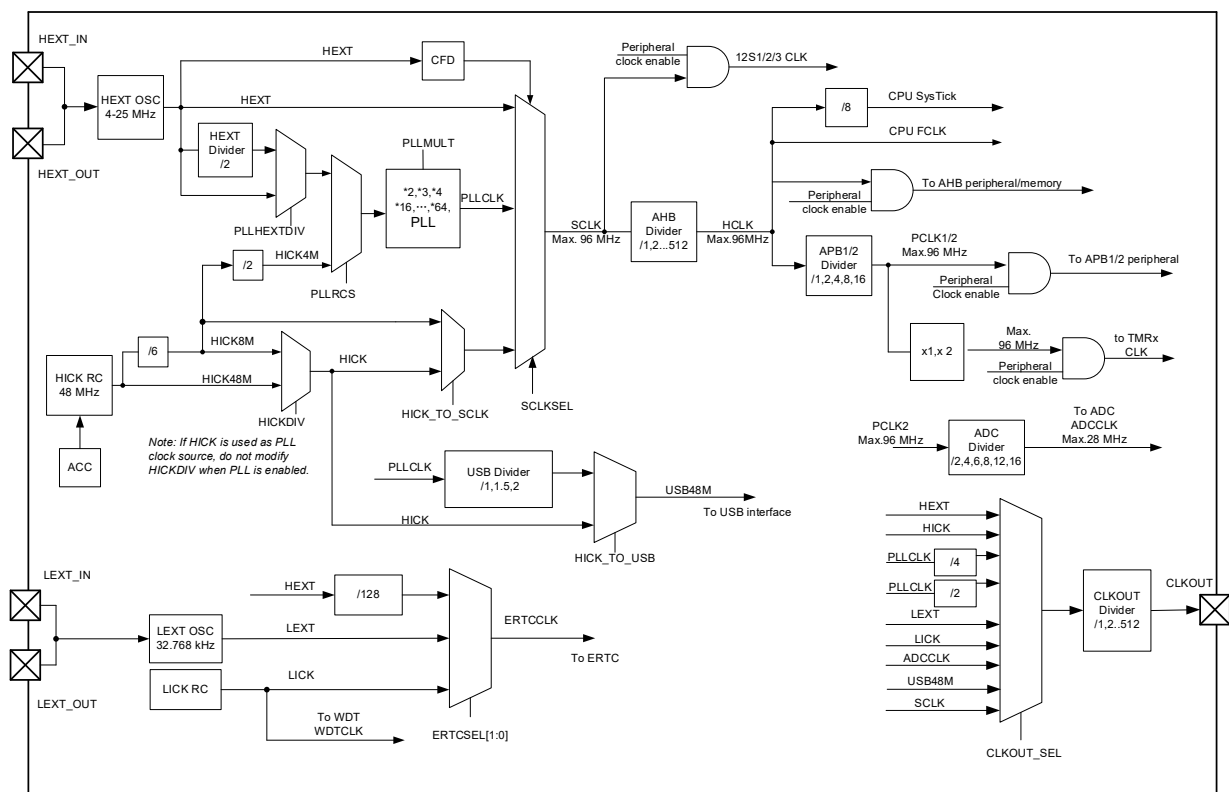
#### Power control register 2 (PWC\_CTRL2)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31:8 | Reserved | 0x000000    | resd | Kept at its default value.   |
| Bit 7:6  | Reserved | 0x2         | resd | Factory default value. Do not change.  |
| Bit 5    | VREXPEN  | 0x0         | rw   | Voltage regulator extra low power mode enable<br>This bit works with the LPSEL and VRSEL bits in the PWC_CTRL register, and it is valid only when VRSEL = 1.<br>0: Voltage regulator extra low power mode disabled<br>1: Voltage regulator extra low power mode enabled<br>Note: To enable this mode, program the VREXPEN bit before the LPSEL and VRSEL bits. |
| Bit 4:0  | Reserved | 0xXX        | resd | Factory default value. Do not change.  |

#### 4 Clock and reset manage (CRM)

## Clock

HT32F49041 series provide different clock sources, including HEXT oscillator, HICK oscillator, PLL clock, LEXT oscillator and LICK oscillator.



### Figure 11. HT32F49041 clock tree

- HEXT (high speed external crystal)
- LEXT (low speed external crystal)
- HICK (high speed internal clock)
- LICK (low speed internal clock)

AHB, APB1 and APB2 all support multiple frequency divisions, with a maximum of 96 MHz.

## Clock sources

### High speed external oscillator (HEXT)

The HEXT includes two clock sources: crystal/ceramic resonator and bypass clock.

The HEXT crystal/ceramic resonator is connected externally to a 4 ~ 25 MHz HEXT crystal that produces a highly accurate clock for the system. The HEXT clock signal is not released until it becomes stable.

An external clock source can be provided by HEXT bypass. Its frequency can be up to 25 MHz. The external clock signal should be connected to the HEXT\_IN pin while the HEXT\_OUT pin should be left floating.

### High speed internal clock (HICK)

The HICK oscillator is clocked by a high-speed RC in the microcontroller. The internal frequency of the HICK clock is 48 MHz. Although it is less accurate, its startup time is shorter than the HEXT crystal oscillator. The HICK clock frequency of each device is calibrated to  $\pm 1\%$  accuracy ( $T_A = 25^\circ\text{C}$ ) in factory. The factory calibration value is loaded in the HICKCAL[7:0] bit of the clock control register. The RC oscillator speed may be affected by voltage or temperature variations. Thus the HICK frequency can be trimmed using the HICKTRIM[5:0] bit in the clock control register.

The HICK clock signal is not released until it becomes stable.

### PLL clock

The HICK or HEXT clock can be used as an input clock source of the PLL. The PLL input clock, after being divided by a pre-divider internally, is sent to the VCO for frequency multiplication, and the VCO output frequency is output after being divided by a post-divider. At the same time, the clock after pre-divider must remain between 2 MHz and 16 MHz, and the VCO operating frequency must be kept between 500 MHz and 1000 MHz. The PLL must be configured before enabling it. The reason is that the configuration parameters cannot be changed once PLL is enabled. The PLL clock signal is not released before it becomes stable.

PLL configuration mode: regular integer frequency multiplication mode, and flexible configuration mode

#### ■ Regular integer frequency multiplication mode (default mode)

PLL clock calculation formula:

PLL output clock = PLL input clock  $\times$  PLL frequency multiplication factor

Configuration procedures:

- Clear PLLCFGGEN (RCC\_PLL[31])
- Set PLL input clock frequency, refer to PLL\_FREF (RCC\_PLL[26:24]) for details
- Set PLL frequency multiplication factor, refer to PLLMULT (RCC\_CFG[30,29], RCC\_CFG[21:18])
- Enable PLL
- Wait for PLL to be stabilized

#### ■ Flexible configuration mode

PLL clock calculation formula:

PLL output clock = PLL input clock  $\times$  PLL frequency multiplication factor / (PLL pre-frequency division factor  $\times$  PLL post-frequency division factor)

$500 \text{ MHz} \leq \text{PLL input clock} \times \text{PLL frequency multiplication factor} / \text{PLL pre-frequency division factor} \leq 1000 \text{ MHz}$

$2 \text{ MHz} \leq \text{PLL input clock} / \text{PLL pre-frequency division factor} \leq 16 \text{ MHz}$

Configuration procedures:

- Set the PLL\_NS, PLL\_MS and PLL\_FR bits of the RCC\_PLL register respectively using the calculated PLL frequency multiplication factor, PLL pre-frequency division factor and PLL post-frequency division factor
- Enable PLLCFGEN (RCC\_PLL[31])
- Enable PLL
- Wait for PLL to be stabilized

Note: 1. The PLL\_FREF and PLL\_MULT bits are not available in flexible configuration mode.

2. In flexible configuration mode, non-integer frequency multiplication is supported.

Examples:

When PLL input clock is 12.288 MHz, PLL output frequency is equal to  $12.288 \times 125 / (2 \times 8) = 96 \text{ MHz}$

When PLL input clock is 5 MHz, PLL output frequency is equal to  $5 \times 108 / (5 \times 1) = 108 \text{ MHz}$

#### Low speed external oscillator (LEXT)

The LEXT oscillator provides two clock sources: LEXT crystal/ceramic resonator and LEXT bypass.

LEXT crystal/ceramic resonator:

The LEXT crystal/ceramic resonator provides a 32.768 KHz low-speed clock source. The LEXT clock signal is not released before it becomes stable.

#### LEXT bypass clock

In this mode, an external clock source with a frequency of 32.768 kHz can be provided. The external clock signal should be connected to the LEXT\_IN pin while the LEXT\_OUT can be released for GPI control.

#### Low speed internal RC oscillator (LICK)

The LICK oscillator is clocked by an internal low-speed RC oscillator. The clock frequency is between 30 kHz and 60 kHz. It acts as a low-power clock source that can be kept running in DeepSleep mode and Standby mode for watchdog and auto-wakeup unit.

The LICK clock signal is not released before it becomes stable.

## System clock

After a system reset, the HICK oscillator is selected as system clock. The system clock can make flexible switch among HICK oscillator, HEXT oscillator and PLL clock. However, a switch from one clock source to another occurs only if the target clock source becomes stable. When the HICK oscillator is used directly or indirectly through the PLL as the system clock, it cannot be stopped.

## Peripheral clock

Most peripherals use HCLK, PCLK1 or PCLK2 clock. The individual peripherals have their dedicated clocks.

System Tick timer (SysTick) is clocked by CPU FCLK (HCLK) or CPU systick (HCLK/8).

ADC is clocked by APB2 divided by 2, 4, 6, 8, 12, 16.

The timers are clocked by APB1/2. In particular, if the APB prescaler is 1, the timer clock frequency is equal to that of APB1/2; otherwise, the timer clock frequency doubles that of the APB1/2 frequency.

The USB clock source can be switched between HICK and PLL frequency divider. If the HICK is selected as a clock source, the USB clock should be set as 48 MHz; If the PLL frequency divider is selected as a clock source, the USB frequency divider provides 48 MHz USBCLK, and thus the PLL must be set as  $48 \times N \times 0.5$  MHz ( $N = 2, 3, 4, 5 \dots$ )

ERTC clock sources: divided HEXT oscillator, LEXT oscillator and LICK oscillator. Once the clock source is selected, it cannot be altered without resetting the battery powered domain. If the LEXT is used as an ERTC clock, the ERTC is not affected when the VDD is powered off. If the HEXT or LICK is selected as an ERTC clock, the ERTC state is not guaranteed when both HEXT and LICK are powered off.

Watchdog is clocked by LICK oscillator. If the watchdog is enabled by either hardware option or software access, the LICK oscillator is forced ON. The clock is provided to the watchdog only after the LICK oscillator temporization.

## Clock fail detector

The clock fail detector (CFD) is designed to respond to HEXT clock failure when the HEXT is used as a system clock, directly or indirectly. If a failure is detected on the HEXT clock, a clock failure event is sent to the brake input of TMR1 and an interrupt is generated. This interrupt is directly linked to CPU NMI so that the software can perform rescue operations. The NMI interrupt keeps executing until the CFD interrupt pending bit is cleared. This is why the CFD interrupt has to be cleared in the NMI service routine. The HEXT clock failure will result in a switch of the system clock to the HICK clock, the CFD to be disabled, HEXT clock to be stopped, and even PLL to be disabled if the HEXT clock is selected as the system clock through PLL.

## Clock output

The microcontroller allows the internal clock signal to be output to external CLKOUT pins. That is, ADCCLK, USB48M, SCLK, LICK, LEXT, HICK, HEXT, PLLCLK/2 and PLLCLK/4 can be used as CLKOUT clocks. When being used as the CLKOUT clock output pin, the corresponding GPIO port registers must be configured accordingly.

## Interrupts

The microcontroller specifies a stable flag for each clock source. As a result, when a clock source is enabled, it is possible to determine if the clock is stable by checking the flag pertaining to the clock source. An interrupt request is generated when the interrupt corresponding to the clock source is enabled.

If a failure is detected on the HEXT clock, the CFD interrupt is generated. Such interrupt is directly linked to CPU NMI.

## Reset

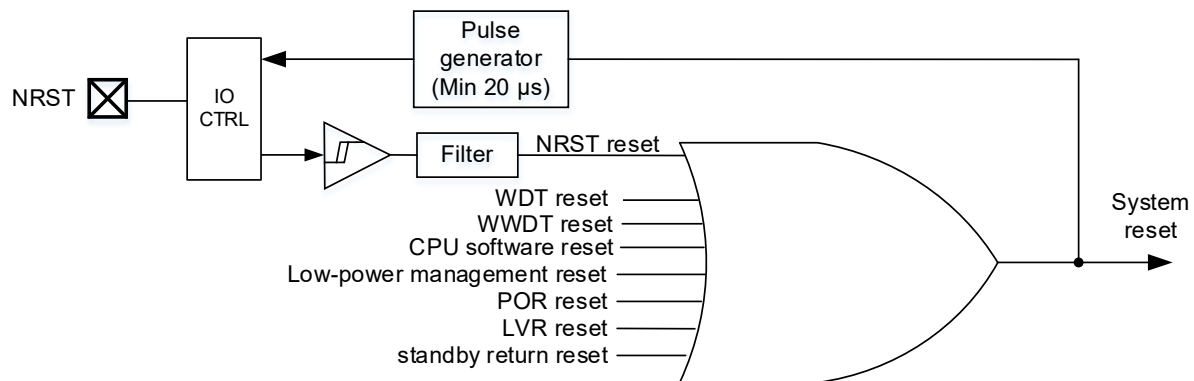
### System reset

HT32F49041 series provide the following system reset sources:

- NRST reset: on the external NRST pin
- WDT reset: watchdog overflow

- WWDT reset: window watchdog overflow
- CPU software reset: Cortex®-M4 software reset
- Low-power management reset: This reset is enabled when entering Standby mode by clearing the nSTDBY\_RST bit in the user system data area; The reset is also triggered when entering Deepsleep mode by clearing the nDEPSLP\_RST in the user system data area.
- POR: power-on reset
- LVR: low voltage reset
- When exiting Standby mode

NRST reset, WDT reset, WWDT reset, software reset and low-power management reset sets all registers to their reset values except the clock control/status register (CRM\_CTRLSTS) and the battery powered domain; the power-on reset, low-voltage reset or reset generated when exiting Standby mode sets all registers to their reset values except the battery powered domain registers.



**Figure 12. System reset circuit**

## Battery powered domain reset

Battery powered domain has two specific reset sources:

- Software reset: triggered by setting the BPDRST bit in the battery powered domain control register (CRM\_BPDC).
- VDD power on, if VDD has been powered off.

Software reset affects only the battery powered domain.

## CRM registers

These peripheral registers have to be accessed by bytes (8 bits), half words (16 bits) or words (32 bits).

**Table 10. CRM register map and reset values**

| Register    | Offset | Reset value |
|-------------|--------|-------------|
| CRM_CTRL    | 0x000  | 0x0000 XX83 |
| CRM_CFG     | 0x004  | 0x0000 0000 |
| CRM_CLKINT  | 0x008  | 0x0000 0000 |
| CRM_APB2RST | 0x00C  | 0x0000 0000 |

| Register        | Offset | Reset value |
|-----------------|--------|-------------|
| CRM_APB1RST     | 0x010  | 0x0000 0000 |
| CRM_AHBEN       | 0x014  | 0x0000 0014 |
| CRM_APB2EN      | 0x018  | 0x0000 0000 |
| CRM_APB1EN      | 0x01C  | 0x0000 0000 |
| CRM_BPDC        | 0x020  | 0x0000 0000 |
| CRM_CTRLSTS     | 0x024  | 0x0C00 0000 |
| CRM_AHBRST      | 0x028  | 0x0000 0000 |
| CRM_PLL         | 0x02C  | 0x0000 1F10 |
| CRM_MISC1       | 0x030  | 0x0000 0000 |
| CRM_OTG_EXTCTRL | 0x044  | 0x0000 0000 |
| CRM_MISC2       | 0x054  | 0x0000 000D |

### Clock control register (CRM\_CTRL)

| Bit       | Name      | Reset value | Type | Description   |
|-----------|-----------|-------------|------|---|
| Bit 30:26 | Reserved  | 0x00        | resd | Kept at its default value.  |
| Bit 25    | PLLSTBL   | 0x0         | ro   | PLL clock stable<br>This bit is set by hardware after PLL is ready.<br>0: PLL clock is not ready.<br>1: PLL clock is ready.   |
| Bit 24    | PLEN      | 0x0         | rw   | PLL enable<br>This bit is set and cleared by software. It can also be cleared by hardware when entering Standby or Deepsleep mode. When the PLL clock is used as the system clock, this bit cannot be cleared.<br>0: PLL is OFF<br>1: PLL is ON           |
| Bit 23:20 | Reserved  | 0x0         | resd | Kept at its default value.  |
| Bit 19    | CFDEN     | 0x0         | rw   | Clock failure detector enable<br>0: OFF<br>1: ON  |
| Bit 18    | HEXTBYPSS | 0x0         | rw   | High speed external crystal bypass<br>This bit can be written only if the HEXT is disabled.<br>0: OFF<br>1: ON  |
| Bit 17    | HEXTSTBL  | 0x0         | ro   | High speed external crystal stable<br>This bit is set by hardware after HEXT becomes stable.<br>0: HEXT is not ready.<br>1: HEXT is ready.  |
| Bit 16    | HEXTEN    | 0x0         | rw   | High speed external crystal enable<br>This bit is set and cleared by software. It can also be cleared by hardware when entering Standby or Deepsleep mode. When the HEXT clock is used as the system clock, this bit cannot be cleared<br>0: OFF<br>1: ON |



| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:8 | HICKCAL  | 0xXX        | rw   | High speed internal clock calibration<br>The default value of this field is the initial factory calibration value.<br>When the HICK output frequency is 48 MHz, it needs adjust 240 kHz (design value) based on this frequency for each HICKCAL value change; when HICK output frequency is 8 MHz (design value), it needs adjust 40 kHz based on this frequency for each HICKCAL value change.<br>Note: This bit can be written only if the HICKCAL_KEY[7:0] is set as 0x5A. |
| Bit 7:2  | HICKTRIM | 0x20        | rw   | High speed internal clock trimming<br>These bits work with the HICKCAL[7:0] to determine the HICK oscillator frequency. The default value is 32, which can trim the HICK to be $\pm 1\%$ .  |
| Bit 1    | HICKSTBL | 0x1         | ro   | High speed internal clock stable<br>This bit is set by hardware after the HICK is ready.<br>0: Not ready<br>1: Ready  |
| Bit 0    | HICKEN   | 0x1         | rw   | High speed internal clock enable<br>This bit is set and cleared by software. It can also be set by hardware when exiting Standby or Deepsleep mode. When a HEXT clock failure occurs. This bit can also be set. When the HICK is used as the system clock, this bit cannot be cleared.<br>0: Disabled<br>1: Enabled   |

### Clock configuration register (CRM\_CFG)

Access: 0 to 2 wait states.

1 or 2 wait states are inserted only when the access occurs during a clock source switch.

| Bit       | Name       | Reset value | Type | Description  |
|-----------|------------|-------------|------|--|
| Bit 31    | Reserved   | 0           | resd | Kept at its default value.   |
| Bit 26:24 | CLKOUT_SEL | 0x0         | rw   | Clock output selection<br>CLKOUT_SEL[3] is the bit 16 of the CRM_MISC1 register.<br>0000: None<br>0001: Reserved<br>0010: LICK<br>0011: LEXT<br>0100: SCLK<br>0101: HICK<br>0110: HEXT<br>0111: PLL/2<br>1100: PLL/4<br>1101: USB<br>1110: ADC |

| Bit                    | Name      | Reset value | Type | Description   |
|------------------------|-----------|-------------|------|---|
| Bit 27<br>Bit 23:22    | USBDIV    | 0x0         | rw   | USB division<br>The PLL clock after division is used as USB clock.<br>000: PLL/1.5<br>001: Forbidden<br>010: PLL/2.5<br>011: PLL/2<br>100: PLL/3.5<br>101: PLL/3<br>110: PLL/4<br>111: PLL/4  |
| Bit 30:29<br>Bit 21:18 | PLLMULT   | 0x00        | rw   | PLL multiplication factor<br>000000: PLL × 2    000001: PLL × 3<br>000010: PLL × 4    000011: PLL × 5<br>.....<br>001100: PLL × 14    001101: PLL × 15<br>001110: PLL × 16    001111: PLL × 16<br>010000: PLL × 17    010001: PLL × 18<br>010010: PLL × 19    010011: PLL × 20<br>.....<br>111110: PLL × 63    111111: PLL × 64 |
| Bit 17                 | PLHEXTDIV | 0           | rw   | HEXT division selection for PLL entry clock<br>0: Forbidden<br>1: HEXT/2  |
| Bit 16                 | PLLRCSC   | 0           | rw   | PLL reference clock select<br>0: HICK-divided clock (4 MHz)<br>1: HEXT clock  |
| Bit 28<br>Bit 15:14    | ADCDIV    | 0x0         | rw   | ADC division<br>The PCLK that is divided by the following factors serves the ADC.<br>000: PCLK/2<br>001: PCLK/4<br>010: PCLK/6<br>011: PCLK/8<br>100: PCLK/2<br>101: PCLK/12<br>110: PCLK/8<br>111: PCLK/16   |
| Bit 13:11              | APB2DIV   | 0x0         | rw   | APB2 division<br>The divided HCLK is used as APB2 clock.<br>0xx: not divided<br>100: divided by 2<br>101: divided by 4<br>110: divided by 8<br>111: divided by 16<br><br>Note: The software must set these bits correctly to ensure that the APB2 clock frequency does not exceed 96 MHz.                                       |

| Bit      | Name    | Reset value | Type | Description  |
|----------|---------|-------------|------|--|
| Bit 10:8 | APB1DIV | 0x0         | rw   | APB1 division<br>The divided HCLK is used as APB1 clock.<br>0xx: not divided<br>100: divided by 2<br>101: divided by 4<br>110: divided by 8<br>111: divided by 16<br>Note: The software must set these bits correctly to ensure that the APB1 clock frequency does not exceed 96 MHz                             |
| Bit 7:4  | AHBDIV  | 0x0         | rw   | AHB division<br>The divided SCLK is used as AHB clock.<br>0xxx: SCLK not divided<br>1000: SCLK divided by 2    1100: SCLK divided by 64<br>1001: SCLK divided by 4    1101: SCLK divided by 128<br>1010: SCLK divided by 8    1110: SCLK divided by 256<br>1011: SCLK divided by 16    1111: SCLK divided by 512 |
| Bit 3:2  | SCLKSTS | 0x0         | R0   | System clock select status<br>00: HICK<br>01: HEXT<br>10: PLL<br>11: Reserved. Kept at its default value.  |
| Bit 1:0  | SCLKSEL | 0x0         | rw   | System clock select<br>00: HICK<br>01: HEXT<br>10: PLL<br>11: Reserved. Kept at its default value.   |

### Clock interrupt register (CRM\_CLKINT)

Access: 0 wait state, accessible by words, half-words and bytes.

| Bit       | Name       | Reset value | Type | Description  |
|-----------|------------|-------------|------|--|
| Bit 31:24 | Reserved   | 0x00        | resd | Kept at its default value.   |
| Bit 23    | CFDFC      | 0           | wo   | Clock failure detection flag clear<br>Writing 1 by software to clear CFDF.<br>0: No effect<br>1: Clear |
| Bit 22:21 | Reserved   | 0x0         | resd | Kept at its default value.   |
| Bit 20    | PLLSTBLFC  | 0           | wo   | PLL stable flag clear<br>Writing 1 by software to clear PLLSTBLF.<br>0: No effect<br>1: Clear          |
| Bit 19    | HEXTSTBLFC | 0           | wo   | HEXT stable flag clear<br>Writing 1 by software to clear HEXTSTBLF.<br>0: No effect<br>1: Clear        |
| Bit 18    | HICKSTBLFC | 0           | wo   | HICK stable flag clear<br>Writing 1 by software to clear HICKSTBLF.<br>0: No effect<br>1: Clear        |

| Bit       | Name        | Reset value | Type | Description  |
|-----------|-------------|-------------|------|--|
| Bit 17    | LEXTSTBLFC  | 0           | wo   | LEXT stable flag clear<br>Writing 1 by software to clear LEXTSTBLF.<br>0: No effect<br>1: Clear  |
| Bit 16    | LICKSTBLFC  | 0           | wo   | LICK stable flag clear<br>Writing 1 by software to clear LICKSTBLF.<br>0: No effect<br>1: Clear  |
| Bit 15:13 | Reserved    | 0x0         | resd | Kept at its default value.   |
| Bit 12    | PLLSTBLIEN  | 0           | rw   | PLL stable interrupt enable<br>0: Disabled<br>1: Enabled   |
| Bit 11    | HEXTSTBLIEN | 0           | rw   | HEXT stable interrupt enable<br>0: Disabled<br>1: Enabled  |
| Bit 10    | HICKSTBLIEN | 0x0         | rw   | HICK stable interrupt enable<br>0: Disabled<br>1: Enabled  |
| Bit 9     | LEXTSTBLIEN | 0           | rw   | LEXT stable interrupt enable<br>0: Disabled<br>1: Enabled  |
| Bit 8     | LICKSTBLIEN | 0           | rw   | LICK stable interrupt enable<br>0: Disabled<br>1: Enabled  |
| Bit 7     | CFDF        | 0           | ro   | Clock Failure Detection flag<br>This bit is set by hardware when the HEXT clock failure occurs.<br>0: No clock failure<br>1: Clock failure |
| Bit 6:5   | Reserved    | 0x0         | resd | Keep at its default value.   |
| Bit 4     | PLLSTBLF    | 0           | ro   | PLL stable flag<br>Set by hardware.<br>0: PLL is not ready.<br>1: PLL is ready.  |
| Bit 3     | HEXTSTBLF   | 0           | ro   | HEXT stable flag<br>Set by hardware.<br>0: HEXT is not ready.<br>1: HEXT is ready.   |
| Bit 2     | HICKSTBLF   | 0           | ro   | HICK stable flag<br>Set by hardware.<br>0: HICK is not ready.<br>1: HICK is ready.   |
| Bit 1     | LEXTSTBLF   | 0           | ro   | LEXT stable flag<br>Set by hardware.<br>0: LEXT is not ready.<br>1: LEXT is ready.   |
| Bit 0     | LICKSTBLF   | 0           | ro   | LICK stable interrupt flag<br>Set by hardware.<br>0: LICK is not ready.<br>1: LICK is ready.   |

## APB2 peripheral reset register (CRM\_APB2RST)

Access: 0 wait state, accessible by words, half-words and bytes.

| Bit       | Name      | Reset value | Type | Description   |
|-----------|-----------|-------------|------|---|
| Bit 31:19 | Reserved  | 0x00        | resd | Kept at its default value.                                  |
| Bit 18    | TMR17RST  | 0           | rw   | TMR17 reset<br>0: Does not reset TMR17<br>1: Reset TMR17    |
| Bit 17    | TMR16RST  | 0           | rw   | TMR16 reset<br>0: Does not reset TMR16<br>1: Reset TMR16    |
| Bit 16    | TMR15RST  | 0           | rw   | TMR15 reset<br>0: Does not reset TMR15<br>1: Reset TMR15    |
| Bit 15    | Reserved  | 0x0         | resd | Kept at its default value.                                  |
| Bit 15    | SDIO1RST  | 0x0         | rw   | SDIO1 reset<br>0: Does not reset SDIO1<br>1: Reset SDIO1    |
| Bit 14    | USART1RST | 0           | rw   | USART1 reset<br>0: Does not reset USART1<br>1: Reset USART1 |
| Bit 13    | Reserved  | 0x0         | resd | Kept at its default value.                                  |
| Bit 12    | SPI1RST   | 0x0         | rw   | SPI1 reset<br>0: Does not reset SPI1<br>1: Reset SPI1       |
| Bit 11    | TMR1RST   | 0           | rw   | TMR1 reset<br>0: Does not reset TMR1<br>1: Reset TMR1       |
| Bit 10    | Reserved  | 0x0         | resd | Kept at its default value.                                  |
| Bit 9     | ADC1RST   | 0           | rw   | ADC1 reset<br>0: Does not reset ADC1<br>1: Reset ADC1       |
| Bit 8:2   | Reserved  | 0x0         | resd | Kept at its default value.                                  |
| Bit 1     | EXINTRST  | 0           | rw   | EXINT reset<br>0: Does not reset EXINT<br>1: Reset EXINT    |
| Bit 0     | SCFGRST   | 0           | rw   | SCFG reset<br>0: Does not reset SCFG<br>1: Reset SCFG       |

## APB1 peripheral reset register (CRM\_APB1RST)

Access: 0 wait state, accessible by words, half-words and bytes.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:29 | Reserved | 0x0         | resd | Kept at its default value.                         |
| Bit 28    | PWCRST   | 0           | rw   | PWC reset<br>0: Does not reset PWC<br>1: Reset PWC |
| Bit 27    | ACCRST   | 0           | rw   | ACC reset<br>0: Does not reset ACC<br>1: Reset ACC |

| Bit       | Name      | Reset value | Type | Description  |
|-----------|-----------|-------------|------|--|
| Bit 26    | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 25    | CAN1RST   | 0x0         | rw   | CAN1 reset<br>0: Does not reset CAN1<br>1: Reset CAN1  |
| Bit 24:23 | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 22    | I2C2RST   | 0           | rw   | I <sup>2</sup> C2 reset<br>0: Does not reset I <sup>2</sup> C2<br>1: Reset I <sup>2</sup> C2 |
| Bit 21    | I2C1RST   | 0           | rw   | I <sup>2</sup> C1 reset<br>0: Does not reset I <sup>2</sup> C1<br>1: Reset I <sup>2</sup> C1 |
| Bit 21    | EDMARST   | 0x0         | rw   | EDMA reset<br>0: Does not reset EDMA<br>1: Reset EDMA  |
| Bit 20    | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 19    | USART4RST | 0           | rw   | USART4 reset<br>0: Does not reset USART4<br>1: Reset USART4                                  |
| Bit 18    | USART3RST | 0           | rw   | USART3 reset<br>0: Does not reset USART3<br>1: Reset USART3                                  |
| Bit 17    | USART2RST | 0           | rw   | USART2 reset<br>0: Does not reset USART2<br>1: Reset USART2                                  |
| Bit 16    | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 15    | SPI3RST   | 0           | rw   | SPI3 reset<br>0: Does not reset SPI3<br>1: Reset SPI3  |
| Bit 14    | SPI2RST   | 0           | rw   | SPI2 reset<br>0: Does not reset SPI2<br>1: Reset SPI2  |
| Bit 13:12 | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 11    | WWDTRST   | 0           | rw   | WWDT reset<br>0: Does not reset WWDT<br>1: Reset WWDT  |
| Bit 10:9  | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 8     | TMR14RST  | 0           | rw   | TMR14 reset<br>0: Does not reset TMR14<br>1: Reset TMR14                                     |
| Bit 7     | TMR13RST  | 0           | rw   | TMR13 reset<br>0: Does not reset TMR13<br>1: Reset TMR13                                     |
| Bit 6     | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 5     | TMR7RST   | 0           | rw   | TMR7 reset<br>0: Does not reset TMR7<br>1: Reset TMR7  |
| Bit 4     | TMR6RST   | 0           | rw   | TMR6 reset<br>0: Does not reset TMR6<br>1: Reset TMR6  |
| Bit 3:2   | Reserved  | 0x0         | resd | Kept at its default value.   |

| Bit   | Name    | Reset value | Type | Description   |
|-------|---------|-------------|------|---|
| Bit 1 | TMR3RST | 0           | rw   | TMR3 reset<br>0: Does not reset TMR3<br>1: Reset TMR3 |
| Bit 0 | TMR2RST | 0           | rw   | TMR2 reset<br>0: Does not reset TMR2<br>1: Reset TMR2 |

## APB peripheral clock enable register (CRM\_AHBEN)

Access: by words, half-words and bytes.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:23 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 22    | GPIOFEN  | 0           | rw   | GPIOF clock enable<br>0: Disabled<br>1: Enabled   |
| Bit 21    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 20    | GPIODEN  | 0           | rw   | GPIOD clock enable<br>0: Disabled<br>1: Enabled   |
| Bit 19    | GPIOCEN  | 0           | rw   | GPIOC clock enable<br>0: Disabled<br>1: Enabled   |
| Bit 18    | GPIOBEN  | 0           | rw   | GPIOB clock enable<br>0: Disabled<br>1: Enabled   |
| Bit 17    | GPIOAEN  | 0           | rw   | GPIOA clock enable<br>0: Disabled<br>1: Enabled   |
| Bit 16:13 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 12    | OTGFS1EN | 0           | rw   | OTGFS1 clock enable<br>0: Disabled<br>1: Enabled  |
| Bit 11:7  | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 6     | CRCEN    | 0           | rw   | CRC clock enable<br>0: Disabled<br>1: Enabled   |
| Bit 5     | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 4     | FLASHEN  | 0           | rw   | FLASH clock enable<br>This bit is used to enable Flash clock in Sleep or Deepsleep mode.<br>0: Disabled<br>1: Enabled |
| Bit 3     | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 2     | SRAMEN   | 1           | rw   | SRAM clock enable<br>This bit is used to enable SRRM clock in Sleep or Deepsleep mode.<br>0: Disabled<br>1: Enabled   |
| Bit 1     | Reserved | 0x0         | resd | Kept at its default value.  |

| Bit   | Name   | Reset value | Type | Description                                    |
|-------|--------|-------------|------|--|
| Bit 0 | DMA1EN | 0x0         | rw   | DMA1 clock enable<br>0: Disabled<br>1: Enabled |

## APB2 peripheral clock enable register (CRM\_AHB2EN)

Access: by words, half-words and bytes.

When accessing peripherals on the APB2, wait states are inserted until the end of the peripheral access on APB2 bus.

| Bit       | Name     | Reset value | Type | Description                                      |
|-----------|----------|-------------|------|--|
| Bit 31:19 | Reserved | 0x00        | resd | Kept at its default value.                       |
| Bit 18    | TMR17EN  | 0           | rw   | TMR17 clock enable<br>0: Disabled<br>1: Enabled  |
| Bit 17    | TMR16EN  | 0           | rw   | TMR16 clock enable<br>0: Disabled<br>1: Enabled  |
| Bit 16    | TMR15EN  | 0           | rw   | TMR15 clock enable<br>0: Disabled<br>1: Enabled  |
| Bit 15    | Reserved | 0x0         | resd | Kept at its default value.                       |
| Bit 14    | USART1EN | 0           | rw   | USART1 clock enable<br>0: Disabled<br>1: Enabled |
| Bit 13    | Reserved | 0x0         | resd | Kept at its default value.                       |
| Bit 12    | SPI1EN   | 0           | rw   | SPI1 clock enable<br>0: Disabled<br>1: Enabled   |
| Bit 11    | TMR1EN   | 0           | rw   | TMR1 clock enable<br>0: Disabled<br>1: Enabled   |
| Bit 10    | Reserved | 0x0         | resd | Kept at its default value.                       |
| Bit 9     | ADC1EN   | 0           | rw   | ADC1 clock enable<br>0: Disabled<br>1: Enabled   |
| Bit 8:1   | Reserved | 0x0         | resd | Kept at its default value.                       |
| Bit 0     | SCFGEN   | 0           | rw   | SCFG clock enable<br>0: Disabled<br>1: Enabled   |



## APB1 peripheral clock enable register (CRM\_AHB1EN)

Access: 0 wait state, accessible by words, half-words and bytes.

No-wait states in most cases. However, when accessing to peripherals on APB1, wait-states are inserted until the end of peripheral access on the APB1 bus.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:29 | Reserved | 0x0         | resd | Kept at its default value.                                  |
| Bit 28    | PWCEN    | 0           | rw   | PWC clock enable<br>0: Disabled<br>1: Enabled               |
| Bit 27    | ACCEN    | 0           | rw   | ACC clock enable<br>0: Disabled<br>1: Enabled               |
| Bit 26    | Reserved | 0x0         | resd | Kept at its default value.                                  |
| Bit 25    | CAN1EN   | 0           | rw   | CANS1 clock enable<br>0: Disabled<br>1: Enabled             |
| Bit 24:23 | Reserved | 0x0         | resd | Kept at its default value.                                  |
| Bit 22    | I2C2EN   | 0           | rw   | I <sup>2</sup> C2 clock enable<br>0: Disabled<br>1: Enabled |
| Bit 21    | I2C1EN   | 0           | rw   | I <sup>2</sup> C1 clock enable<br>0: Disabled<br>1: Enabled |
| Bit 20    | Reserved | 0x0         | resd | Kept at its default value.                                  |
| Bit 19    | USART4EN | 0           | rw   | USART4 clock enable<br>0: Disabled<br>1: Enabled            |
| Bit 18    | USART3EN | 0           | rw   | USART3 clock enable<br>0: Disabled<br>1: Enabled            |
| Bit 18    | USART2EN | 0           | rw   | USART2 clock enable<br>0: Disabled<br>1: Enabled            |
| Bit 16    | Reserved | 0x0         | resd | Kept at its default value.                                  |
| Bit 15    | SPI3EN   | 0           | rw   | SPI3 clock enable<br>0: Disabled<br>1: Enabled              |
| Bit 14    | SPI2EN   | 0           | rw   | SPI2 clock enable<br>0: Disabled<br>1: Enabled              |
| Bit 13:12 | Reserved | 0x0         | resd | Kept at its default value.                                  |
| Bit 11    | WWDTEN   | 0           | rw   | WWDT clock enable<br>0: Disabled<br>1: Enabled              |
| Bit 10:9  | Reserved | 0x0         | resd | Kept at its default value.                                  |
| Bit 8     | TMR14EN  | 0           | rw   | TMR14 clock enable<br>0: Disabled<br>1: Enabled             |

| Bit     | Name     | Reset value | Type | Description                                     |
|---------|----------|-------------|------|---|
| Bit 7   | TMR13EN  | 0           | rw   | TMR13 clock enable<br>0: Disabled<br>1: Enabled |
| Bit 6   | Reserved | 0x0         | resd | Kept at its default value.                      |
| Bit 5   | TMR7EN   | 0           | rw   | TMR7 clock enable<br>0: Disabled<br>1: Enabled  |
| Bit 4   | TMR6EN   | 0           | rw   | TMR6 clock enable<br>0: Disabled<br>1: Enabled  |
| Bit 3:2 | Reserved | 0x0         | resd | Kept at its default value.                      |
| Bit 1   | TMR3EN   | 0           | rw   | TMR3 clock enable<br>0: Disabled<br>1: Enabled  |
| Bit 0   | TMR2EN   | 0           | rw   | TMR2 clock enable<br>0: Disabled<br>1: Enabled  |

### Battery powered domain control register (CRM\_BPDC)

Access: 0 to 3 wait states, accessible by words, half-words or bytes. Wait states are inserted in the case of consecutive accesses to this register.

Note: LEXTEN, LEXTBYPSS, ERTCSEL, and ERTCEN bits of the battery powered domain control register CRM\_BDC) are in the battery powered domain. As a result, these bits are write protected after reset, and can only be modified by setting the BPWEN bit in the power control register (PWR\_CTRL). These bits could be reset only by battery powered domain reset. Any internal or external reset does not affect these bits.

| Bit       | Name      | Reset value | Type | Description  |
|-----------|-----------|-------------|------|--|
| Bit 31:17 | Reserved  | 0x0000      | resd | Kept at its default value.   |
| Bit 16    | BPDRST    | 0x0         | rw   | Battery powered domain software reset<br>0: Do not reset battery powered domain software<br>1: Reset battery powered domain software   |
| Bit 15    | ERTCEN    | 0x0         | rw   | ERTC clock enable<br>Set and cleared by software.<br>0: Disabled<br>1: Enabled   |
| Bit 14:10 | Reserved  | 0x00        | resd | Kept at its default value.   |
| Bit 9:8   | ERTCSEL   | 0x0         | rw   | ERTC clock selection<br>Once the ERTC clock source is selected, it cannot be changed until the BPDRST bit is reset.<br>00: No clock<br>01: LEXT<br>10: LICK<br>11: Divided HEXT (with the ERTC_DIV bit in the CRM_CFG) |
| Bit 7:3   | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 2     | LEXTBYPSS | 0           | rw   | Low speed external crystal bypass<br>0: Disabled<br>1: Enabled   |

| Bit   | Name     | Reset value | Type | Description  |
|-------|----------|-------------|------|--|
| Bit 1 | LEXTSTBL | 0           | ro   | Low speed external oscillator stable<br>Set by hardware after the LEXT is ready.<br>0: LEXT is not ready.<br>1: LEXT is ready. |
| Bit 0 | LEXTEN   | 0           | rw   | External low-speed oscillator enable<br>0: Disabled<br>1: Enabled  |

### Control/status register (CRM\_CTRLSTS)

Reset flag can only be cleared by power reset or by setting the RSTFC bit, while others are cleared by system reset.

Access: 0 to 3 wait states, accessible by words, half-words or bytes. Wait states are inserted in the case of consecutive accesses to this register.

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31   | LPRSTF   | 0           | ro   | Low-power reset flag<br>Set by hardware. Cleared by writing to the RSTFC bit.<br>0: No low-power reset occurs<br>1: Low-power reset occurs                                     |
| Bit 30   | WWDTRSTF | 0           | ro   | Window watchdog timer reset flag<br>Set by hardware. Cleared by writing to the RSTFC bit.<br>0: No window watchdog timer reset occurs<br>1: Window watchdog timer reset occurs |
| Bit 29   | WDTRSTF  | 0           | ro   | Watchdog timer reset flag<br>Set by hardware. Cleared by writing to the RSTFC bit.<br>0: No watchdog timer reset occurs<br>1: Watchdog timer reset occurs.                     |
| Bit 28   | SWRSTF   | 0           | ro   | Software reset flag<br>Set by hardware. Cleared by writing to the RSTFC bit.<br>0: No software reset occurs<br>1: Software reset occurs.                                       |
| Bit 27   | PORRSTF  | 1           | ro   | POR/LVR reset flag<br>Set by hardware. Cleared by writing to the RSTFC bit.<br>0: No POR/LVR reset occurs<br>1: POR/LVR reset occurs.  |
| Bit 26   | NRSTF    | 1           | rw   | NRST pin reset flag<br>Set by hardware. Cleared by writing to the RSTFC bit.<br>0: No NRST pin reset occurs<br>1: NRST pin reset occurs  |
| Bit 25   | Reserved | 0           | resd | Kept at its default value.   |
| Bit 24   | RSTFC    | 0           | rw   | Reset flag clear<br>Cleared by writing 1 through software.<br>0: No effect<br>1: Clear the reset flag.   |
| Bit 23:2 | Reserved | 0x000000    | resd | Kept at its default value.   |
| Bit 1    | LICKSTBL | 0           | ro   | LICK stable<br>0: LICK is not ready.<br>1: LICK is ready.  |
| Bit 0    | LICKEN   | 0           | rw   | LICK enable<br>0: Disabled<br>1: Enabled   |

## APB peripheral reset register (CRM\_APBRSR)

Access: 0 wait state, accessible by words, half-words and bytes.

| Bit       | Name      | Reset value | Type | Description   |
|-----------|-----------|-------------|------|---|
| Bit 31:23 | Reserved  | 0x0000      | resd | Kept at its default value.                                  |
| Bit 22    | GPIOFRST  | 0           | rw   | GPIOF reset<br>0: Does not reset GPIOF<br>1: Reset GPIOF    |
| Bit 21    | Reserved  | 0x0         | resd | Kept at its default value.                                  |
| Bit 20    | GPIODRST  | 0           | rw   | GPIOD reset<br>0: Does not reset GPIOD<br>1: Reset GPIOD    |
| Bit 19    | GPIOCRST  | 0           | rw   | GPIOC reset<br>0: Does not reset GPIOC<br>1: Reset GPIOC    |
| Bit 18    | GPIOBRST  | 0           | rw   | GPIOB reset<br>0: Does not reset GPIOB<br>1: Reset GPIOB    |
| Bit 17    | GPIOARST  | 0           | rw   | GPIOA reset<br>0: Does not reset GPIOA<br>1: Reset GPIOA    |
| Bit 16:13 | Reserved  | 0x0         | resd | Kept at its default value.                                  |
| Bit 12    | OTGFS1RST | 0x0         | rw   | OTGFS1 reset<br>0: Does not reset OTGFS1<br>1: Reset OTGFS1 |
| Bit 11:0  | Reserved  | 0x0000      | resd | Kept at its default value.                                  |

## PLL configuration register (CRM\_PLL)

Access: 0 wait state, by words, half-words and bytes.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31    | PLLCFGEN | 0x0         | rw   | PLL configuration enable<br>0: Common integer multiplication mode, which is done by PLL_FREF and PLLMULT registers.<br>1: Flexible configuration mode, which is done by PLL_MS/PLL_NS/PLL_FR registers.  |
| Bit 30:27 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 26:24 | PLL_FREF | 0x0         | rw   | PLL input clock selection<br>This field is valid only if PLLCFGEN = 0.<br>000: 3.9 ~ 5 MHz<br>001: 5.2 ~ 6.25 MHz<br>010: 7.8125 ~ 8.33 MHz<br>011: 8.33 ~ 12.5 MHz<br>100: 15.625 ~ 20.83 MHz<br>101: 20.83 ~ 31.25 MHz<br>110: Reserved<br>111: Reserved |
| Bit 23:17 | Reserved | 0x00        | resd | Kept at its default value.   |
| Bit 16:8  | PLL_NS   | 0x1F        | rw   | PLL multiplication factor<br>PLL_NS range (31 ~ 500)   |
| Bit 7:4   | PLL_MS   | 0x1         | rw   | PLL pre-division<br>PLL_MS range (1 ~ 15)  |

| Bit     | Name     | Reset value | Type | Description  |
|---------|----------|-------------|------|--|
| Bit 3   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 2:0 | PLL_FR   | 0x0         | rw   | PLL post-division factor<br>PLL_FR range (0 ~ 5)<br>000: PLL post-division = 1, divided by 1<br>001: PLL post-division = 2, divided by 2<br>010: PLL post-division = 4, divided by 4<br>011: PLL post-division = 8, divided by 8<br>100: PLL post- division = 16, divided by 16<br>101: PLL post- division = 32, divided by 32<br>Others: Reserved<br>It should be noted the relationship between the PLL-FR values and post-division factors. |

Note: PLL clock formulas:

PLL output clock = PLL input clock × PLL frequency multiplication factor / (PLL pre-divider factor × PLL post-divider factor)

500 MHz ≤ PLL input clock × PLL frequency multiplication factor / PLL pre-divider factor ≤ 1000 MHz

2 MHz ≤ PLL input clock / PLL pre-divider factor ≤ 16 MHz

### Additional register 1 (CRM\_MISC1)

Access: 0 to 3 wait states, accessible by words, half-words or bytes.

| Bit       | Name          | Reset value | Type | Description   |
|-----------|---------------|-------------|------|---|
| Bit 31:28 | CLKOUTDIV     | 0x0         | rw   | Clock output division<br>0xxx: Clock output<br>1000: Clock output divided by 2<br>1001: Clock output divided by 4<br>1010: Clock output divided by 8<br>1011: Clock output divided by 16<br>1100: Clock output divided by 64<br>1101: Clock output divided by 128<br>1110: Clock output divided by 256<br>1111: Clock output divided by 512                 |
| Bit 27:26 | Reserved      | 0x0         | resd | Kept its default value.   |
| Bit 25    | HICKDIV       | 0x0         | rw   | HICK 6 divider selection<br>This bit is used to select HICK or HICK /6. If the HICK/6 is selected, the clock frequency is 8 MHz. Otherwise, the clock frequency is 48 MHz.<br>0: HICK/6<br>1: HICK<br>Note: 1. When the HICK is used as PLL clock source, the HICKDIV must not change during PLL enable.<br>2. In any case, HICK always input 4 MHz to PLL. |
| Bit 24:17 | Reserved      | 0x00        | resd | Kept at its default value.  |
| Bit 16    | CLKOUT_SEL[3] | 0           | rw   | Clock output selection<br>This bit works with the bit [26:24] of the CRM_CFG register.  |
| Bit 15:8  | Reserved      | 0x00        | resd | Kept at its default value.  |
| Bit 7:0   | HICKCAL_KEY   | 0x00        | rw   | HICK calibration key<br>The HICKCAL [7:0] can be written only when this field is set 0x5A.  |

## OTG\_FS extended control register (CRM\_OTG\_EXTCTRL)

Access: 0 wait state, accessible by words, half-words or bytes.

| Bit      | Name       | Reset value | Type | Description  |
|----------|------------|-------------|------|--|
| Bit 31   | Reserved   | 0x0         | resd | Kept at its default value.   |
| Bit 30   | USBDIV_RST | 0x0         | rw   | USB divider reset<br>0: Does not reset USB divider<br>1: Reset USB divider<br><br>Note: Configure this bit to reset USB divider every time the PLL is enabled and PLLSTBL is set. After the USB divider is reset, perform two consecutive write operations to the CRM_CFG BIT27 (i.e., the MSB of USB divider factor). The value being written is inverse to the current CRM_CFG BIT27 value. For example, when the CRM_CFG BIT27 value is 0, write "1" and then "0" to the CRM_CFG BIT27 after USB divider reset. |
| Bit 29:0 | Reserved   | 0x0         | resd | Kept at its default value.   |

## Additional register 2 (CRM\_MISC2)

| Bit       | Name         | Reset value | Type | Description  |
|-----------|--------------|-------------|------|--|
| Bit 31:10 | Reserved     | 0x0000      | resd | Kept at its default value.   |
| Bit 9     | HICK_TO_SCLK | 0           | rw   | HICK as system clock frequency select<br>When HICK is used as a clock source of SCLKSEL, the SCLK frequency is:<br>0: Fixed 8 MHz, that is, HICK/6<br>1: 48 M or 8 M, depending on HICKDIV bit   |
| Bit 8     | HICK_TO_USB  | 0x0         | rw   | USB 48MHz clock source select<br>0: USB 48M clock source is PLL or PLL-division<br>1: USB 48M clock source is HICK or HICK/6<br><br>Note: The USB must work at 48M, meaning that HICKDIV = 1 must be asserted to ensure that it can select HICK 48 MHz output. |
| Bit 7:4   | Reserved     | 0x0         | resd | Fixed 0x0. Do not change.  |
| Bit 3:0   | Reserved     | 0x0         | resd | Fixed 0x0. Do not change.  |

# 5 Flash memory controller (FLASH)

## FLASH introduction

Flash memory is divided into three parts: main Flash memory, information block and Flash memory registers.

- Main Flash memory is up to 64 KB
- Information block consists of 4 KB bootloader and the user system data area. The bootloader uses USART1 or USART2 for ISP programming.

Main Flash memory contains bank 1 (64 KB), which is divided into 64 sectors of 1Kbytes.

**Table 11. Flash memory architecture (64 K)**

| Bank  |                  | Name                      | Address range             |
|---|------------------|---------------------------|---------------------------|
| Main memory                                 | Bank1<br>(64 KB) | Sector 0                  | 0x0800 0000 – 0x0800 03FF |
|   |                  | Sector 1                  | 0x0800 0400 – 0x0800 07FF |
|   |                  | Sector 2                  | 0x0800 0800 – 0x0800 0BFF |
|   |                  | ...                       | ...                       |
|   |                  | Sector 63                 | 0x0800 FC00 – 0x0800 FFFF |
| Information block<br>512 B user system data |                  | 4 KB boot memory          | 0x1FFF E400 – 0x1FFF F3FF |
|   |                  | 0x1FFF F800 – 0x1FFF F9FF |                           |

Main Flash memory contains bank 1 (32 KB), which is divided into 32 sectors of 1Kbytes.

**Table 12. Flash memory architecture (32 K)**

| Bank  |                  | Name                      | Address range             |
|---|------------------|---------------------------|---------------------------|
| Main memory                                 | Bank1<br>(32 KB) | Sector 0                  | 0x0800 0000 – 0x0800 03FF |
|   |                  | Sector 1                  | 0x0800 0400 – 0x0800 07FF |
|   |                  | Sector 2                  | 0x0800 0800 – 0x0800 0BFF |
|   |                  | ...                       | ...                       |
|   |                  | Sector 31                 | 0x0800 FC00 – 0x0800 FFFF |
| Information block<br>512 B user system data |                  | 4 KB boot memory          | 0x1FFF E400 – 0x1FFF F3FF |
|   |                  | 0x1FFF F800 – 0x1FFF F9FF |                           |

### User system data area

The system data will be read from the information block of Flash memory whenever a system reset occurs, and is saved in the user system data register (FLASH\_USD) and erase/programming protection status register (FLASH\_EPPS).

Each system data occupies two bytes, where the low bytes corresponds to the contents in the system data area, and the high bytes represent the inverse code that is used to verify the correctness of the selected bit. When the high byte is not equal to the inverse code of the low byte (except when both high and low byte are all 0xFF), the system data loader will issue a system data error flag (USDERR) and the corresponding system data and their inverse codes are forced 0xFF.

Note: The update of the contents in the user system data area becomes effective only after a system reset.

**Table 13. User system data area**

| Address     | Bit     | Description   |
|-------------|---------|---|
| 0x1FFF_F800 | [7:0]   | FAP[7:0]: Flash memory access protection (access protection enable/disable result is stored in bits [1] and [26] in the FLASH_USD register)<br>0xA5: Flash access protection disabled<br>0XCC: High-level Flash access protection enabled<br>Others; Low-level Flash access protection enabled<br>Note: SWD is disabled as soon as high-level Flash access protection is enabled. |
|             | [15:8]  | nFAP[7:0]: Inverse code of FAP[7:0]   |
|             | [23:16] | SSB[7:0]: System configuration byte (stored in the FLASH_USD[9:2])  |
|             |         | Bit 7 Reserved  |
|             |         | Bit 6 (nWDT_STDBY) 0: WDT stops counting while entering Standby mode<br>1: WDT does not stop counting while entering Standby mode   |
|             |         | Bit 5 (nWDT_DEPSLP) 0: WDT stops counting while entering Deepsleep mode<br>1: WDT does not stop counting while entering Deepsleep mode  |
|             |         | Bit 4 (nBOOT1) nBOOT1: It defines boot mode together with BOOT0 pin.<br>0: Boot from SRAM<br>1: Boot from boot memory   |
|             |         | Bit 3 Reserved.   |
|             |         | Bit 2 (nSTDBY_RST) 0: Reset occurs when entering Standby mode<br>1: No reset occurs when entering Standby mode  |
|             |         | Bit 1 (nDEPSLP_RST) 0: Reset occurs when entering Deepsleep mode<br>1: No reset occurs when entering Deepsleep mode   |
|             |         | Bit 0 (nWDT_ATO_EN) 0: Watchdog is enabled<br>1: Watchdog is disabled   |
|             | [31:24] | nSSB[7:0]: Inverse code of SSB[7:0]   |
| 0x1FFF_F804 | [7:0]   | Data0[7:0]: User data 0 (It is stored in the FLASH_USD[17:10] register)   |
|             | [15:8]  | nData0[7:0]: Inverse code of Data0[7:0]   |
|             | [23:16] | Data1[7:0]: User data 1 (It is stored in the FLASH_USD[25: 18] register)  |
|             | [31:24] | nData1[7:0]: Inverse code of Data1[7:0]   |
| 0x1FFF_F808 | [7:0]   | EPP0[7:0]:Flash erase/write protection byte 0 (in the FLASH_EPPS[7:0])<br>This field is used to protect sectors 0~31 of main Flash memory. Each bit takes care of 4 KB sectors (1 KB/sector).<br>0: Erase/write protection is enabled<br>1: Erase/write protection is disabled  |
|             | [15:8]  | nEPP0[7:0]: Inverse code of EPP0[7:0]   |
|             | [23:16] | EPP1[7:0]: Flash erase/write protection byte 1 (stored in the FLASH_EPPS[15:8])<br>This field is used to protect sectors 32 ~ 63 of main Flash memory. Each bit takes care of 4 KB sectors (1 KB/sector).<br>0: Erase/write protection is enabled<br>1: Erase/write protection is disabled  |
|             |         | nEPP1[7: 0]: Inverse code of EPP1[7:0]  |



| Address     | Bit     | Description   |
|-------------|---------|---|
| 0x1FFF_F80C | [7:0]   | EPP2[7:0]: Flash erase/write protection byte 2 (stored in the FLASH_EPPS[23:16])<br>Reserved  |
|             | [15:8]  | nEPP2[7:0]: Inverse code of EPP2[7:0]   |
|             | [23:16] | EPP3[7:0]: Flash erase/write protection byte 3 (stored in the FLASH_EPPS[31:24])<br>Bit [6:0] is reserved.<br>Bit [7] is used to protect main Flash memory extension area.<br>0: Erase/write protection is enabled<br>1: Erase/write protection is disabled |
|             | [31:24] | nEPP3[7:0]: Inverse code of EPP3[7:0]   |
| 0x1FFF_F810 | [7:0]   | Data2[7:0]: User system data 2  |
|             | [15:8]  | nData2[7:0]: Inverse code of Data2[7:0]   |
|             | [23:16] | Data3[7:0]: User system data 3  |
|             | [31:24] | nData3[7:0]: Inverse code of Data3[7:0]   |
| 0x1FFF_F814 | [7:0]   | Data4[7:0]: User system data 4  |
|             | [15:8]  | nData4[7:0]: Inverse code of Data4[7:0]   |
|             | [23:16] | Data5[7:0]: User system data 5  |
|             | [31:24] | nData5[7:0]: Inverse code of Data5[7:0]   |
| ...         | ...     |   |
| ...         | ...     |   |
| 0x1FFF_F9FC | [7:0]   | Data248[7:0]: User system data 248  |
|             | [15:8]  | nData248[7:0]: Inverse code of Data248[7:0]   |
|             | [23:16] | Data249[7:0]: User system data 249  |
|             | [31:24] | nData249[7:0]: Inverse code of Data249[7:0]   |

## Flash memory operation

### Unlock/lock

After reset, Flash memory is protected, by default. FLASH\_CTRL cannot be written. Write and erase operation can be performed only when the Flash memory is unlocked.

#### Unlock procedure:

Flash memory block can be unlocked by writing KEY1 (0x45670123) and KEY2 (0xCDEF89AB) to the FLASH\_UNLOCK register.

Note: Writing an incorrect key sequence leads to a bus error and the Flash memory is also locked until the next reset.

#### Lock procedure:

Flash memory block can be locked by setting the OPLK bit in the FLASH\_CTRL register.

### Erase operation

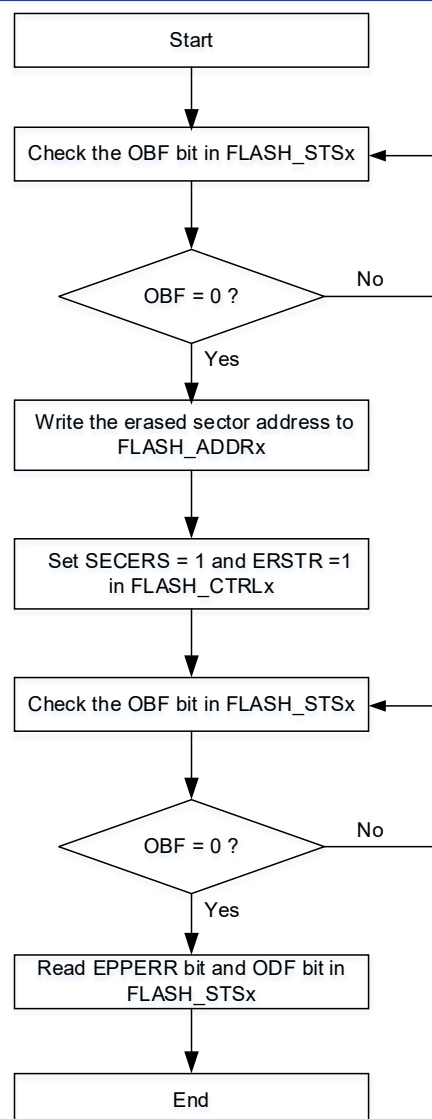
Erase operation must be done before programming. Flash memory erase includes sector erase and mass erase.

## Sector erase

Any sector in the Flash memory and its extension area can be erased with sector erase function independently. Below should be followed during sector erase:

- Check the OBF bit in the FLASH\_STS register to confirm that there is no other programming operation in progress;
- Write the sector to be erased in the FLASH\_ADDR register
- Set the SECERS and ERSTR bit in the FLASH\_CTRL register to enable sector erase
- Wait until the OBF bit becomes “0” in the FLASH\_STS register. Read the EPPERR bit and ODF bit in the FLASH\_STS register to verify the erased sectors.

Note: When the boot memory is configured as the Flash memory extension area, performing sector-erase operation erases the entire Flash memory extension area.



**Figure 13. Flash memory sector erase process**

### Mass erase

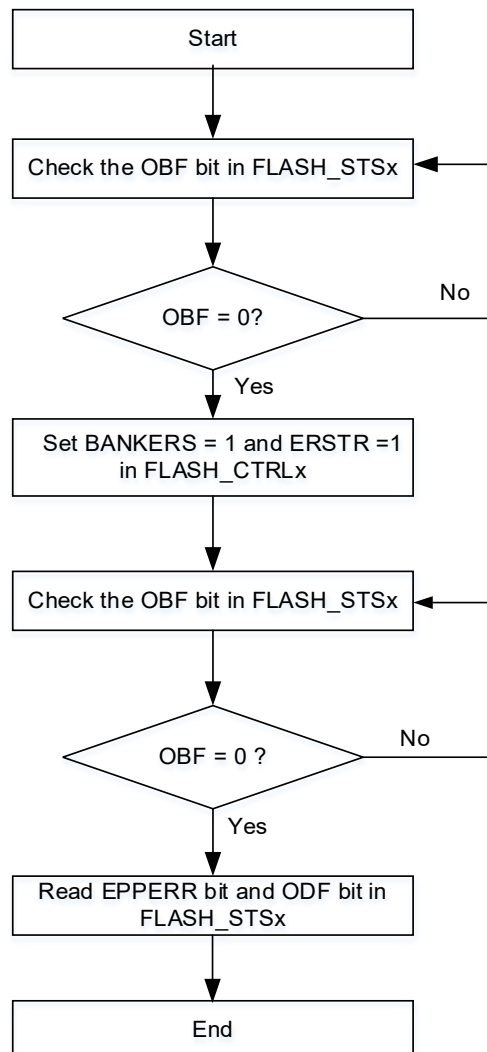
Mass erase function can erase all the Flash memory.

The following process is recommended:

- Check the OBF bit in the FLASH\_STS register to confirm that there is no other programming operation in progress;
- Set the BANKERS and ERSTR bits in the FLASH\_CTRL register to enable mass erase;
- Wait until the OBF bit becomes “0” in the FLASH\_STS register. Read the EPPERR bit and ODF bit in the FLASH\_STS register to verify the erased results.

Note: 1. When the boot memory is configured as the Flash memory extension area, performing bank-erase operation erases automatically the entire the entire Flash memory and its extension area.

2. Read access during erase operation halts the CPU and waits until the completion of erase.
3. Internal HICK must be enabled prior to erase operation.



**Figure 14. Flash memory mass erase process**

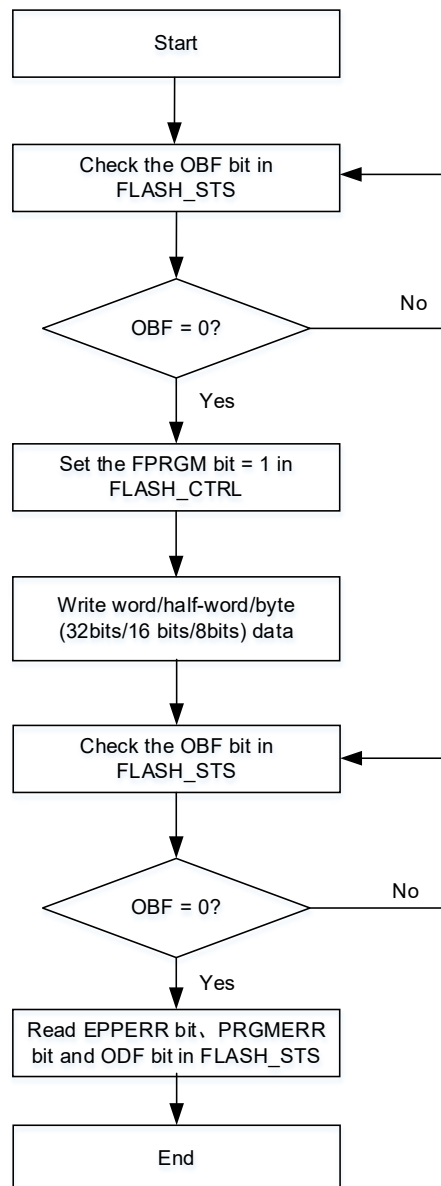
## Programming operation

The Flash memory can be programmed with 32 bits, 16 bits or 8 bits at a time.

The following process is recommended:

- Check the OBF bit in the FLASH\_STS register to confirm that there is no other programming operation in progress;
- Set the FPRGM bit in the FLASH\_CTRL register, so that the Flash memory programming instructions can be received;
- Write the data (word/half-word/byte) to be programmed to the designated address;
- Wait until the OBF bit in the FLASH\_STS register becomes “0”; read the EPPERR, PRGMERR and ODF bits to verify the programming result.

Note: 1. When the address to be written is not erased in advance, the programming operation is not executed unless the data to be written is all 0. In this case, a programming error is reported by the PRGMERR bit in the FLASH\_STS register.  
2. CPU halts during programming and waits until the completion of programming.  
3. Internal HICK must be enabled prior to programming.



**Figure 15. Flash memory programming process**

## Read operation

Flash memory can be accessed through AHB bus of the CPU.

## Main Flash memory extension area

Bootloader code area can also be programmed as the extension area of the main Flash memory to store user-application code. When used as main Flash memory extension area, it behaves like the main Flash memory, including read, unlock, erase and programming operations.

## User system data area operation

### Unlock/lock

After reset, user system data area is protected, by default. Write and erase operations can be performed only after the Flash memory is unlocked before the unlock operation for the user system data area.

#### Unlock procedure:

Flash memory block can be unlocked by writing KEY1 (0x45670123) and KEY2 (0xCDEF89AB) to the FLASH\_UNLOCK register;

When KEY1 (0x45670123) and KEY2 (0xCDEF89AB) is written to the FLASH\_USD\_UNLOCK register, the USDULKS bit in the FLASH\_CTRL register will be automatically set by hardware, indicating that it supports write/erase operation to the user system data area.

Note: Writing an incorrect key sequence leads to bus error and the Flash memory is also locked until the next reset.

#### Lock procedure:

User system data area is locked by clearing the USDULKS bit in the FLASH\_CTRL register by software.

### Erase operation

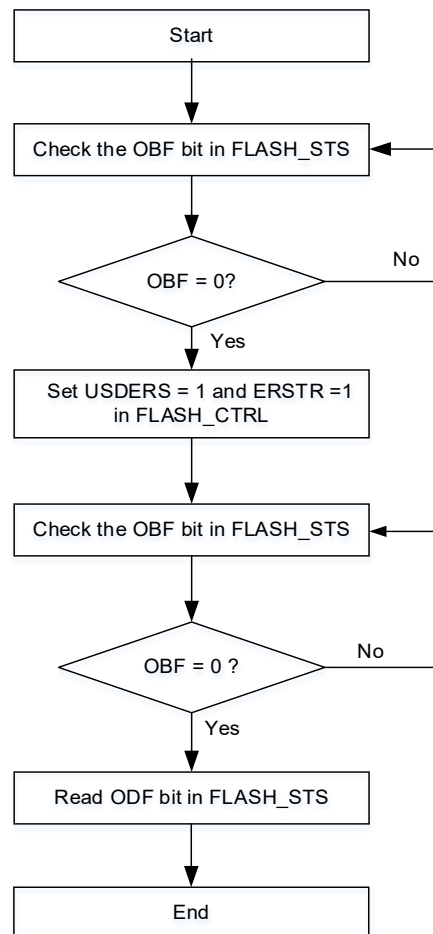
Erase operation must be done before programming. User system data area can perform erase operation independently.

Below should be followed during sector erase:

- Check the OBF bit in the FLASH\_STS register to confirm that there is no other programming operation in progress;
- Set the USDERS and ERSTR bit in the FLASH\_CTRL register to enable erase operation;
- Wait until the OBF bit in the FLASH\_STS register becomes “0”. Read the ODF bit in the FLASH\_STS register to verify the erase result.

Note: Read operation to the Flash memory during programming halts CPU and waits until the completion of erase.

The internal HICK must be enabled prior to erase operation.



**Figure 16. System data area erase process**

## Programming operation

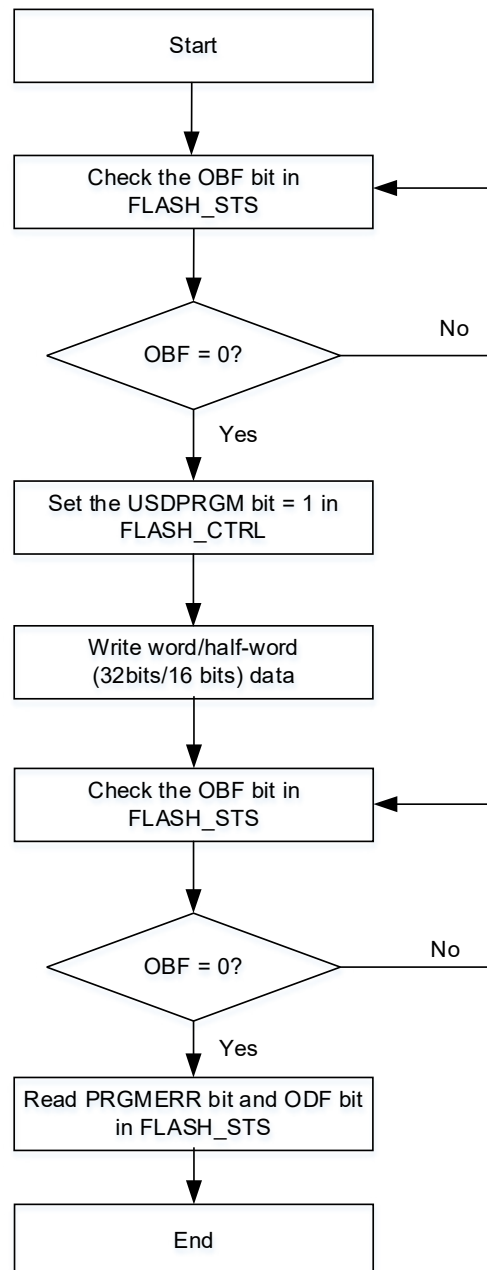
The User system data area can be programmed with 16 or 32 bits at a time.

The following process is recommended:

- Check the OBF bit in the FLASH\_STS register to confirm that there is no other programming operation in progress;
- Set the USDPGM bit in the FLASH\_CTRL register, so that the programming instructions for the user system data area can be received;
- Write the data (half-word/word) to be programmed to the designated address;
- Wait until the OBF bit in the FLASH\_STS register becomes “0”, read the PRGMERR and ODF bits to verify the programming result.

Note: CPU halts during programming and waits until the completion of programming.

The internal HICK must be enabled prior to programming operation.



**Figure 17. System data area programming process**

## Read operation

User system data area can be accessed through AHB bus of the CPU.



## Flash memory protection

Flash memory includes access and erase/program protection.

### Access protection

Flash memory access protection is divided into two parts: high-level and low-level.

Once enabled, only the Flash program is allowed to read Flash memory data. This read operation is not permitted in debug mode or by booting from non-Flash memory.

#### Low-level access protection

When the contents in the nFAP and FAP bytes are different from 0x5A and 0xA5, and 0x33 and 0xCC, the low-level Flash memory access protection is enabled after a system reset.

When the Flash access is protected, the user can re-erase the system data area, and unlock Flash access protection (switching from protected to unprotected state will trigger bank erase on the Flash memory automatically) by writing 0xA5 to FAP byte, and then perform a system reset. Subsequently, the system data loader will be reloaded with system data and updated with Flash memory access protection disable state (FAP byte)

#### High-level access protection

When the content in the nFAP is equal to 0x33, and the content in the FAP byte is equal to 0xCC, the high-level Flash memory access protection is enabled after a system reset.

Once enabled, it cannot be unlocked, and it is not permissible for users to re-erase and write the system data area.

Note: 1. The main memory extension area can also be protected.

2. If the access protection bit is set in debug mode, then the debug mode has to be cleared by POR instead of system reset in order to resume access to Flash memory data

3. The SWD is disabled as soon as the high-level access protection is enabled.

Table 14 shows Flash memory access limits when Flash access protection is enabled.

**Table 14. Flash memory access limit**

| Block                 | Protection level      | Access limits                                   |            |                                |                             |             |       |
|-----------------------|-----------------------|---|------------|--------------------------------|-----------------------------|-------------|-------|
| Read                  |                       | In debug mode or boot from SRAM and boot memory |            |                                | Boot from main Flash memory |             |       |
|                       |                       | Read  | Write      | Erase                          | Read                        | Write       | Erase |
| Main Flash memory     | Low-level protection  | Not allowed                                     |            | Not allowed <sup>(1) (2)</sup> |                             | Accessible  |       |
|                       | High-level protection | Not allowed                                     |            |                                |                             | Accessible  |       |
| User system data area | Low-level protection  | Not allowed                                     | Accessible |                                |                             | Accessible  |       |
|                       | High-level protection | Not allowed                                     |            |                                |                             | Not allowed |       |

(1) Main Flash memory is cleared automatically by hardware only when the access protection is disabled;

(2) Only sector erase is forbidden, and bank erase is not affected.

### Erase/program protection

For 64 KB and less Flash memory, erase/program protection is performed on the basis of 4 sectors. This is used to protect the contents in the Flash memory against inadvertent operation when the program crash occurs.

Erase/program operation is not permitted under one of the following events, and the EPPERR bit is set accordingly when

- Erasing/programming the sectors (in Flash memory and its extension area) where erase/program protection is enabled
- Performing bank erase on the sectors where erase/program protection enabled
- When the Flash access protection is enabled, the sectors 0~3 in the main Flash memory will be protected against erase/program automatically
- When the Flash access protection is enabled, the main Flash memory is protected against erase/program when it is in debug mode or when it is started from non-main Flash memory.

## Read access

To increase system clock frequency, program the number of wait states to access the Flash memory through the WTCYC bit in the FLASH\_PSR register.

The Flash read times can be decreased through the PFT\_EN, PFT\_EN2 and PFT\_LAT\_DIS bits in the FLASH\_PSR register.

## Special functions

### Security library settings

Security library is a defined area protected by a code in the main memory. This area is only executable but cannot be read (Except for I-Code and D-code buses), written, or deleted, unless a correct code is keyed in. Security library includes instruction security library and data security library.

#### Advantages of security library:

Security library is protected by codes so that solution providers can program core algorithm into this area;

Security library cannot be read or deleted (including ISP/IAP/SWD) but only executed unless code defined by the solution provider is keyed in;

The rest of the area can be used for secondary development by solution providers;

Solution providers can sell core algorithm with security library function and do not have to develop full solutions for every customer.

Security library helps prevent from deliberate damage or changing terminal application codes.

Note:

Security library code must be programmed based on sector level, and the start address must be aligned with Flash memory address;

Only I-Code bus is permitted to read instruction security library;

I-Code and D-Code bus are able to read read-only area;

In an attempt of writing or erasing security library code, a warning will be given by WRPRTFLR = 1 in the FLASH\_STS register;

Executing bank erase in the main memory will not erase the security library.

By default, security library setting register is unreadable and write protected. To enable write access to this register, security library must be unlocked first by writing 0xA35F6D24 to the SLIB\_

UNLOCK register, and checking the SLIB\_ULKF bit in the SLIB\_MISC\_STS register to verify if it is unlocked successfully and then writing the programmed value into the security library setting register.

Follow the steps below to enable security library:

- Check the OBF bit in the FLASH\_STS register to ensure that there is no other ongoing programming operation;
- Write 0xA35F6D24 to the SLIB\_KEYR register to unlock security library;
- Check the SLIB\_ULKF bit of SLIB\_MISC\_STS register to verify that it is unlocked successfully;
- If the security library is located in main Flash memory, it is necessary to set the sector to be protected (including instruction and read-only area) in the SLIB\_SET\_RANG register; if the security library is in the main Flash memory extension area, it is required to set the EM\_SLIB\_SET register;
- Wait until the OBF bit becomes “0”;
- Set a password in the SLIB\_SET\_PWD register;
- Wait until the OBF bit becomes “0”;
- Program the code to be saved in security library;
- Perform system reset, and then reload security library setting word;
- Read the SLIB\_STS0/STS1 register to verify the security library settings.

Note: It is not permitted to configure the main Flash memory and its extension area as security library simultaneously; Security library should enabled when the Flash access protection is disabled.

Follow the steps below to unlock security library:

- Write the previously set security library password to the SLIB\_PWD\_CLR register;
- Wait until the OBF bit becomes “0”;
- Perform system reset, and then reload security library setting word;
- Read the SLIB\_STS0 register to check the security library settings.

Note: Disabling the security library will automatically perform bank erase on the main memory and its extension area, as well as on security library setting block.

## Bootloader used as Flash memory extension

There is only one chance for users to program the bootloader as the main Flash extension area, which will have the same features as those of Flash memory after successful configuration as follows:

- Read the bit 0 in the SLIB\_STS0 register to obtain the current mode of boot memory;
- Write the value 0xA35F6D24 to the SLIB\_UNLOCK register to unlock the current mode of boot memory;
- Write 0xFF to the bit [7:0] in the BTM\_MODE\_SET register;
- Wait until the OBF bit becomes “0”;
- Perform a system reset, and reload setting words;
- Read the SLIB\_STS0 register to verify.

- Note: 1. The above-mentioned process must be performed when the Flash memory access protection is disabled.
2. After this feature is enabled, the Flash memory will be used as the boot space, instead of the original boot memory.

## CRC verify

The optional CRC check for security library code or user code is performed on a sector level.

- Polynomial generator: 0x4C11DB7,
- That is,  $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$
- CRC initial value: 0x00000000

CRC verify procedure as follows:

- Check the OBF bit in the FLASH\_STS register to confirm that there is no other programming operation in progress;
- Program the start address of the code to be CRC check in the FLASH\_CRC\_ADDR register;
- Program the code count (in terms of sectors) to be CRC check through the bit [5:0] in the FLASH\_CRC\_CTRL register;
- Enable CRC verify by setting the bit 16 of the FLASH\_CRC\_CTRL register;
- Wait until the OBF bit becomes “0”;
- Read the FLASH\_CRC\_CHKR register to verify.

- Note: 1. The values of the FLASH\_CRC\_ADDR register must be aligned with the start address of the sector;
2. CRC verify must not cross the main Flash memory and its extension area.

## Flash memory registers

These peripheral registers must be accessed by words (32 bits).

**Table 15. Flash memory register map and reset value**

| Register         | Offset | Reset value |
|------------------|--------|-------------|
| FLASH_PSR        | 0x00   | 0x0000 0030 |
| FLASH_UNLOCK     | 0x04   | 0xFFFF XXXX |
| FLASH_USD_UNLOCK | 0x08   | 0xFFFF XXXX |
| FLASH_STS        | 0x0C   | 0x0000 0000 |
| FLASH_CTRL       | 0x10   | 0x0002 0080 |
| FLASH_ADDR       | 0x14   | 0x0000 0000 |
| FLASH_USD        | 0x1C   | 0x03FF FFFC |
| FLASH_EPPS       | 0x20   | 0xFFFF FFFF |
| SLIB_STS0        | 0x74   | 0x00FF 0000 |
| SLIB_STS1        | 0x78   | 0xFFFF FFFF |
| SLIB_PWD_CLR     | 0x7C   | 0xFFFF FFFF |
| SLIB_MISC_STS    | 0x80   | 0x0000 0000 |
| FLASH_CRC_ADDR   | 0x84   | 0x0000 0000 |
| FLASH_CRC_CTRL   | 0x88   | 0x0000 0000 |
| FLASH_CRC_CHKR   | 0x8C   | 0x0000 0000 |
| SLIB_SET_PWD     | 0x160  | 0x0000 0000 |

| Register       | Offset | Reset value |
|----------------|--------|-------------|
| SLIB_SET_RANGE | 0x164  | 0x0000 0000 |
| EM_SLIB_SET    | 0x168  | 0x0000 0000 |
| BTM_MODE_SET   | 0x16C  | 0x0000 0000 |
| SLIB_UNLOCK    | 0x170  | 0x0000 0000 |

### Flash performance select register (FLASH\_PSR)

| Bit      | Name        | Reset value | Type | Description   |
|----------|-------------|-------------|------|---|
| Bit 31:9 | Reserved    | 0x00000     | resd | Kept at its default value.  |
| Bit 8    | PFT_LAT_DIS | 0           | ro   | Prefetch latency disable<br>0: Prefetch buffer latency enabled, one-wait state for buffer access<br>1: Prefetch buffer latency disabled, 0-wait state for buffer access<br>Note: It is recommended to set this bit to 1, and do not change.                                     |
| Bit 7    | PFT_ENF2    | 0           | rw   | Prefetch enable flag 2<br>When this bit is set, it indicates that Flash prefetch buffer 2 is enabled.   |
| Bit 6    | PFT_EN2     | 0           | rw   | Prefetch enable 2<br>0: Prefetch buffer 2 is disabled<br>1: Prefetch buffer 2 is enabled<br>Note: It is recommended to set this bit to 1, and do not change.  |
| Bit 5    | PFT_ENF     | 1           | rw   | Prefetch enable flag<br>When this bit is set, it indicates that Flash prefetch buffer is enabled  |
| Bit 4    | PFT_EN      | 1           | rw   | Prefetch enable<br>0: Prefetch buffer is disabled<br>1: Prefetch buffer is enabled  |
| Bit 3    | Reserved    | 0           | resd | Always 0.   |
| Bit 2:0  | WTCYC       | 0x0         | rw   | Wait states<br>The wait states depends on the size of the system clock, and they are in terms of system clocks.<br>000: Zero wait state when 0MHz<system clock≤32MHz<br>001: One wait state when 32MHz<system clock≤64MHz<br>010: Two wait states when 64MHz<system clock≤96MHz |

### Flash unlock register (FLASH\_UNLOCK)

| Bit      | Name  | Reset value | Type | Description  |
|----------|-------|-------------|------|--|
| Bit 31:0 | UKVAL | 0XXXXX XXXX | wo   | Unlock key value<br>This is used to unlock Flash memory bank and its extension area. |

Note: All these bits are write-only, and return 0 when being read.

### Flash user system data unlock register (FLASH\_USD\_UNLOCK)

| Bit      | Name      | Reset value | Type | Description                       |
|----------|-----------|-------------|------|-----------------------------------|
| Bit 31:0 | USD_UKVAL | 0XXXXX XXXX | wo   | User system data Unlock key value |

Note: All these bits are write-only, and return 0 when being read.

## Flash status register (FLASH\_STS)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31:6 | Reserved | 0x0000000   | resd | Kept at its default value  |
| Bit 5    | ODF      | 0           | rw   | Operation done flag<br>This bit is set by hardware when Flash memory operations (program/erase) is completed. It is cleared by writing "1".                    |
| Bit 4    | EPPERR   | 0           | rw   | Erase/program protection error<br>This bit is set by hardware when programming the erase/program-protected Flash memory address. It is cleared by writing "1". |
| Bit 3    | Reserved | 0           | resd | Kept at its default value.   |
| Bit 2    | PRGMERR  | 0           | rw   | Programming error<br>When the programming address is not "0xFFFF", this bit is set by hardware. It is cleared by writing "1".                                  |
| Bit 1    | Reserved | 0           | resd | Kept at its default value.   |
| Bit 0    | OBF      | 0           | ro   | Operation busy flag<br>When this bit is set, it indicates that Flash memory operation is in progress. It is cleared when operation is completed.               |

## Flash control register (FLASH\_CTRL)

| Bit        | Name     | Reset value | Type | Description  |
|------------|----------|-------------|------|--|
| Bit 31:18  | Reserved | 0x0000      | resd | Kept at its default value  |
| Bit 17     | LPMEN    | 1           | rw   | Low power mode enable<br>0: Low power mode is disabled;<br>1: Low power mode enabled.<br>After this bit is set, the Flash memory enters low-power mode as soon as the MCU moves into DeepSleep mode.   |
| Bit 16:13  | Reserved | 0x0         | resd | Kept its default value   |
| Bit 12     | ODIFE    | 0           | rw   | Operation done flag interrupt enable<br>0: Interrupt is disabled;<br>1: Interrupt is enabled.  |
| Bit 11,8,3 | Reserved | 0           | resd | Kept its default value   |
| Bit 10     | ERRIE    | 0           | rw   | Error interrupt enable<br>This bit enables EPPERR or PROGERR interrupt.<br>0: Interrupt is disabled;<br>1: Interrupt is enabled.   |
| Bit 9      | USDULKS  | 0           | rw   | User system data unlock success<br>This bit is set by hardware when the user system data is unlocked properly, indicating that erase/program operation to the user system data is allowed. This bit is cleared by writing "0", which will re-lock the user system data area.   |
| Bit 7      | OPLK     | 1           | rw   | Operation lock<br>This bit is set by default, indicating that Flash memory is protected against operations. This bit is cleared by hardware after unlock, indicating that erase/program operation to Flash memory is allowed. Writing "1" can re-lock Flash memory operations. |

| Bit   | Name    | Reset value | Type | Description   |
|-------|---------|-------------|------|---|
| Bit 6 | ERSTR   | 0           | rw   | Erase start<br>An erase operation is triggered when this bit is set. This bit is cleared by hardware after the completion of the erase operation. |
| Bit 5 | USDERS  | 0           | rw   | User system data erase<br>It indicates the user system data erase.  |
| Bit 4 | USDPRGM | 0           | rw   | User system data program<br>It indicates the user system data program.  |
| Bit 2 | BANKERS | 0           | rw   | Bank erase<br>It indicates bank erase operation.  |
| Bit 1 | SECERS  | 0           | rw   | Sector erase<br>It indicates sector erase operation.  |
| Bit 0 | FPRGM   | 0           | rw   | Flash program<br>It indicates Flash program operation.  |

### Flash address register (FLASH\_ADDR)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 31:0 | FA   | 0x0000 0000 | wo   | Flash address<br>Select the address of the bank/sector to be erased. |

### User system data register (FLASH\_USD)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:27 | Reserved | 0x00        | resd | Kept at its default value  |
| Bit 26    | FAP_HL   | 0           | ro   | Flash access protection high level<br>The status of the Flash access protection is determined by bit 26 and bit 1.<br>00: Flash access protection disabled, and FAP = 0xA5<br>01: Low-level Flash access protection enabled, and FAP = non-0xCC and non-0xA5.<br>10: Reserved<br>11: High-level Flash access protection, and FAP = 0xCC<br>The SWD is disabled as soon as the high-level access protection is enabled. |
| Bit 25:18 | USER_D1  | 0xFF        | ro   | User data 1  |
| Bit 17:10 | USER_D0  | 0xFF        | ro   | User data 0  |
| Bit 9:2   | SSB      | 0xFF        | ro   | System setting byte<br>Includes the system setting bytes in the loaded user system data area<br>Bit 9: Unused<br>Bit 8: nSTDBY_WDT<br>Bit 7: nDEPSLP_WDT<br>Bit 6: nBOOT1<br>Bit 5: Unused<br>Bit 4: nSTDBY_RST<br>Bit 3: nDEPSLP_RST<br>Bit 2: nWDT_ATO_EN  |
| Bit 1     | FAP      | 0           | ro   | Flash access protection<br>Access to Flash memory is not allowed when this bit is set.   |

| Bit   | Name   | Reset value | Type | Description  |
|-------|--------|-------------|------|--|
| Bit 0 | USDERR | 0           | ro   | User system data error<br>When this bit is set, it indicates that certain byte does not match its inverse code in the user system data area. At this point, this byte and its inverse code will be forced to 0xFF when being read. |

### Erase/program protection status register (FLASH\_EPPS)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 31:0 | EPPS | 0xFFFF FFFF | ro   | Erase/Program protection status<br>This register reflects the erase/program protection byte status in the loaded user system data. |

### Flash security library status register 0 (SLIB\_STS0)

For Flash security library only.

| Bit       | Name            | Reset value | Type | Description  |
|-----------|-----------------|-------------|------|--|
| Bit 31:24 | Reserved        | 0x00        | resd | Kept at its default value  |
| Bit 23:16 | EM_SLIB_INST_SS | 0xFF        | ro   | Extension memory sLib instruction start sector<br>Others: invalid<br>00000000: Sector 0<br>00000001: Sector 1<br>00000010: Sector 2<br>00000011: Sector 3<br>11111111: No instruction sLib<br>Others: Invalid                              |
| Bit 15:4  | Reserved        | 0x000       | resd | Kept at its default value  |
| Bit 3     | SLIB_ENF        | 0           | ro   | SLIB_ENF: sLib enable flag<br>When this bit is set, it indicates that the main Flash memory is partially or completely (depending on the setting of SLIB_STS1) used as security library code.  |
| Bit 2     | EM_SLIB_ENF     | 0           | ro   | Extension memory sLib enable flag<br>When this bit is set, it indicates that the bootloader code area is used as the Flash extension area (BTM_AP_ENF is set), and stores security library code.   |
| Bit 1     | Reserved        | 0           | resd | Kept at its default value  |
| Bit 0     | BTM_AP_ENF      | 0           | ro   | Boot memory store application code enabled flag<br>When this bit is set, it indicates that the bootloader memory can be used as main Flash extension area to store user application code; otherwise, it is only used for system boot code. |



## Flash security library status register 1 (SLIB\_STS1)

For Flash security library only.

| Bit       | Name         | Reset value | Type | Description  |
|-----------|--------------|-------------|------|--|
| Bit 31:22 | SLIB_ES      | 0x3FF       | ro   | Security library end sector<br>0000000001: Sector 1<br>0000000010: Sector 2<br>...<br>0000111111: Sector 31 (the last sector of 32 KB main Flash memory)<br>...<br>0000111111: Sector 63 (the last sector of 64 KB main Flash memory)  |
| Bit 21:11 | SLIB_INST_SS | 0x7FF       | ro   | Security library instruction start sector<br>0000000000: Sector 0<br>0000000001: Sector 1<br>0000000010: Sector 2<br>...<br>0000011111: Sector 31 (the last sector of 32 KB main Flash memory)<br>...<br>0000111111: Sector 63 (the last sector of 64 KB main Flash memory)<br>1111111111: No instruction sLib |
| Bit 10:0  | SLIB_SS      | 0x7FF       | ro   | Security library start sector<br>0000000000: Sector 0<br>0000000001: Sector 1<br>0000000010: Sector 2<br>...<br>0000011111: Sector 31 (the last sector of 32 KB main Flash memory)<br>...<br>0000111111: Sector 63 (the last sector of 64 KB main Flash memory)  |

## Security library password clear register (SLIB\_PWD\_CLR)

Only used in Flash security library.

| Bit      | Name          | Reset value | Type | Description  |
|----------|---------------|-------------|------|--|
| Bit 31:0 | SLIB_PCLR_VAL | 0x0000 0000 | wo   | Security library password clear value<br>This register is used to key in a correct sLib password in order to unlock sLib function.<br>The write status of this register is indicated by bit 0 and bit 1 of the SLIB_MISC_STS register. |

## Security library additional status register (SLIB\_MISC\_STS)

For Flash security library only.

| Bit      | Name         | Reset value | Type | Description  |
|----------|--------------|-------------|------|--|
| Bit 31:3 | Reserved     | 0x0000000   | resd | Kept at its default value  |
| Bit 2    | SLIB_ULKF    | 0           | ro   | Security library unlock flag<br>When this bit is set, it indicates that sLib-related setting registers can be configured.  |
| Bit 1    | SLIB_PWD_OK  | 0           | ro   | Security library password ok<br>This bit is set by hardware when the password is correct.  |
| Bit 0    | SLIB_PWD_ERR | 0           | ro   | Security library password error<br>This bit is set by hardware when the password is incorrect and the setting value of the password clear register is different from 0xFFFF FFFF.<br>Note: When this bit is set, the hardware will no longer agree to re-program the password clear register until the next reset. |

## Flash CRC address register (FLASH\_CRC\_ADDR)

For main Flash memory and its extension area.

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:0 | CRC_ADDR | 0x0000 0000 | wo   | CRC address<br>This register is used to select a start address of the sector to be CRC checked. |

Note: All these bits are write-only, and return no response when being read.

## Flash CRC control register (FLASH\_CRC\_CTRL)

For main Flash memory and its extension area.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:17 | Reserved | 0x0000      | resd | Kept at its default value.  |
| Bit 16    | CRC_STRT | 0           | WO   | CRC start<br>This bit is used to enable CRC check for user code or sLib code. It is automatically cleared after enabling CRC by hardware.<br>Note: CRC data ranges from CRC_ADDR to CRC_ADDR + CRC_SN × 1 |
| Bit 15: 0 | CRC_SN   | 0x0000      | wo   | CRC sector number<br>This bit defines the sector to be CRC checked.   |

## Flash CRC check result register (FLASH\_CRC\_CHKR)

For Flash memory and its extension area.

| Bit      | Name     | Reset value | Type | Description      |
|----------|----------|-------------|------|------------------|
| Bit 31:0 | CRC_CHKR | 0x0000 0000 | ro   | CRC check result |

Note: All these bits are read-only, and return no response when being written.

## Security library password setting register (SLIB\_SET\_PWD)

For Flash security library password setting only.

| Bit      | Name          | Reset value | Type | Description   |
|----------|---------------|-------------|------|---|
| Bit 31:0 | SLIB_PSET_VAL | 0x0000 0000 | ro   | sLib password setting value<br>Note: This register can be written only after sLib is unlocked.<br>It is used to set a password of sLib. Writing 0xFFFF_FFFF or 0x0000_0000 has no effect. |

Note: All these bits are write-only, and return 0 when being read.

## Security library address setting register (SLIB\_SET\_RANGE)

For Flash security library address setting only.

| Bit       | Name         | Reset value | Type | Description   |
|-----------|--------------|-------------|------|---|
| Bit 31:22 | SLIB_ES_SET  | 0x000       | wo   | Security library end sector setting<br>These bits are used to set the security library end page.<br>0000000000: Sector 0<br>0000000001: Sector 1<br>0000000010: Sector 2<br>...<br>0000011111: Sector 31 (the last sector of 32 KB main Flash memory)<br>...<br>0000111111: Sector 63 (the last sector of 64 KB main Flash memory)  |
| Bit 21:11 | SLIB_ISS_SET | 0x000       | wo   | Security library instruction start sector setting<br>These bits are used to set the security library instruction start page.<br>0000000000: Sector 0<br>0000000001: Sector 1<br>0000000010: Sector 2<br>...<br>0000011111: Sector 31 (the last sector of 32 KB main Flash memory)<br>...<br>0000111111: Sector 63 (the last sector of 64 KB main Flash memory)<br>1111111111: No instruction sLib |
| Bit 10:0  | SLIB_SS_SET  | 0x000       | wo   | Security library start sector setting<br>These bits are used to set the security library start page.<br>0000000000: Sector 0<br>0000000001: Sector 1<br>0000000010: Sector 2<br>...<br>0000011111: Sector 31 (the last sector of 32 KB main Flash memory)<br>...<br>0000111111: Sector 63 (the last sector of 64 KB main Flash memory)  |

Note: 1. All these bits are write-only, and return 0 when being read.  
2. This register can be written only when security library is unlocked.  
3. Being out of the Flash address range is an invalid setting.

## Flash extension memory security library setting register (EM\_SLIB\_SET)

For Flash extension area only.

| Bit       | Name            | Reset value | Type | Description  |
|-----------|-----------------|-------------|------|--|
| Bit 31:24 | Reserved        | 0x00        | resd | Kept at its default value  |
| Bit 23:16 | EM_SLIB_ISS_SET | 0x000       | wo   | Extension memory sLib instruction start sector<br>00000000: Sector 0<br>00000001: Sector 1<br>00000010: Sector 2<br>00000011: Sector 3<br>11111111: No instruction sLib<br>Others: Invalid<br>Note: When it is set to 0xFF, it indicates that the extension area from sectors 0 to 3 is the security library, read-only. |
| Bit 15:0  | EM_SLIB_SET     | 0x000       | wo   | Extension memory sLib setting<br>Extension memory is configured as security library by writing 0x5AA5.   |

Note: All these bits are write-only, and return no response when being read.

## Boot memory mode setting register (BTM\_MODE\_SET)

For boot memory only.

| Bit      | Name         | Reset value | Type | Description   |
|----------|--------------|-------------|------|---|
| Bit 31:8 | Reserved     | 0x000000    | resd | Kept at its default value.  |
| Bit 7:0  | BTM_MODE_SET | 0x00        | wo   | Boot memory mode setting<br>0xFF: Boot memory serves as a system area that stores system boot code<br>Others: Boot memory serves a Flash extension area that stores application code<br>Note: This register can be set only when Flash access protection is disabled. |

Note: All these bits are write-only, and return no response when being read.

## Security library unlock register (FLASH\_UNLOCK)

For security library register unlock only.

| Bit      | Name       | Reset value | Type | Description  |
|----------|------------|-------------|------|--|
| Bit 31:0 | SLIB_UKVAL | 0x0000 0000 | wo   | Security library unlock key value<br>Fixed key value is 0xA35F_6D24, used for security library setting register unlock |

Note: All these bits are write-only, and return 0 when being read.

# 6 GPIOs and IOMUX

## Introduction

HT32F49041 series supports up to 55 bidirectional I/O pins, namely PA0-PA15, PB0-PB15, PC0-PC15, PD2, PF0-PF1, and PF4-PF7. Each of these pins features communication, control and data collection. In addition, their main features also include:

- Each pin can be configured by software as general-purpose input/output, multiplexed function and analog pin. When used as general-purpose output or IOMUX, each pin can be configured as push-pull/open-drain output.
- Each pin with individual weak pull-up/pull-down capability.
- Each pin output drive capability is configurable by software.
- Each pin can be configured as external interrupt input.
- Each pin supports write protection function.

## Function overview

### GPIO structure

I/O port registers can be accessed by words, half-words or bytes, and each I/O port bit can be programmed freely.

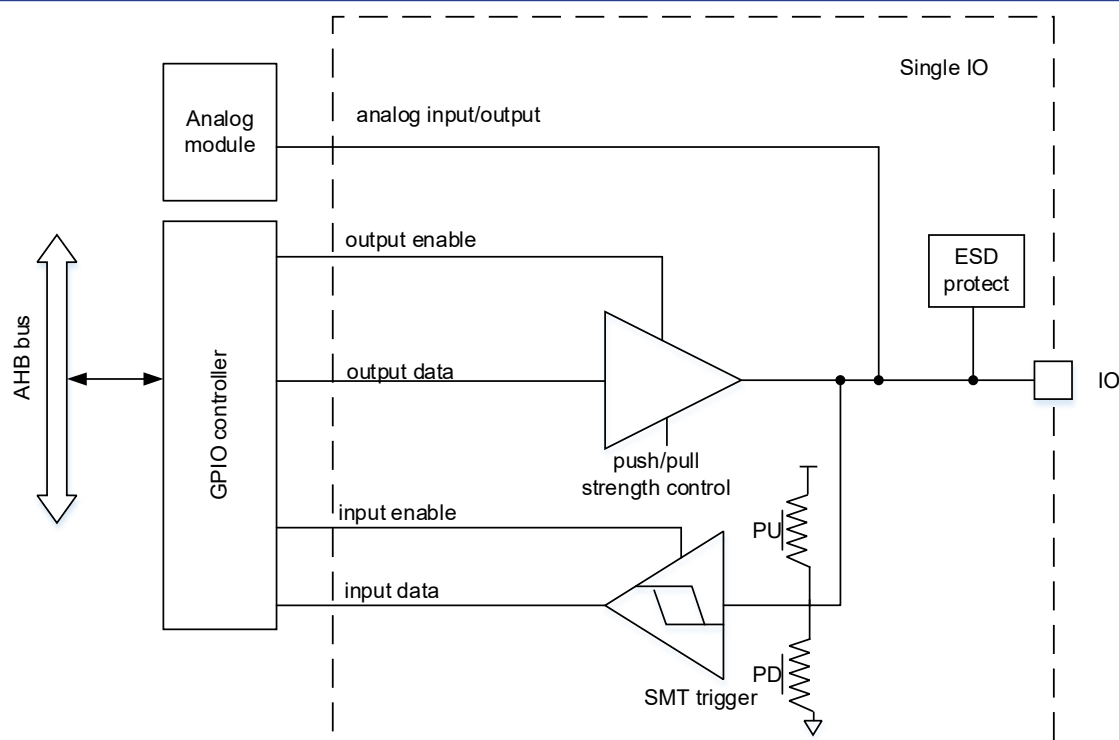


Figure 18. GPIO basic structure

## GPIO reset status

After power-on or system reset, all pins are configured as floating input mode except SWD-related pins. SWD pin configuration are as follows:

- PA13/SWDIO multiplexed pull-up
- PA14/SWCLK multiplexed pull-down

## General-purpose input configuration

| Mode            | IOMC | PULL |
|-----------------|------|------|
| Floating input  | 00   | 00   |
| Pull-down input |      | 10   |
| Pull-up input   |      | 01   |

When an I/O port is configured as general-purpose input:

- Get I/O states by reading the input data register.
- Schmitt-trigger input is activated.

Note: In floating input mode, it is recommended to set the unused pins as analog input mode in order to avoid leakage caused by interference from unused pins in a complex environment.

## Analog mode configuration

| Mode        | IOMC | PULL   |
|-------------|------|--------|
| Analog mode | 11   | Unused |

When an I/O port is configured as analog mode:

- Schmitt-trigger input is disabled.
- Output data register is invalid.
- Pull-up/pull-down configuration is invalid.
- I/O states cannot be obtained by reading the input data register.

## General-purpose output configuration

| Mode                                 | IOMC | OM | HDRV   | ODRV[1:0] | PULL     |
|--------------------------------------|------|----|--|-----------|----------|
| Push-pull without pull-up/pull-down  | 01   | 0  | 000: Output mode, normal sourcing/sinking strength<br>001: Output mode, large sourcing/sinking strength<br>010: Output mode, normal sourcing/sinking strength<br>011: Output mode, normal sourcing/sinking strength<br>1xx: Output mode, maximum sourcing/sinking strength |           | 00 or 11 |
| Push-pull with pull-up               | 01   | 0  |  |           | 01       |
| Push-pull with pull-down             | 01   | 0  |  |           | 10       |
| Open-drain without pull-up/pull-down | 01   | 1  |  |           | 00 or 11 |
| Open-drain with pull-up              | 01   | 1  |  |           | 01       |
| Open-drain with pull-down            | 01   | 1  |  |           | 10       |

When an I/O port is configured as general-purpose output:

- Schmitt-trigger input is enabled.
- In open-drain mode, output 0 when the output data register is configured as 0. When the output data register is configured as 1, the pin is in high-impedance state if there is no internal pull-up/pull-down, or the pin is pulled up to 1 or pulled down to 0 if there is internal pull-up or pull down.
- In push-pull mode, output register is used to output 0/1.

- Get I/O states by reading the input data register.
- GPIO set/clear register is used to set/clear the corresponding GPIO output data registers.

Note: If both IOCB and IOSB bits are set in the GPIO set/clear register, the IOSB takes priority.

## GPIO port write protection

Each I/O port supports write protection function. When write protection is enabled, I/O port configuration cannot be modified until the next reset or power-on.

## IOMUX structure

Several peripheral functions can be mapped on each IO pin. Peripheral input/output corresponding to an I/O pin is selected through IOMUX input/output table. Each I/O pin has up to 8 IOMUX mapping options for flexible selection, configured through the GPIOx\_MUXL (for pin 0 to 7) and GPIOx\_MUXH (for pin 8 to 15) registers.

Each I/O pin is connected to only one peripheral's pin by setting the GPIOx\_MUXL or GPIOx\_MUXH register so that there can be no conflict between peripherals sharing the same pin.

While being used as multiplexed function input, the I/O pin is disconnected from GPIO controller and it is controlled by IOMUX controller.

In multiplexed input mode, the I/O pin can be configured as multiplexed mode by setting the GPIOx\_CFGR register, and it can be configured as floating, pull-up/pull-down by setting the GPIOx\_PULL register according to the peripheral characteristics.

When used as multiplexed output or bidirectional IOMUX, the I/O pin can be configured as multiplexed mode by setting the GPIOx\_CFGR register, as push-pull/open-drain mode by setting the GPIOx\_OMODE register or pull-up/pull-down by setting the GPIOx\_PULL register according to the peripheral characteristics. Configure output drive capability by setting the GPIOx\_ODRVR and GPIOx\_HDRV register.

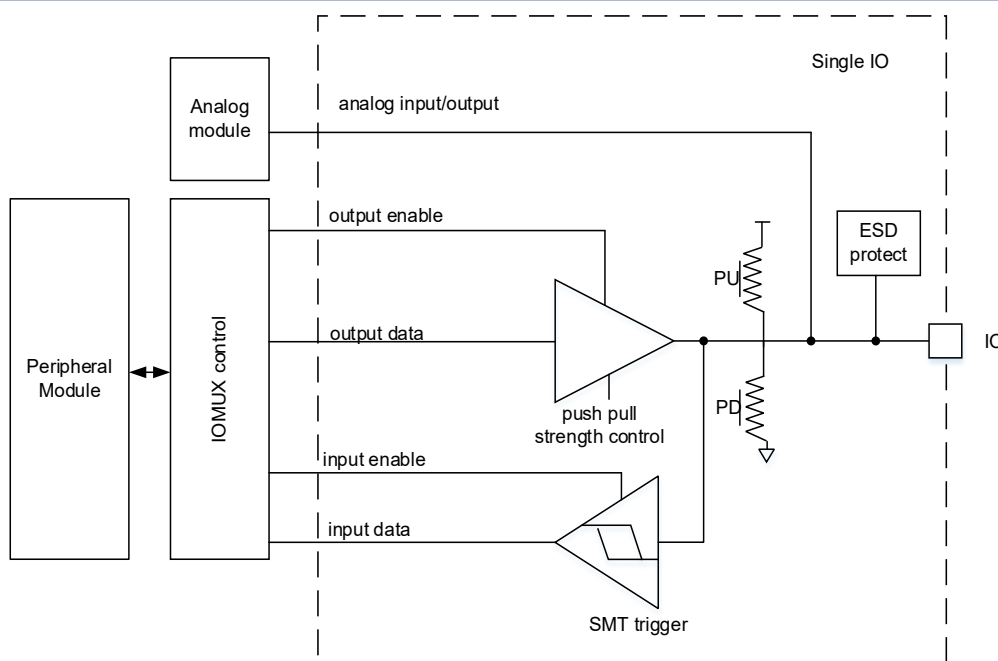


Figure 19. IOMUX structure

## Multiplexed function configuration

| Mode  | IOMC | OM | HDRV   | ODRV[1:0] | PULL     |
|---|------|----|--|-----------|----------|
| Multiplexed function push-pull without pull-up/pull-down  | 10   | 0  | 000: output mode, normal sourcing/sinking strength<br>001: output mode, large sourcing/sinking strength<br>010: output mode, normal sourcing/sinking strength<br>011: output mode, normal sourcing/sinking strength<br>1xx: output mode, maximum sourcing/sinking strength<br><br>Valid for multiplexed function output or bidirectional multiplexed function only | 00 or 11  | 00 or 11 |
| Multiplexed function push-pull with pull-up               | 10   | 0  |  |           | 01       |
| Multiplexed function push-pull with pull-down             | 10   | 0  |  |           | 10       |
| Multiplexed function open-drain without pull-up/pull-down | 10   | 1  |  |           | 00 or 11 |
| Multiplexed function open-drain with pull-up              | 10   | 1  |  |           | 01       |
| Multiplexed function open-drain with pull-down            | 10   | 1  |  |           | 10       |
| Multiplexed function input without pull-up/pull-down      | 10   | x  |  |           | 00 or 11 |
| Multiplexed function input with pull-up                   | 10   | x  |  |           | 01       |
| Multiplexed function input with pull-down                 | 10   | x  |  |           | 10       |

When an I/O port is configured as multiplexed function port:

- Schmitt-trigger input is activated.
- In open-drain output mode, output 0 when the peripheral outputs 0. When the peripheral outputs 1, the pin is in high-impedance state if there is no internal pull-up/pull-down, or it is pulled up to 1 or pulled down to 0 if there is internal pull-up or pull-down.
- In push-pull output mode, output 0/1 according to the peripheral output.

## IOMUX input/output

The multiplexed function of each IO port line is configured through the GPIOx\_MUXL (for pin 0 to 7) or GPIOx\_MUXH (for pin 8 to 15) register.

**Table 16. Port A multiplexed function configuration with GPIOA\_MUX\* register**

| Pin | MUX0                    | MUX1          | MUX2         | MUX3                 | MUX4       | MUX5       | MUX6                | MUX7                  |
|-----|-------------------------|---------------|--------------|----------------------|------------|------------|---------------------|-----------------------|
| PA0 | USART2_RX               | USART2_CTS    | TMR2_CH1_EXT | I2C2_SCL             | USART4_TX  | TMR1_EXT   |                     |                       |
| PA1 | EVENTOUT                | USART2_RTS_DE | TMR2_CH2     | I2C2_SDA             | USART4_RX  | TMR15_CH1N | I2C1_SMBA           | SPI3_MOSI/<br>I2S3_SD |
| PA2 | TMR15_CH1               | USART2_TX     | TMR2_CH3     |                      | CAN_RX     |            |                     |                       |
| PA3 | TMR15_CH2               | USART2_RX     | TMR2_CH4     |                      | CAN_TX     | I2S2_MCK   |                     |                       |
| PA4 | SPI1_CS /<br>I2S1_WS    | USART2_CK     | OTG_FS_NOE   | SPI3_CS /<br>I2S3_WS | TMR14_CH1  | I2C1_SCL   | SPI2_CS/<br>I2S2_WS |                       |
| PA5 | SPI1_SCK /<br>I2S1_CK   |               | TMR2_CH1_EXT | USART3_CK            | USART3_RX  |            |                     |                       |
| PA6 | SPI1_MISO /<br>I2S1_MCK | TMR3_CH1      | TMR1_BRK     | USART3_RX            | USART3_CTS | TMR16_CH1  | I2S2_MCK            | TMR13_CH1             |



| Pin  | MUX0                | MUX1          | MUX2         | MUX3              | MUX4                 | MUX5       | MUX6                 | MUX7              |
|------|---------------------|---------------|--------------|-------------------|----------------------|------------|----------------------|-------------------|
| PA7  | SPI1_MOSI / I2S1_SD | TMR3_CH2      | TMR1_CH1N    | USART3_TX         | TMR14_CH1            | TMR17_CH1  | EVENTOUT             | I2C2_SCL          |
| PA8  | CLKOUT              | USART1_CK     | TMR1_CH1     | OTG_FS_SOF        | USART2_TX            | EVENTOUT   |                      | I2C2_SCL          |
| PA9  | TMR15_BRK           | USART1_TX     | TMR1_CH2     | OTG_FS_VBUS       | I2C1_SCL             | CLKOUT     | SPI3_SCK / I2S3_CK   | I2C2_SMBA         |
| PA10 | TMR17_BRK           | USART1_RX     | TMR1_CH3     | OTG_FS_ID         | I2C1_SDA             | RTC_REFIN  | SPI3_MOSI / I2S3_SD  |                   |
| PA11 | EVENTOUT            | USART1_CTS    | TMR1_CH4     | SPI3_CS / I2S3_WS | CAN_RX               | I2C2_SCL   | I2C1_SMBA            |                   |
| PA12 | EVENTOUT            | USART1_RTS_DE | TMR1_EXT     |                   | CAN_TX               | I2C2_SDA   | SPI3_MISO / I2S3_MCK |                   |
| PA13 | SWDIO               | IR_OUT        | OTG_FS_NOE   | I2Sext_SD         | SPI3_MISO / I2S3_MCK | I2C1_SDA   | SPI2_MISO / I2S2_MCK |                   |
| PA14 | SWCLK               | USART2_TX     |              |                   | SPI3_MOSI / I2S3_SD  | I2C1_SMBA  | SPI2_MOSI / I2S2_SD  |                   |
| PA15 | SPI1_CS / I2S1_WS   | USART2_RX     | TMR2_CH1_EXT | EVENTOUT          | USART4_RTS_DE        | OTG_FS_NOE | SPI2_CS / I2S2_WS    | SPI3_CS / I2S3_WS |

**Table 17. Port B multiplexed function configuration with GPIOB\_MUX\* register**

| Pin  | MUX0                 | MUX1     | MUX2       | MUX3      | MUX4                | MUX5               | MUX6                 | MUX7                 |
|------|----------------------|----------|------------|-----------|---------------------|--------------------|----------------------|----------------------|
| PB0  | EVENTOUT             | TMR3_CH3 | TMR1_CH2N  | USART2_RX | USART3_CK           | USART3_RTS_DE      | I2S1_MCK             | SPI1_MISO / I2S1_MCK |
| PB1  | TMR14_CH1            | TMR3_CH4 | TMR1_CH3N  | USART2_CK | USART3_RTS_DE       | USART3_CTS         | SPI2_SCK / I2S2_CK   | SPI1_MOSI / I2S1_SD  |
| PB2  |                      | TMR3_EXT |            |           | SPI3_MOSI / I2S3_SD |                    |                      | I2C1_SMBA            |
| PB3  | SPI1_SCK / I2S1_CK   | EVENTOUT | TMR2_CH2   |           | USART1_RTS_DE       | USART2_CTS         | SPI2_SCK / I2S2_CK   | SWO                  |
| PB4  | SPI1_MISO / I2S1_MCK | TMR3_CH1 | EVENTOUT   | I2Sext_SD | USART1_CTS          | TMR17_BRK          | SPI2_MISO / I2S2_MCK | I2C1_SDA             |
| PB5  | SPI1_MOSI / I2S1_SD  | TMR3_CH2 | TMR16_BRK  | I2C1_SMBA | USART1_CK           | USART2_RTS_DE      | SPI2_MOSI / I2S2_SD  |                      |
| PB6  | USART1_TX            | I2C1_SCL | TMR16_CH1N |           | USART4_CK           |                    | I2S1_MCK             | SPI3_CS / I2S3_WS    |
| PB7  | USART1_RX            | I2C1_SDA | TMR17_CH1N |           | USART4_CTS          |                    |                      | SPI3_SCK / I2S3_CK   |
| PB8  | USART1_TX            | I2C1_SCL | TMR16_CH1  | EVENTOUT  | CAN_RX              |                    |                      | SPI3_MISO / I2S3_MCK |
| PB9  | IR_OUT               | I2C1_SDA | TMR17_CH1  | EVENTOUT  | CAN_TX              | SPI2_CS / I2S2_WS  | I2S1_MCK             | SPI3_MOSI / I2S3_SD  |
| PB10 |                      | I2C2_SCL | TMR2_CH3   |           | USART3_TX           | SPI2_SCK / I2S2_CK |                      |                      |
| PB11 | EVENTOUT             | I2C2_SDA | TMR2_CH4   |           | USART3_RX           |                    |                      |                      |
| PB12 | SPI2_CS / I2S2_WS    | EVENTOUT | TMR1_BRK   |           | USART3_CK           | TMR15_BRK          | SPI3_CS / I2S3_WS    | I2C2_SMBA            |

| Pin  | MUX0                 | MUX1       | MUX2      | MUX3       | MUX4          | MUX5      | MUX6                 | MUX7 |
|------|----------------------|------------|-----------|------------|---------------|-----------|----------------------|------|
| PB13 | SPI2_SCK / I2S2_CK   | TMR15_CH1N | TMR1_CH1N | CLKOUT     | USART3_CTS    | I2C2_SCL  | SPI3_SCK / I2S3_CK   |      |
| PB14 | SPI2_MISO / I2S2_MCK | TMR15_CH1  | TMR1_CH2N | I2Sext_SD  | USART3_RTS_DE | I2C2_SDA  | SPI3_MISO / I2S3_MCK |      |
| PB15 | SPI2_MOSI / I2S2_SD  | TMR15_CH2  | TMR1_CH3N | TMR15_CH1N |               | RTC_REFIN | SPI3_MOSI / I2S3_SD  |      |

**Table 18. Port C multiplexed function configuration with GPIOC\_MUX\* register**

| Pin  | MUX0      | MUX1                 | MUX2     | MUX3      | MUX4                 | MUX5                | MUX6                | MUX7     |
|------|-----------|----------------------|----------|-----------|----------------------|---------------------|---------------------|----------|
| PC0  | EVENTOUT  | I2C2_SCL             |          |           |                      |                     |                     | I2C1_SCL |
| PC1  | EVENTOUT  | I2C2_SDA             |          |           | SPI3_MOSI / I2S3_SD  | SPI1_MOSI / I2S1_SD | SPI2_MOSI / I2S2_SD | I2C1_SDA |
| PC2  | EVENTOUT  | SPI2_MISO / I2S2_MCK |          | I2Sext_SD |                      |                     |                     |          |
| PC3  | EVENTOUT  | SPI2_MOSI / I2S2_SD  |          |           |                      |                     |                     |          |
| PC4  | EVENTOUT  | USART3_TX            |          |           | TMR13_CH1            |                     | I2S1_MCK            |          |
| PC5  |           | USART3_RX            |          |           |                      |                     |                     |          |
| PC6  | TMR3_CH1  | I2C1_SCL             | TMR1_CH1 |           |                      | I2S2_MCK            |                     |          |
| PC7  | TMR3_CH2  | I2C1_SDA             | TMR1_CH2 |           |                      | I2S2_MCK            | SPI2_SCK / I2S2_CK  |          |
| PC8  | TMR3_CH3  |                      | TMR1_CH3 |           |                      |                     |                     |          |
| PC9  | TMR3_CH4  | I2C2_SDA             | TMR1_CH4 |           |                      | OTG_FS_NOE          |                     | I2C1_SDA |
| PC10 | USART4_TX | USART3_TX            |          |           | SPI3_SCK / I2S3_CK   |                     |                     |          |
| PC11 | USART4_RX | USART3_RX            |          | I2Sext_SD | SPI3_MISO / I2S3_MCK |                     |                     |          |
| PC12 | USART4_CK | USART3_CK            |          |           | SPI3_MOSI / I2S3_SD  |                     |                     |          |
| PC13 |           |                      |          |           |                      |                     |                     |          |
| PC14 |           |                      |          |           |                      |                     |                     |          |
| PC15 |           |                      |          |           |                      |                     |                     |          |

**Table 19. Port D multiplexed function configuration with GPIOD\_MUX\* register**

| Pin | MUX0     | MUX1          | MUX2 | MUX3 | MUX4 | MUX5 | MUX6 | MUX7 |
|-----|----------|---------------|------|------|------|------|------|------|
| PD2 | TMR3_EXT | USART3_RTS_DE |      |      |      |      |      |      |

**Table 20. Port F multiplexed function configuration with GPIOE\_MUX\* register**

| Pin | MUX0 | MUX1     | MUX2      | MUX3 | MUX4 | MUX5 | MUX6              | MUX7 |
|-----|------|----------|-----------|------|------|------|-------------------|------|
| PF0 |      |          | TMR1_CH1  |      |      |      |                   |      |
| PF1 |      |          | TMR1_CH2N |      |      |      | SPI2_CS / I2S2_WS |      |
| PF4 |      | I2C1_SDA | TMR2_CH1  |      |      |      |                   |      |

| Pin | MUX0     | MUX1     | MUX2     | MUX3 | MUX4      | MUX5 | MUX6 | MUX7 |
|-----|----------|----------|----------|------|-----------|------|------|------|
| PF5 |          | I2C1_SCL | TMR2_CH2 |      |           |      |      |      |
| PF6 | I2C2_SCL |          |          |      | USART4_RX |      |      |      |
| PF7 | I2C2_SDA |          |          |      | USART4_TX |      |      |      |

Note: EVENTOUT represents the TXEV signal of Cortex-M.

## Peripheral MUX function configuration

IOMUX function configuration as follows:

- To use a peripheral pin in MUX output, it is configured as multiplexed push-pull/open-drain output.
- To use a peripheral pin in MUX input, it is configured as floating input/pull-up/pull-down input.
- For ADC peripherals, the pins of analog channels should be configured as analog mode.
- For I<sup>2</sup>C peripherals that intend to use pins as bidirectional functions, open-drain mode is required.
- For USB peripherals, configure corresponding IOMUX and enable corresponding clocks in CRM, there is no need of GPIO status configuration

## IOMUX mapping priority

The unique peripheral multiplexed function can be configured through the GPIOx\_MUXL/GPIOx\_MUXH register, except individual pins that may be directly owned by hardware.

Some pins have been directly owned by specific hardware feature, whatever GPIO configuration.

**Table 21. Pins owned by hardware**

| Pin  | Enable bit  | Description   |
|------|---|---|
| PA0  | PWC_CTRLSTS[8] = 1  | Once enabled, PA0 pin acts as WKUP1 of PWC.                                 |
| PA2  | PWC_CTRLSTS[11] = 1   | Once enabled, PA2 pin acts as WKUP4 of PWC                                  |
| PB5  | PWC_CTRLSTS[13] = 1   | Once enabled, PB5 pin acts as WKUP6 of PWC                                  |
| PB15 | PWC_CTRLSTS[14] = 1   | Once enabled, PB15 pin acts as WKUP7 of PWC                                 |
| PC5  | PWC_CTRLSTS[12] = 1   | Once enabled and PC13 is not occupied, the PC15 can be used as WKUP5 of PWC |
| PC13 | (PWC_CTRLSTS[9] = 1) & (ERTC_CTRL[23:21] = 3'b000) & (ERTC_CTRL[11] = 0) & (ERTC_TAMP[0] = 0) | Once enabled, PC13 pin acts as WKUP2 of PWC                                 |
| PC13 | (ERTC_CTRL[23: 21] != 3'b000)   (ERTC_CTRL[11] != 0)   (ERTC_TAMP[0] != 0)                    | Once enabled, the PC13 is used as RTC channel input and output              |
| PC14 | CRM_BPDC[0] = 1   | Once enabled, the PC14 is used as LEXT channel                              |
| PC15 | CRM_BPDC[0] = 1   | Once enabled, the PC15 is used as LEXT channel                              |
| PF0  | CRM_CTRL[16] = 1  | Once enabled, the PF0 is used as HEXT channel                               |
| PF1  | CRM_CTRL[16] = 1 & CRM_CTRL[18] = 0   | Once enabled, the PF1 is used as HEXT channel                               |

## External interrupt/wake-up lines

Each pin can be used as an external interrupt input. The corresponding pin should be configured as input mode.

## GPIO registers

Table 22 lists GPIO register map and their reset values. These peripheral registers must be accessed by words (32 bits).

**Table 22. GPIO register map and reset values**

| Register                    | Offset | Reset value                   |
|-----------------------------|--------|-------------------------------|
| GPIOA_CFGR                  | 0x00   | 0x2800 0000                   |
| GPIOx_CFGR (x = B, C, D, F) | 0x00   | 0x0000 0000                   |
| GPIOx_OMODER                | 0x04   | 0x0000 0000                   |
| GPIOx_ODRVR                 | 0x08   | 0x0C00 0000(A)<br>0x0000 0000 |
| GPIOA_PULL                  | 0x0C   | 0x2400 0000(A)                |
| GPIOx_PULL (x = B, C, D, F) | 0x0C   | 0x0000 0000                   |
| GPIOx_IDT                   | 0x10   | 0x0000 XXXX                   |
| GPIOx_ODT                   | 0x14   | 0x0000 0000                   |
| GPIOx_SCR                   | 0x18   | 0x0000 0000                   |
| GPIOx_WPR                   | 0x1C   | 0x0000 0000                   |
| GPIOx_MUXL                  | 0x20   | 0x0000 0000                   |
| GPIOx_MUXH                  | 0x24   | 0x0000 0000                   |
| GPIOx_CLR                   | 0x28   | 0x0000 0000                   |
| GPIOx_HDRV                  | 0x3C   | 0x0000 0000                   |

### GPIO configuration register (GPIOx\_CFGR) (x = A/B/C/D/F)

Address offset: 0x00

Reset values: 0x28000000 for port A, 0x00000000 for other ports

| Bit         | Name  | Reset value | Type | Description  |
|-------------|-------|-------------|------|--|
| Bit 2y+1:2y | IOMCy | 0x2800 0000 | rw   | GPIOx mode configuration (y = 0 ~ 15)<br>00: Input mode (reset state)<br>01: General-purpose output mode<br>10: Multiplexed function mode<br>11: Analog mode |

### GPIO output mode register (GPIOx\_OMODER) (x = A/B/C/D/F)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Always 0.   |
| Bit 15:0  | OM       | 0x0000      | rw   | GPIOx output mode configuration (x = 0 ~ 15)<br>These field is used to configure the output mode of the GPIOx:<br>0: Push-pull (reset state)<br>1: Open-drain |

## GPIO drive capability register (GPIOx\_ODRVR) (x = A/B/C/D/F)

Address offset: 0x08

Reset values: 0x0C00 0000 for port A, 0x00000000 for other ports

| Bit          | Name  | Reset value | Type | Description   |
|--------------|-------|-------------|------|---|
| Bit 2y+1: 2y | ODRVy | 0x0000 0000 | rw   | GPIOx drive capability (y = 0...15)<br>This field is used to configure the IO port drive capability.<br>x0: Normal sourcing/sinking strength<br>01: Large sourcing/sinking strength<br>11: Normal sourcing/sinking strength |

## GPIO pull-up/pull-down register (GPIOx\_PULL) (x = A/B/C/D/F)

Address offset: 0x0C

Reset values: 0x2400 0000 for port A, 0x00000000 for other ports

| Bit          | Name  | Reset value | Type | Description  |
|--------------|-------|-------------|------|--|
| Bit 2y+1: 2y | PULLy | 0x6400 0000 | rw   | GPIOx pull-up/pull-down configuration (y = 0...15)<br>This field is used to configure the pull-up/pull-down of the IO port.<br>00, 11: No pull-up, pull-down<br>01: Pull-up<br>10: Pull-down |

## GPIO input register (GPIOx\_IDT) (x = A/B/C/D/F)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Always 0.   |
| Bit 15:0  | IDT      | 0xFFFF      | ro   | GPIOx input data<br>Indicates the input status of I/O port. Each bit corresponds to an I/O. |

## GPIO output register (GPIOx\_ODT) (x = A/B/C/D/F)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Always 0.  |
| Bit 15:0  | ODT      | 0x0000      | rw   | GPIOx output data<br>Each bit represents an I/O port.<br>GPIOx indicates the output status of I/O port.<br>0: Low<br>1: High |

### GPIO set/clear register (GPIOx\_SCR) (x = A/B/C/D/F)

| Bit       | Name | Reset value | Type | Description   |
|-----------|------|-------------|------|---|
| Bit 31:16 | IOCB | 0x0000      | wo   | GPIOx clear bit<br>The corresponding ODT register bit is cleared by writing “1” to these bits. Otherwise, the corresponding ODT register bit remains unchanged, which acts as ODT register bit operations.<br>0: No action to the corresponding ODT bits<br>1: Clear the corresponding ODT bits |
| Bit 15:0  | IOSB | 0x0000      | wo   | GPIOx set bit<br>The corresponding ODT register bit is set by writing “1” to these bits. Otherwise, the corresponding ODT register bit remains unchanged, which acts as ODT register bit operations.<br>0: No action to the corresponding ODT bits<br>1: Set the corresponding ODT bits         |

### GPIO write protection register (GPIOx\_WPR) (x = A/B/C/D/F)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:17 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 16    | WPSEQ    | 0x0         | rw   | Write protect sequence<br>Write protect enable sequence bit and WPEN bit must be enabled at the same time to achieve write protection for some I/O bits.<br>Write protect enable bit is executed four times in the order below: write “1” → write “0” → write “1” → read. Note that the value of WPEN bit cannot be modified during this period. |
| Bit 15:0  | WPEN     | 0x0000      | rw   | Write protect enable<br>Each bit corresponds to an I/O port.<br>0: No effect.<br>1: Write protect  |

### GPIO multiplexed function low register (GPIOx\_MUXL) (x = A/B/C/D/F)

Address offset: 0x20

Reset value: 0x00000000

| Bit         | Name  | Reset value | Type | Description  |
|-------------|-------|-------------|------|--|
| Bit 4y+3:4y | MUXLy | 0x0         | rw   | Multiplexed function select for GPIOx pin y (y = 0...7)<br>This field is used to configure multiplexed function I/Os.<br>0000: MUX0<br>0001: MUX1<br>0010: MUX2<br>0011: MUX3<br>0100: MUX4<br>0101: MUX5<br>0110: MUX6<br>0111: MUX7<br>1000 ~ 1111: Reserved |

### GPIO multiplexed function high register (GPIOx\_MUXH) (x = A/B/C/D/F)

| Bit         | Name  | Reset value | Type | Description   |
|-------------|-------|-------------|------|---|
| Bit 4y+3:4y | MUXHy | 0x0         | rw   | <p>Multiplexed function select for GPIOx pin y (y = 8...15)</p> <p>This field is used to configure multiplexed function IOs 0000: MUX0</p> <p>0001: MUX1</p> <p>0010: MUX2</p> <p>0011: MUX3</p> <p>0100: MUX4</p> <p>0101: MUX5</p> <p>0110: MUX6</p> <p>0111: MUX7</p> <p>1000 ~ 1111: Reserved</p> |

### GPIO port bit clear register (GPIOx\_CLR) (x = A/B/C/D/F)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 15:0  | IOCB     | 0x0000      | wo   | <p>GPIOx clear bit</p> <p>The corresponding ODT register bit is cleared by writing "1" to these bits. Otherwise, the corresponding ODT register bit remains unchanged, which acts as ODT register bit operations.</p> <p>0: No action to the corresponding ODT bits</p> <p>1: Clear the corresponding ODT bits</p> |

### GPIO huge current control register (GPIOx\_HDRV) (x = A/B/C/D/F)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 15:0  | HDRV     | 0x0000      | rw   | <p>Huge sourcing/sinking strength control</p> <p>0: Not active</p> <p>1: GPIO is configured as maximum sourcing/sinking strength</p> |

# 7 System configuration controller (SCFG)

## Introduction

This device contains a set of system configuration register. The system configuration controller is mainly used to:

- Manage the external interrupts connected to the GPIOs
- Control the memory mapping mode
- Manage IRTMR GPIO configurations

## SCFG registers

Table 23 shows SCFG register map and their reset values.

These peripheral registers must be accessed by words (32 bits).

**Table 23. SCFG register map and reset values**

| Register     | Offset | Reset value |
|--------------|--------|-------------|
| SCFG_CFG1    | 0x00   | 0x0000 000X |
| SCFG_EXINTC1 | 0x08   | 0x0000 0000 |
| SCFG_EXINTC2 | 0x0C   | 0x0000 0000 |
| SCFG_EXINTC3 | 0x10   | 0x0000 0000 |
| SCFG_EXINTC4 | 0x14   | 0x0000 0000 |
| SCFG_CFG2    | 0x18   | 0x0000 0000 |

### SCFG configuration register 1 (SCFG\_CFG1)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:20 | Reserved | 0x000       | resd | Kept at its default value.   |
| Bit 19    | PB14_UH  | 0x0         | rw   | <p>PB14 Ultra high sourcing/sinking strength</p> <p>This bit is written by software to control the PB14 PAD sourcing/sinking strength.</p> <p>0: Not active</p> <p>1: Corresponding GPIO is switched to ultra high sourcing/sinking strength</p> <p>When this bit is set, the control bits of GPIOx_OTYPER &amp; GPIOx_HDRV become invalid</p> |
| Bit 18    | PB13_UH  | 0x0         | rw   | <p>PB13 Ultra high sourcing/sinking strength</p> <p>This bit is written by software to control the PB13 PAD sourcing/sinking strength.</p> <p>0: Not active</p> <p>1: Corresponding GPIO is switched to ultra high sourcing/sinking strength</p> <p>When this bit is set, the control bits of GPIOx_OTYPER &amp; GPIOx_HDRV become invalid</p> |



| Bit      | Name        | Reset value | Type | Description   |
|----------|-------------|-------------|------|---|
| Bit 17   | PB9_UH      | 0x0         | rw   | PB9 Ultra high sourcing/sinking strength<br>This bit is written by software to control the PB9 PAD sourcing/sinking strength.<br>0: Not active<br>1: Corresponding GPIO is switched to ultra high sourcing/sinking strength<br>When this bit is set, the control bits of GPIOx_OTYPER & GPIOx_HDRV become invalid |
| Bit 16   | PB8_UH      | 0x0         | rw   | PB8 Ultra high sourcing/sinking strength<br>This bit is written by software to control the PB8 PAD sourcing/sinking strength.<br>0: Not active<br>1: Corresponding GPIO is switched to ultra high sourcing/sinking strength<br>When this bit is set, the control bits of GPIOx_OTYPER & GPIOx_HDRV become invalid |
| Bit 15:8 | Reserved    | 0x0         | resd | Kept at its default value.  |
| Bit 7:6  | IR_SRC_SEL  | 0x0         | rw   | Infrared modulation envelope signal source selection<br>This field is used to select the infrared modulation envelope signal source.<br>00: TMR16<br>01: USART1<br>10: USART4<br>11: Reserved   |
| Bit 5    | IR_POL      | 0           | rw   | Infrared output polarity selection<br>0: Infrared output (IR_OUT) is not inversed<br>1: Infrared output (IR_OUT) is inversed  |
| Bit 4    | PA11_12_PMP | 0x0         | rw   | PA11 and PA12 remap<br>This bit is set and cleared by software. This bit controls PA9/PA10 and PA11/PA12 remap on small packages (20-pin).<br>0: No remap (PA9/PA10 corresponds to PA9/PA10)<br>1: Remap (PA11/PA12 is mapped onto PA9/PA10)  |
| Bit 3:2  | Reserved    | 0x0         | resd | Kept at its default value.  |
| Bit 1:0  | MEM_MAP_SEL | 0xX         | rw   | Memory address mapping selection<br>This field is read-only, indicating the boot mode after reset.<br>X0: Boot from main Flash memory<br>01: Boot from system memory<br>11: Boot from internal SRAM   |

### SCFG external interrupt configuration register 1 (SCFG\_EXINTC1)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 15:12 | EXINT3   | 0x0         | rw   | EXINT3 input source configuration<br>These bits are used to select the input source for the EXINT3 external interrupt.<br>0000: GPIOA pin3<br>0001: GPIOB pin3<br>0010: GPIOC pin3<br>Others: Reserved |

| Bit      | Name   | Reset value | Type | Description  |
|----------|--------|-------------|------|--|
| Bit 11:8 | EXINT2 | 0x0         | rw   | EXINT2 input source configuration<br>These bits are used to select the input source for the EXINT2 external interrupt.<br>0000: GPIOA pin2<br>0001: GPIOB pin2<br>0010: GPIOC pin2<br>0011: GPIOD pin2<br>Others: Reserved |
| Bit 7:4  | EXINT1 | 0x0         | rw   | EXINT1 input source configuration<br>These bits are used to select the input source for the EXINT1 external interrupt.<br>0000: GPIOA pin1<br>0001: GPIOB pin1<br>0010: GPIOC pin1<br>0101: GPIOF pin1<br>Others: Reserved |
| Bit 3:0  | EXINT0 | 0x0         | rw   | EXINT0 input source configuration<br>These bits are used to select the input source for the EXINT0 external interrupt.<br>0000: GPIOA pin0<br>0001: GPIOB pin0<br>0010: GPIOC pin0<br>0101: GPIOF pin0<br>Others: Reserved |

### SCFG external interrupt configuration register 2 (SCFG\_EXINTC2)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 15:12 | EXINT7   | 0x0         | rw   | EXINT7 input source configuration<br>These bits are used to select the input source for the EXINT7 external interrupt.<br>0000: GPIOA pin7<br>0001: GPIOB pin7<br>0010: GPIOC pin7<br>0101: GPIOF pin7<br>Others: Reserved |
| Bit 11:8  | EXINT6   | 0x0         | rw   | EXINT6 input source configuration<br>These bits are used to select the input source for the EXINT6 external interrupt.<br>0000: GPIOA pin6<br>0001: GPIOB pin6<br>0010: GPIOC pin6<br>0101: GPIOF pin6<br>Others: Reserved |
| Bit 7:4   | EXINT5   | 0x0         | rw   | EXINT5 input source configuration<br>These bits are used to select the input source for the EXINT5 external interrupt.<br>0000: GPIOA pin5<br>0001: GPIOB pin5<br>0010: GPIOC pin5<br>0101: GPIOF pin5<br>Others: Reserved |

| Bit     | Name   | Reset value | Type | Description  |
|---------|--------|-------------|------|--|
| Bit 3:0 | EXINT4 | 0x0         | rw   | EXINT4 input source configuration<br>These bits are used to select the input source for the EXINT4 external interrupt.<br>0000: GPIOA pin4<br>0001: GPIOB pin4<br>0010: GPIOC pin4<br>0101: GPIOF pin4<br>Others: Reserved |

### SCFG external interrupt configuration register 3 (SCFG\_EXINTC3)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.  |
| Bit 15:12 | EXINT11  | 0x0         | rw   | EXINT11 input source configuration<br>These bits are used to select the input source for the EXINT11 external interrupt.<br>0000: GPIOA pin11<br>0001: GPIOB pin11<br>0010: GPIOC pin11<br>Others: Reserved |
| Bit 11:8  | EXINT10  | 0x0         | rw   | EXINT10 input source configuration<br>These bits are used to select the input source for the EXINT10 external interrupt.<br>0000: GPIOA pin10<br>0001: GPIOB pin10<br>0010: GPIOC pin10<br>Others: Reserved |
| Bit 7:4   | EXINT9   | 0x0         | rw   | EXINT9 input source configuration<br>These bits are used to select the input source for the EXINT9 external interrupt.<br>0000: GPIOA pin9<br>0001: GPIOB pin9<br>0010: GPIOC pin9<br>Others: Reserved      |
| Bit 3:0   | EXINT8   | 0x0         | rw   | EXINT8 input source configuration<br>These bits are used to select the input source for the EXINT8 external interrupt.<br>0000: GPIOA pin8<br>0001: GPIOB pin8<br>0010: GPIOC pin8<br>Others: Reserved      |

### SCFG external interrupt configuration register 4 (SCFG\_EXINTC4)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value   |
| Bit 15:12 | EXINT15  | 0x0         | rw   | EXINT15 input source configuration<br>These bits are used to select the input source for the EXINT15 external interrupt.<br>0000: GPIOA pin15<br>0001: GPIOB pin15<br>0010: GPIOC pin15<br>Others: Reserved |
| Bit 11:8  | EXINT14  | 0x0         | rw   | EXINT14 input source configuration<br>These bits are used to select the input source for the EXINT14 external interrupt.<br>0000: GPIOA pin14<br>0001: GPIOB pin14<br>0010: GPIOC pin14<br>Others: Reserved |
| Bit 7:4   | EXINT13  | 0x0         | rw   | EXINT13 input source configuration<br>These bits are used to select the input source for the EXINT13 external interrupt.<br>0000: GPIOA pin13<br>0001: GPIOB pin13<br>0010: GPIOC pin13<br>Others: Reserved |
| Bit 3:0   | EXINT12  | 0x0         | rw   | EXINT12 input source configuration<br>These bits are used to select the input source for the EXINT12 external interrupt.<br>0000: GPIOA pin12<br>0001: GPIOB pin12<br>0010: GPIOC pin12<br>Others: Reserved |

### SCFG configuration register 2 (SCFG\_CFG2)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:30 | I2S_FD   | 0x00        | resd | I <sup>2</sup> S full duplex<br>Using this field, any two of I <sup>2</sup> S can be configured as full-duplex mode. If not needed, this field must remain 00 to avoid unexpected results. See 13.3.2 for more information.<br>00: SPI/I <sup>2</sup> S1 ~ 3 works independently<br>01: Combine I <sup>2</sup> S1 and I <sup>2</sup> S3 into full-duplex mode<br>10: Combine I <sup>2</sup> S2 and I <sup>2</sup> S3 into full-duplex mode<br>11: Combine I <sup>2</sup> S1 and I <sup>2</sup> S2 into full-duplex mode |
| Bit 29:3  | Reserved | 0x0000 000  | resd | Kept at its default value.  |
| Bit 2     | PVM_LK   | 0           | rw   | PVM lock enable<br>0: Disconnect PVM interrupt from TMR1/TMR15/16/17 brake input. PVMSEL and PVMEN bits can be changed by software.<br>1: Connect PVM interrupt and TMR1/TMR15/16/17 brake input. PVMSEL and PVMEN bits are read-only, unchangeable.  |
| Bit 1:0   | Reserved | 0x0         | resd | Kept at its default value.  |

## 8 External interrupt/Event controller (EXINT)

### EXINT introduction

EXINT consists of 22 interrupt lines EXINT\_LINE[23:0], each of which can generate an interrupt or event by edge detection trigger or software trigger. EXINT can enable or disable an interrupt or event independently through software configuration, and utilizes different edge detection modes (rising edge, falling edge or both edges) as well as trigger modes (edge detection, software trigger or both triggers) to respond to the trigger source in order to generate an interrupt or event.

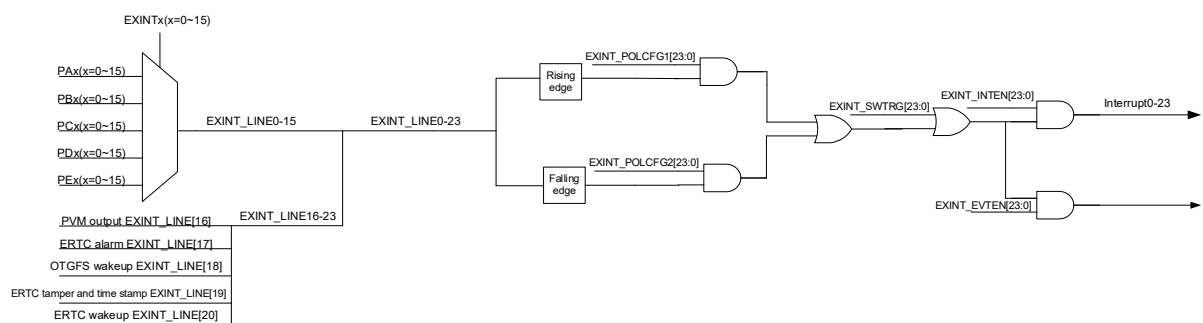


Figure 20. External interrupt/Event controller block diagram

Main features:

- EXINT 0 ~ 15 mapping IO can be configured independently
- Independent trigger selection on each interrupt line
- Independent enable bit on each interrupt
- Independent enable bit on each event
- Up to 22 software trigger that can be generated and cleared independently
- Independent status bit on each interrupt
- Each interrupt can be cleared independently.

### Function overview and configuration procedure

With up to 22 interrupt lines EXINT\_LINE[23:0], EXINT can detect not only GPIO external interrupt sources but also six internal sources such as PVM output, ERTC alarm, OTGFS wakeup, RTC wakeup, RTC tamper and time stamp events, RTC alarm and I2C1 wakeup events. The GPIO interrupt sources can be selected with SCFG\_EXINTCx register. It should be noted that these input sources are mutually exclusive. For example, EXINT\_LINE0 is allowed to select only one of PA0/PB0/PC0/PD0 pins, instead of taking both PA0 and PB0 as the input sources at the same time.

EXINT supports multiple edge detection modes, including rising edge, falling edge or both edges, selected by EXINT\_POLCFG1 and EXINT\_POLCFG2 register. Active edge trigger detected on the interrupt line can be used to generate an event or interrupt.

In addition, EXINT supports independent software trigger for the generation of an event or interrupt. This is achieved by setting the corresponding bits in the EXINT\_SWTRG register.

EXINT can enable or disable an interrupt or event individually through software configuration such as EXINT\_INTEN and EXINT\_EVTEN register, indicating that the corresponding interrupt or event must be enabled prior to either edge detection or software trigger.

EXINT also features an independent interrupt status bit. Reading access to EXINT\_INTSTS register can obtain the corresponding interrupt status. The status flag is cleared by writing “1” to this register.

#### Interrupt initialization procedure

1. Select an interrupt source by setting SCFG\_EXINTCx register (This is required if GPIO is used as an interrupt source).
2. Select an trigger mode by setting EXINT\_POLCFG1 and EXINT\_POLCFG2 register.
3. Enable interrupt or event by setting EXINT\_INTEN and EXINT\_EVTEN register.
4. Generate software trigger by setting EXINT\_SWTRG register (This is applied to software trigger interrupt only).

Note: If there is a need to modify interrupt source configuration, then switch off interrupt enable register and event enable register first before re-starting interrupt initialization configuration.

#### Interrupt clear procedure

- Writing “1” to the EXINT\_INTSTS register to clear the interrupts generated, and the corresponding bits in the EXINT\_SWTRG register.

## EXINT registers

These peripheral registers must be accessed by words (32 bits).

Table 24 shows EXINT register map and their reset value.

**Table 24. External interrupt/event controller register map and reset value**

| Register      | Offset | Reset value |
|---------------|--------|-------------|
| EXINT_INTEN   | 0x00   | 0x0000 0000 |
| EXINT_EVTEN   | 0x04   | 0x0000 0000 |
| EXINT_POLCFG1 | 0x08   | 0x0000 0000 |
| EXINT_POLCFG2 | 0x0C   | 0x0000 0000 |
| EXINT_SWTRG   | 0x10   | 0x0000 0000 |
| EXINT_INTSTS  | 0x14   | 0x0000 0000 |

#### Interrupt enable register (EXINT\_INTEN)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:24 | Reserved | 0x000       | resd | Forced to 0 by hardware.   |
| Bit 23:0  | INTENx   | 0x00000     | rw   | Interrupt enable or disable on line x<br>0: Interrupt request is disabled.<br>1: Interrupt request is enabled.<br>Note: Bit 21 and bit 22 are reserved and unused. |

### Event enable register (EXINT\_EVTEN)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:24 | Reserved | 0x000       | resd | Forced to 0 by hardware.   |
| Bit 23:0  | EVTENx   | 0x00000     | rw   | Event enable or disable on line x<br>0: Event request is disabled.<br>1: Event request is enabled.<br>Note: Bit 21 and bit 22 are reserved and unused. |

### Polarity configuration register 1 (EXINT\_POLCFG1)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:24 | Reserved | 0x000       | resd | Forced to 0 by hardware.  |
| Bit 23:0  | RPx      | 0x00000     | rw   | Rising polarity configuration bit on line x<br>These bits are used to select a rising edge to trigger an interrupt and event on line x.<br>0: Rising trigger on line x is disabled.<br>1: Rising trigger on line x is enable.<br>Note: Bit 21 and 22 are reserved and unused. |

### Polarity configuration register 2 (EXINT\_POLCFG2)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:24 | Reserved | 0x000       | resd | Forced to be 0 by hardware.  |
| Bit 23:0  | FPx      | 0x00000     | rw   | Falling polarity configuration bit on line x<br>These bits are used to select a falling edge to trigger an interrupt and event on line x.<br>0: Falling trigger on line x is disabled.<br>1: Falling trigger on line x is enabled.<br>Note: Bit 21 and bit 22 are reserved and unused. |

### Software trigger register (EXINT\_SWTRG)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:24 | Reserved | 0x000       | resd | Forced to 0 by hardware.  |
| Bit 23:0  | SWTx     | 0x00000     | rw   | Software trigger on line x<br>If the corresponding bit in EXINT_INTEN register is 1, the software writes to this bit. The hardware sets the corresponding bit in the EXINT_INTSTS automatically to generate an interrupt.<br>If the corresponding bit in the EXINT_EVTEN register is 1, the software writes to this bit. The hardware generates an event on the corresponding interrupt line automatically.<br>0: Default value<br>1: Software trigger generated<br>Note: 1. This bit is cleared by writing 1 to the corresponding bit in the EXINT_INTSTS register.<br>2. Bit 21 and bit 22 are reserved and unused. |

### Interrupt status register (EXINT\_INTSTS)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:24 | Reserved | 0x000       | resd | Forced to 0 by hardware.   |
| Bit 23:0  | LINEx    | 0x00000     | rw   | Line x status bit<br>0: No interrupt occurred.<br>1: Interrupt occurred.<br>Note: Bit 21 and bit 22 are reserved and unused. |



# 9 DMA controller (DMA)

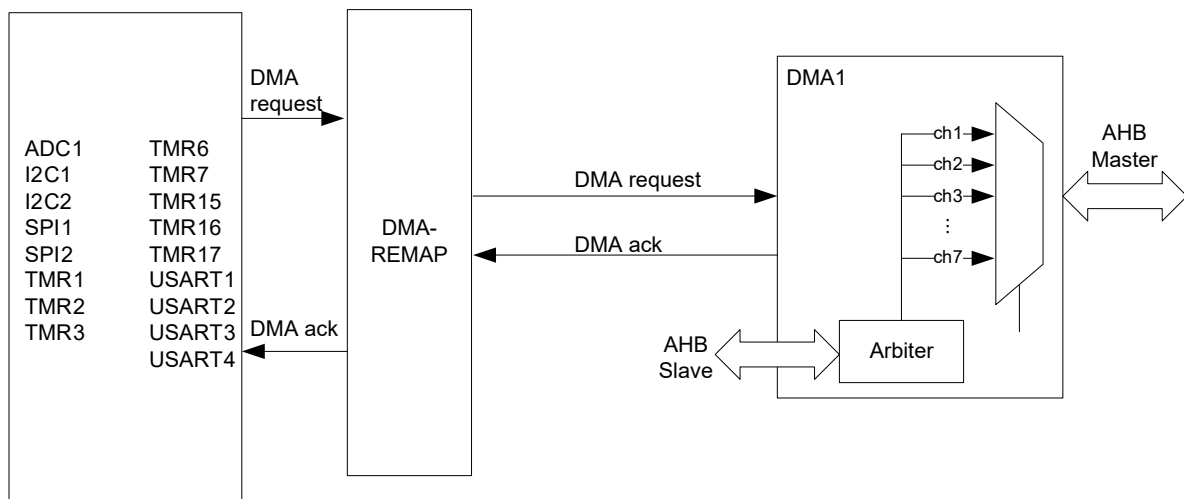
## Introduction

Direct memory access (DMA) controller is designed for 32-bit MCU applications with the aim of enhancing system performance and reducing the generation of interrupts.

One DMA controller is available in the microcontroller. Each controller contains 7 DMA channels. Each channel manages memory access requests from one or more peripherals. An arbiter is available for coordinating the priority of each DMA request.

## Main features

- AMBA compliant (Rev. 2.0)
- Only support AHB OKAY and ERROR responses
- HBUSREQ and HGRANT of AHB master interface are not supported
- Support 7 channels
- Peripheral-to-memory, memory-to-peripheral, and memory-to-memory transfers
- Support hardware handshake
- Support 8-bit, 16-bit and 32-bit data transfers
- Programmable amount of data to be transferred: up to 65535
- Support flexible mapping



**Figure 21. DMA block diagram**

Note: The number of DMA peripherals in Figure 21 may decrease depending on different models.

## Function overview

### DMA configuration

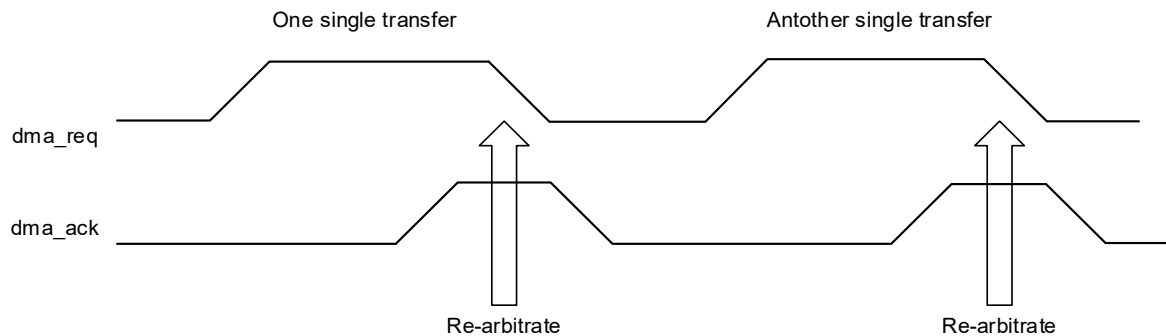
1. Set the peripheral address in the DMA\_CxPADD register  
The initial peripheral address for data transfer remains unchanged during transmission.
2. Set the memory address in the DMA\_CxMADDR register  
The initial memory address for data transfer remains unchanged during transmission.
3. Configure the amount of data to be transferred in the DMA\_CxDTCNT register  
Programmable data transfer size is up to 65535. This value is decremented after each data transfer.
4. Configure the channel setting in the DMA\_CxCTRL register  
Including channel priority, data transfer direction/width, address incremented mode, circular mode and interrupt mode
  - Channel priority (CHPL)  
There are four levels, including very high priority, high priority, medium priority and low priority. If the two channels have the same priority level, then the channel with lower number will get priority over the one with higher number. For example, channel 1 has priority over channel 2.
  - Data transfer direction (DTD)  
Memory-to-peripheral (M2P), peripheral-to-memory (P2M)
  - Address incremented mode (PINCM/MINCM)  
In incremented mode, the subsequent transfer address is the previous address plus transfer width (PWIDTH/MWIDTH).
  - Circular mode (LM)  
In circular mode, the contents in the DMA\_CxDTCNT register is automatically reloaded with the initially programmed value after the completion of the last transfer.
  - Memory-to-memory mode (M2M)  
This mode indicates that DMA channels perform data transfer without requests from peripherals. Circular mode and memory-to-memory mode cannot be used at the same time.
5. In non-M2M mode, configure flexible mapping mode through the DMA\_SRC\_SEL0/1 register, write DMA request ID in the corresponding bit of the corresponding channel.
6. Enable DMA transfer by setting the CHEN bit in the DMA\_CxCTRL register

### Handshake mechanism

In P2M and M2P mode, the peripherals need to send a request signal to the DMA controller. The DMA channel will send the peripheral transfer request (single) until the signal is acknowledged. After the completion of peripheral transmission, the DMA controller sends the acknowledge signal to the peripheral. The peripheral then releases its request as soon as it receives the acknowledge signal. At the same time, the DMA controller releases the acknowledge signal as well.

## Arbiter

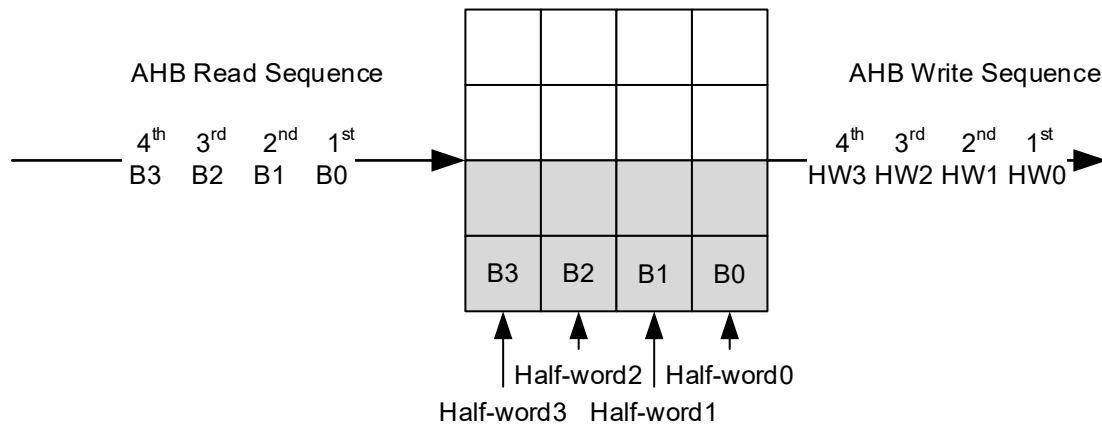
When several channels are enabled simultaneously, the arbiter will restart arbitration after full data transfer by the master controller. The channel with very high priority waits until the channel of the master controller has completed data transfers before taking control of it. The master controller will re-arbitrate to serve other channels as long as the channel completes a single transfer based on the master controller priority.



**Figure 22. Re-arbitrate after request/acknowledge**

## Programmable data transfer width

Transfer width of the source data and destination data is programmable through the PWIDTH and MWIDTH bits in the DMA\_CxCTRL register. When PWIDTH is not equal to MWIDTH, it can be aligned according to the settings of PWIDTH / MWIDTH.



**Figure 23. PWIDTH: byte, MWIDTH: half-word**

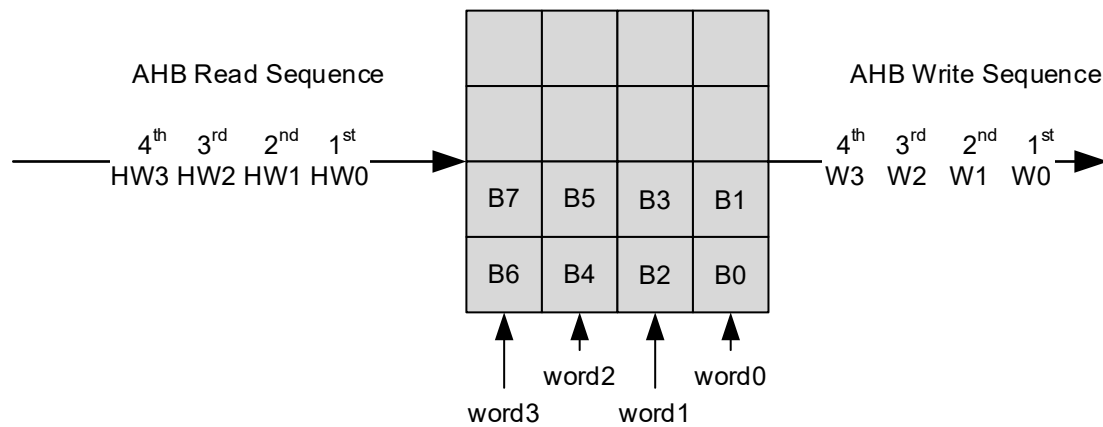


Figure 24. PWIDTH: half-word, MWIDTH: word

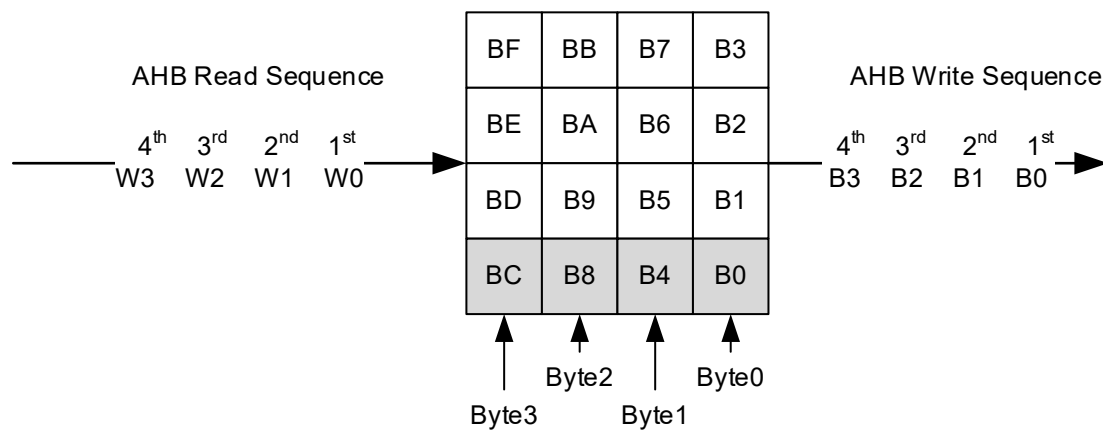


Figure 25. PWIDTH: word, MWIDTH: byte

## Errors

Table 25. DMA error event

| Error event    |  |
|----------------|--|
| Transfer error | AHB response error occurred during DMA read/write access |

## Interrupts

An interrupt can be generated on a DMA half-transfer, transfer complete and transfer error. Each channel has its specific interrupt flag, clear and enable bits, as shown in the table below.

Table 26. DMA interrupt requests

| Interrupt event    | Event flag bit | Clear control bit | Enable control bit |
|--------------------|----------------|-------------------|--------------------|
| Half transfer      | HDTF           | HDTFC             | HDTIEN             |
| Transfer completed | FDTF           | FDTFC             | FDTIEN             |
| Transfer error     | DTERRF         | DTERRFC           | DTERRIEN           |

## Flexible DMA request mapping

In flexible request mode (DMA\_FLEX\_EN = 1), the request source for each channel is selected through the CHx\_SRC register (x = 1 ~ 7). For example, to configure the DMA channel 1 as I2C1\_TX, and channel 3 to I2C1\_RX, others unused, then DMA\_FLEX\_EN = 1, CH1\_SRC = 11, CH3\_SRC = 1, CH[2/4/5/6/7]\_SRC = 0 must be asserted.

Table 27 lists the DMA flexible request sources.

**Table 27. DMA flexible request sources**

| CHx_SRC | Request source | CHx_SRC | DMA source              | CHx_SRC | Request source            | CHx_SRC | Request source |
|---------|----------------|---------|-------------------------|---------|---------------------------|---------|----------------|
| 0       | Unselected     | 16      | SPI1/I2S1_RX            | 32      | TMR3_CH1                  | 48      | —              |
| 1       | —              | 17      | SPI1/I2S1_TX            | 33      | TMR3_CH2                  | 49      | TMR17_OVERFLOW |
| 2       | —              | 18      | SPI2/I2S2_RX            | 34      | TMR3_CH3                  | 50      | USART1_RX      |
| 3       | —              | 19      | SPI2/I2S2_TX            | 35      | TMR3_CH4                  | 51      | USART1_TX      |
| 4       | —              | 20      | TMR1_CH1                | 36      | TMR3_TRIG                 | 52      | USART2_RX      |
| 5       | ADC1           | 21      | TMR1_CH2                | 37      | TMR3_OVERFLOW             | 53      | USART2_TX      |
| 6       | —              | 22      | TMR1_CH3                | 38      | TMR6_OVERFLOW             | 54      | USART3_RX      |
| 7       | —              | 23      | TMR1_CH4                | 39      | TMR7_OVERFLOW             | 55      | USART3_TX      |
| 8       | —              | 24      | TMR1_TRIG/<br>TMR1_HALL | 40      | TMR15_CH1                 | 56      | USART4_RX      |
| 9       | —              | 25      | TMR1_OVERFLOW           | 41      | TMR15_CH2                 | 57      | USART4_TX      |
| 10      | I2C1_RX        | 26      | TMR2_CH1                | 42      | TMR15_TRIG/<br>TMR15_HALL | 58      | —              |
| 11      | I2C1_TX        | 27      | TMR2_CH2                | 43      | TMR15_OVERFLOW            | 59      | —              |
| 12      | I2C2_RX        | 28      | TMR2_CH3                | 44      | TMR16_CH1                 | 60      | SPI3/I2S3_RX   |
| 13      | I2C2_TX        | 29      | TMR2_CH4                | 45      | —                         | 61      | SPI3/I2S3_TX   |
| 14      | —              | 30      | TMR2_TRIG               | 46      | TMR16_OVERFLOW            |         | —              |
| 15      | —              | 31      | TMR2_OVERFLOW           | 47      | TMR17_CH1                 |         | —              |

## DMA registers

Table 28 shows DMA register map and their reset values.

These peripheral registers must be accessed by bytes (8 bits), half-words (16 bits) or words (32 bits).

**Table 28. DMA register map and reset value**

| Register    | Offset | Reset value |
|-------------|--------|-------------|
| DMA_STS     | 0x00   | 0x0000 0000 |
| DMA_CLR     | 0x04   | 0x0000 0000 |
| DMA_C1CTRL  | 0x08   | 0x0000 0000 |
| DMA_C1DTCNT | 0x0C   | 0x0000 0000 |
| DMA_C1PADDR | 0x10   | 0x0000 0000 |

| Register     | Offset | Reset value |
|--------------|--------|-------------|
| DMA_C1MADDR  | 0x14   | 0x0000 0000 |
| DMA_C2CTRL   | 0x1C   | 0x0000 0000 |
| DMA_C2DTCNT  | 0x20   | 0x0000 0000 |
| DMA_C2PADDR  | 0x24   | 0x0000 0000 |
| DMA_C2MADDR  | 0x28   | 0x0000 0000 |
| DMA_C3CTRL   | 0x30   | 0x0000 0000 |
| DMA_C3DTCNT  | 0x34   | 0x0000 0000 |
| DMA_C3PADDR  | 0x38   | 0x0000 0000 |
| DMA_C3MADDR  | 0x3C   | 0x0000 0000 |
| DMA_C4CTRL   | 0x44   | 0x0000 0000 |
| DMA_C4DTCNT  | 0x48   | 0x0000 0000 |
| DMA_C4PADDR  | 0x4C   | 0x0000 0000 |
| DMA_C4MADDR  | 0x50   | 0x0000 0000 |
| DMA_C5CTRL   | 0x58   | 0x0000 0000 |
| DMA_C5DTCNT  | 0x5C   | 0x0000 0000 |
| DMA_C5PADDR  | 0x60   | 0x0000 0000 |
| DMA_C5MADDR  | 0x64   | 0x0000 0000 |
| DMA_C6CTRL   | 0x6C   | 0x0000 0000 |
| DMA_C6DTCNT  | 0x70   | 0x0000 0000 |
| DMA_C6PADDR  | 0x74   | 0x0000 0000 |
| DMA_C6MADDR  | 0x78   | 0x0000 0000 |
| DMA_C7CTRL   | 0x80   | 0x0000 0000 |
| DMA_C7DTCNT  | 0x84   | 0x0000 0000 |
| DMA_C7PADDR  | 0x88   | 0x0000 0000 |
| DMA_C7MADDR  | 0x8C   | 0x0000 0000 |
| DMA_SRC_SEL0 | 0xA0   | 0x0000 0000 |
| DMA_SRC_SEL1 | 0xA4   | 0x0000 0000 |

### DMA interrupt status register (DMA\_STS)

Access: 0 wait state, accessible by bytes, half-words or words.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:28 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 27    | DTERRF7  | 0x0         | ro   | Channel 7 data transfer error event flag<br>0: No transfer error occurred.<br>1: Transfer error occurred.  |
| Bit 26    | HDTF7    | 0x0         | ro   | Channel 7 half transfer event flag<br>0: No half-transfer event occurred.<br>1: Half-transfer event occurred.  |
| Bit 25    | FDTF7    | 0x0         | ro   | Channel 7 transfer complete event flag<br>0: No transfer complete event occurred.<br>1: Transfer complete event occurred.  |
| Bit 24    | GF7      | 0x0         | ro   | Channel 7 global event flag<br>0: No transfer error, half transfer or transfer complete event occurred.<br>1: Transfer error, half transfer or transfer complete event occurred. |

| Bit    | Name    | Reset value | Type | Description  |
|--------|---------|-------------|------|--|
| Bit 23 | DTERRF6 | 0x0         | ro   | Channel 6 data transfer error event flag<br>0: No transfer error occurred.<br>1: Transfer error occurred.  |
| Bit 22 | HDTF6   | 0x0         | ro   | Channel 6 half transfer event flag<br>0: No half-transfer event occurred.<br>1: Half-transfer event occurred.  |
| Bit 21 | FDTF6   | 0x0         | ro   | Channel 6 transfer complete event flag<br>0: No transfer complete event occurred.<br>1: Transfer complete event occurred.  |
| Bit 20 | GF6     | 0x0         | ro   | Channel 6 global event flag<br>0: No transfer error, half transfer or transfer complete event occurred.<br>1: Transfer error, half transfer or transfer complete event |
| Bit 19 | DTERRF5 | 0x0         | ro   | Channel 5 data transfer error event flag<br>0: No transfer error occurred.<br>1: Transfer error occurred.  |
| Bit 18 | HDTF5   | 0x0         | ro   | Channel 5 half transfer event flag<br>0: No half-transfer event occurred.<br>1: Half-transfer event occurred.  |
| Bit 17 | FDTF5   | 0x0         | ro   | Channel 5 transfer complete event flag<br>0: No transfer complete event occurred.<br>1: Transfer complete event occurred.  |
| Bit 16 | GF5     | 0x0         | ro   | Channel 5 global event flag<br>0: No transfer error, half transfer or transfer complete event occurred.<br>1: Transfer error, half transfer or transfer complete event |
| Bit 15 | DTERRF4 | 0x0         | ro   | Channel 4 data transfer error event flag<br>0: No transfer error occurred.<br>1: Transfer error occurred.  |
| Bit 14 | HDTF4   | 0x0         | ro   | Channel 4 half transfer event flag<br>0: No half-transfer event occurred.<br>1: Half-transfer event occurred.  |
| Bit 13 | FDTF4   | 0x0         | ro   | Channel 4 transfer complete event flag<br>0: No transfer complete event occurred.<br>1: Transfer complete event occurred.  |
| Bit 12 | GF4     | 0x0         | ro   | Channel 4 global event flag<br>0: No transfer error, half transfer or transfer complete event occurred.<br>1: Transfer error, half transfer or transfer complete event |
| Bit 11 | DTERRF3 | 0x0         | ro   | Channel 3 data transfer error event flag<br>0: No transfer error occurred.<br>1: Transfer error occurred.  |
| Bit 10 | HDTF3   | 0x0         | ro   | Channel 3 half transfer event flag<br>0: No half-transfer event occurred.<br>1: Half-transfer event occurred.  |
| Bit 9  | FDTF3   | 0x0         | ro   | Channel 3 transfer complete event flag<br>0: No transfer complete event occurred.<br>1: Transfer complete event occurred.  |
| Bit 8  | GF3     | 0x0         | ro   | Channel 3 global event flag<br>0: No transfer error, half transfer or transfer complete event occurred.<br>1: Transfer error, half transfer or transfer complete event |

| Bit   | Name    | Reset value | Type | Description  |
|-------|---------|-------------|------|--|
| Bit 7 | DTERRF2 | 0x0         | ro   | Channel 2 data transfer error event flag<br>0: No transfer error occurred.<br>1: Transfer error occurred.  |
| Bit 6 | HDTF2   | 0x0         | ro   | Channel 2 half transfer event flag<br>0: No half-transfer event occurred.<br>1: Half-transfer event occurred.  |
| Bit 5 | FDTF2   | 0x0         | ro   | Channel 2 transfer complete event flag<br>0: No transfer complete event occurred.<br>1: Transfer complete event occurred.  |
| Bit 4 | GF2     | 0x0         | ro   | Channel 2 global event flag<br>0: No transfer error, half transfer or transfer complete event occurred.<br>1: Transfer error, half transfer or transfer complete event |
| Bit 3 | DTERRF1 | 0x0         | ro   | Channel 1 data transfer error event flag<br>0: No transfer error occurred.<br>1: Transfer error occurred.  |
| Bit 2 | HDTF1   | 0x0         | ro   | Channel 1 half transfer event flag<br>0: No half-transfer event occurred.<br>1: Half-transfer event occurred.  |
| Bit 1 | FDTF1   | 0x0         | ro   | Channel 1 transfer complete event flag<br>0: No transfer complete event occurred.<br>1: Transfer complete event occurred.  |
| Bit 0 | GF1     | 0x0         | ro   | Channel 1 global event flag<br>0: No transfer error, half transfer or transfer complete event occurred.<br>1: Transfer error, half transfer or transfer complete event |

### DMA interrupt flag clear register (DMA\_CLR)

Access: 0 wait state, accessible by bytes, half-words or words.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:28 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 27    | DTERRFC7 | 0x0         | rw1c | Channel 7 data transfer error flag clear<br>0: No effect<br>1: Clear the DTERRF flag in the DMA_STS register                     |
| Bit 26    | HDTFC7   | 0x0         | rw1c | Channel 7 half transfer flag clear<br>0: No effect<br>1: Clear the HDTF7 flag in the DMA_STS register                            |
| Bit 25    | FDTFC7   | 0x0         | rw1c | Channel 7 transfer complete flag clear<br>0: No effect<br>1: Clear the FDTF7 flag in the DMA_STS register                        |
| Bit 24    | GFC7     | 0x0         | rw1c | Channel 7 global interrupt flag clear<br>0: No effect<br>1: Clear the DTERRF7, HDTF7, FDTF7 and GF7 flag in the DMA_STS register |
| Bit 23    | DTERRFC6 | 0x0         | rw1c | Channel 6 data transfer error flag clear<br>0: No effect<br>1: Clear the DTERRF6 flag in the DMA_STS register                    |
| Bit 22    | HDTFC6   | 0x0         | rw1c | Channel 6 half transfer flag clear<br>0: No effect<br>1: Clear the HDTF6 flag in the DMA_STS register                            |



| Bit    | Name     | Reset value | Type | Description  |
|--------|----------|-------------|------|--|
| Bit 21 | FDTFC6   | 0x0         | rw1c | Channel 6 transfer complete flag clear<br>0: No effect<br>1: Clear the FDTF6 flag in the DMA_STS register                        |
| Bit 20 | GFC6     | 0x0         | rw1c | Channel 6 global interrupt flag clear<br>0: No effect<br>1: Clear the DTERRF6, HDTF6, FDTF6 and GF6 flag in the DMA_STS register |
| Bit 19 | DTERRFC5 | 0x0         | rw1c | Channel 5 data transfer error flag clear<br>0: No effect<br>1: Clear the DTERRF5 flag in the DMA_STS register                    |
| Bit 18 | HDTFC5   | 0x0         | rw1c | Channel 5 half transfer flag clear<br>0: No effect<br>1: Clear the HDTF5 flag in the DMA_STS register                            |
| Bit 17 | FDTFC5   | 0x0         | rw1c | Channel 5 transfer complete flag clear<br>0: No effect<br>1: Clear the FDTF5 flag in the DMA_STS register                        |
| Bit 16 | GFC5     | 0x0         | rw1c | Channel 5 global interrupt flag clear<br>0: No effect<br>1: Clear the DTERRF5, HDTF5, FDTF5 and GF5 in the DMA_STS register      |
| Bit 15 | DTERRFC4 | 0x0         | rw1c | Channel 4 data transfer error flag clear<br>0: No effect<br>1: Clear the DTERRF4 flag in the DMA_STS register                    |
| Bit 14 | HDTFC4   | 0x0         | rw1c | Channel 4 half transfer flag clear<br>0: No effect<br>1: Clear the HDTF4 flag in the DMA_STS register                            |
| Bit 13 | FDTFC4   | 0x0         | rw1c | Channel 4 transfer complete flag clear<br>0: No effect<br>1: Clear the FDTF4 flag in the DMA_STS register                        |
| Bit 12 | GFC4     | 0x0         | rw1c | Channel 4 global interrupt flag clear<br>0: No effect<br>1: Clear the DTERRF4, HDTF4, FDTF4 and GF4 flag in the DMA_STS register |
| Bit 11 | DTERRFC3 | 0x0         | rw1c | Channel 3 data transfer error flag clear<br>0: No effect<br>1: Clear the DTERRF3 flag in the DMA_STS register                    |
| Bit 10 | HDTFC3   | 0x0         | rw1c | Channel 3 half transfer flag clear<br>0: No effect<br>1: Clear the HDTF3 flag in the DMA_STS register                            |
| Bit 9  | FDTFC3   | 0x0         | rw1c | Channel 3 transfer complete flag clear<br>0: No effect<br>1: Clear the FDTF3 flag in the DMA_STS register                        |
| Bit 8  | GFC3     | 0x0         | rw1c | Channel 3 global interrupt flag clear<br>0: No effect<br>1: Clear the DTERRF3, HDTF3, FDTF3 and GF3 flag in the DMA_STS register |
| Bit 7  | DTERRFC2 | 0x0         | rw1c | Channel 2 data transfer error flag clear<br>0: No effect<br>1: Clear the DTERRF2 flag in the DMA_STS register                    |
| Bit 6  | HDTFC2   | 0x0         | rw1c | Channel 2 half transfer flag clear<br>0: No effect<br>1: Clear the HDTF2 flag in the DMA_STS register                            |

| Bit   | Name     | Reset value | Type | Description   |
|-------|----------|-------------|------|---|
| Bit 5 | FDTFC2   | 0x0         | rw1c | Channel 2 transfer complete flag clear<br>0: No effect<br>1: Clear the FDTF2 flag in the DMA_STS register                   |
| Bit 4 | GFC2     | 0x0         | rw1c | Channel 2 global interrupt flag clear<br>0: No effect<br>1: Clear the DTERRF2, HDTF2, FDTF2 and GF2 in the DMA_STS register |
| Bit 3 | DTERRFC1 | 0x0         | rw1c | Channel 1 data transfer error flag clear<br>0: No effect<br>1: Clear the DTERRF1 flag in the DMA_STS register               |
| Bit 2 | HDTFC1   | 0x0         | rw1c | Channel 1 half transfer flag clear<br>0: No effect<br>1: Clear the HDTF1 flag in the DMA_STS register                       |
| Bit 1 | FDTFC1   | 0x0         | rw1c | Channel 1 transfer complete flag clear<br>0: No effect<br>1: Clear the FDTF1 flag in the DMA_STS register                   |
| Bit 0 | GFC1     | 0x0         | rw1c | Channel 1 global interrupt flag clear<br>0: No effect<br>1: Clear the DTERRF1, HDTF1, FDTF1 and GF1 in the DMA_STS register |

### DMA channel-x configuration register (DMA\_CxCTRL) (x = 1...7)

Access: 0 wait state, accessible by bytes, half-words or words.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:15 | Reserved | 0x00000     | resd | Kept at its default value.  |
| Bit 14    | M2M      | 0x0         | rw   | Memory to memory mode<br>0: Disabled<br>1: Enabled                                    |
| Bit 13:12 | CHPL     | 0x0         | rw   | Channel priority level<br>00: Low<br>01: Medium<br>10: High<br>11: Very high          |
| Bit 11:10 | MWIDTH   | 0x0         | rw   | Memory data bit width<br>00: 8 bits<br>01: 16 bits<br>10: 32 bits<br>11: Reserved     |
| Bit 9: 8  | PWIDTH   | 0x0         | rw   | Peripheral data bit width<br>00: 8 bits<br>01: 16 bits<br>10: 32 bits<br>11: Reserved |
| Bit 7     | MINCM    | 0x0         | rw   | Memory address increment mode<br>0: Disabled<br>1: Enabled.                           |
| Bit 6     | PINCM    | 0x0         | rw   | Peripheral address increment mode<br>0: Disabled<br>1: Enabled.                       |

| Bit   | Name     | Reset value | Type | Description  |
|-------|----------|-------------|------|--|
| Bit 5 | LM       | 0x0         | rw   | Circular mode<br>0: Disabled<br>1: Enabled.                                |
| Bit 4 | DTD      | 0x0         | rw   | Data transfer direction<br>0: Read from peripherals<br>1: Read from memory |
| Bit 3 | DTERRIEN | 0x0         | rw   | Data transfer error interrupt enable<br>0: Disabled<br>1: Enabled.         |
| Bit 2 | HDTIEN   | 0x0         | rw   | Half-transfer interrupt enable<br>0: Disabled<br>1: Enabled.               |
| Bit 1 | FDTIEN   | 0x0         | rw   | Transfer complete interrupt enable<br>0: Disabled<br>1: Enabled.           |
| Bit 0 | CHEN     | 0x0         | rw   | Channel enable<br>0: Disabled<br>1: Enabled.                               |

### DMA channel-x number of data register (DMA\_CxDTCNT) (x = 1...7)

Access: 0 wait state, accessible by bytes, half-words or words.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 15:0  | CNT      | 0x0000      | rw   | Number of data to transfer<br>The number of data to transfer is from 0x0 to 0xFFFF. This register can only be written when the CHEN bit in the corresponding channel is set 0. The value is decremented after each DMA transfer.<br>Note: This register holds the number of data to transfer, instead of transfer size. The transfer size is calculated by data width. |

### DMA channel-x peripheral address register (DMA\_CxPADDR) (x = 1...7)

Access: 0 wait state, accessible by bytes, half-words or words.

| Bit      | Name  | Reset value | Type | Description   |
|----------|-------|-------------|------|---|
| Bit 31:0 | PADDR | 0x0000 0000 | rw   | Peripheral base address<br>Base address of peripheral data register is the source or destination of data transfer.<br>Note: The register can only be written when the CHEN bit in the corresponding channel is set 0. |

## DMA channel-x memory address register (DMA\_CxMADDR) (x = 1...7)

Access: 0 wait state, accessible by bytes, half-words or words.

| Bit      | Name  | Reset value | Type | Description   |
|----------|-------|-------------|------|---|
| Bit 31:0 | MADDR | 0x0000 0000 | rw   | Memory base address<br>Memory address is the source or destination of data transfer.<br>Note: The register can only be written when the CHEN bit in the corresponding channel is set 0. |

## DMA channel source register 0 (DMA\_SRC\_SEL0)

Access: 0 wait state, accessible by bytes, half-words or words.

| Bit       | Name    | Reset value | Type | Description   |
|-----------|---------|-------------|------|---|
| Bit 31:24 | CH4_SRC | 0x00        | rw   | CH4 source select<br>When DMA_FLEX_EN = 1, channel 4 is selected by the CH4_SRC. Refer to Section 9.3.7 for more information. |
| Bit 23:16 | CH3_SRC | 0x00        | rw   | CH3 source select<br>When DMA_FLEX_EN = 1, channel 3 is selected by the CH3_SRC. Refer to Section 9.3.7 for more information. |
| Bit 15:8  | CH2_SRC | 0x00        | rw   | CH2 source select<br>When DMA_FLEX_EN = 1, channel 2 is selected by the CH2_SRC. Refer to Section 9.3.7 for more information. |
| Bit 7:0   | CH1_SRC | 0x00        | rw   | CH1 source select<br>When DMA_FLEX_EN = 1, channel 1 is selected by the CH1_SRC. Refer to Section 9.3.7 for more information. |

## DMA channel source register 1 (DMA\_SRC\_SEL1)

Access: 0 wait state, accessible by bytes, half-words or words.

| Bit       | Name        | Reset value | Type | Description   |
|-----------|-------------|-------------|------|---|
| Bit 31:25 | Reserved    | 0x00        | resd | Kept at its default value.  |
| Bit 24    | DMA_FLEX_EN | 0x0         | rw   | DMA flexible mapping mode selection<br>0: Fixed mapping mode<br>1: Flexible mapping mode                                      |
| Bit 23:16 | CH7_SRC     | 0x00        | rw   | CH7 source select<br>When DMA_FLEX_EN = 1, channel 7 is selected by the CH7_SRC. Refer to Section 9.3.7 for more information. |
| Bit 15:8  | CH6_SRC     | 0x00        | rw   | CH6 source select<br>When DMA_FLEX_EN = 1, channel 6 is selected by the CH6_SRC. Refer to Section 9.3.7 for more information. |
| Bit 7:0   | CH5_SRC     | 0x00        | rw   | CH5 source select<br>When DMA_FLEX_EN = 1, channel 5 is selected by the CH5_SRC. Refer to Section 9.3.7 for more information. |

# 10 CRC calculation unit (CRC)

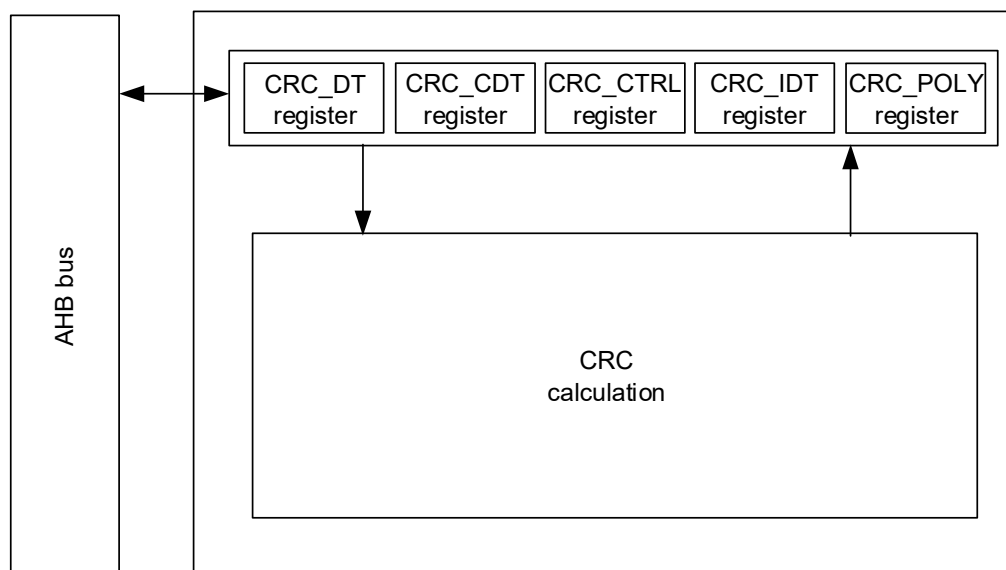
## CRC introduction

The Cyclic Redundancy Check (CRC) is an independent peripheral with CRC check feature. It follows CRC32/MPEG-2 standard.

The CRC\_CTRL register is used to select output data reverse (word, REVOD = 1) or input data reverse (byte, REVID = 01; half-word, REVID = 10; word: REVID = 11). CRC calculation unit is also equipped with initialization function. After each CRC reset, the value in the CRC\_IDT register is written into the data register (CRC\_DT) by CRC.

The CRC\_POLY register is used to set different polynomial coefficient. The polynomial size can be set as 7 bits, 8 bits, 16 bits or 32 bits through the POLY-SIZE bit in the CRC\_CTR register.

Users can write the data to go through CRC check and read the calculated result through CRC\_DT register. Note that the calculation result is the combination of the previous result and the current value to be calculated.



**Figure 26. CRC calculation unit block diagram**

### Main features

- Use CRC-32 code
- Support the generation of polynomial
- 4 HCLK cycles for each CRC calculation
- Support input/output data format toggle
- Perform write/read operation through CRC\_DT register
- Set an initialization value with the CRC\_IDT register. The value is loaded with CRC\_DT register after each CRC reset.

## CRC functional description

According to CRC calculation principle: the input data is taken as dividend, and the generator polynomial as a division. Using mod 2 division logic, the input data divided by the generator polynomial gets a remainder, that is, the CRC value.

### CRC calculation procedure

- Input data reverse. After data input, reverse input data depending on the REVID value in CRC\_CTRL register
- Initialization. The first data input needs to be XOR-ed with the initial value defined in the CRC\_IDT register. If it is not the first data input, the initial value is the previously calculated result.
- CRC calculation. Dividing the input data by the generator polynomial (0x4C11DB7) using mod 2 division method produces a remainder, that is, CRC value.
- Output data toggle. Select whether to perform word toggle before output CRC value through the REVOD bit in the CRC\_CTRL register
- XOR calculation. The XOR-ed result is fixed at 0x0000 0000

### CRC-32/MPEG-2 parameters

- Generator polynomial: 0x4C11DB7  
that is,  $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$
- Initial value: 0xFFFF FFFF, in order to avoid that 1-byte 0x00 data to be calculated has the same result as that of multiple-byte 0x00.
- XOR-ed value: 0x0000 0000, indicating that the CRC result will not be XOR-ed.

### Toggle function

- Byte reverse, 8 bits in a group, and sequence is reversed within a group. As shown in figure below, if the original data is 0x12345678, it is reversed as 0x482C6A1E.
- Half-word reverse, 16 bits in a group, and sequence is reversed within a group.
- Word reverse, 32 bits in a group, and sequence is reversed within a group.

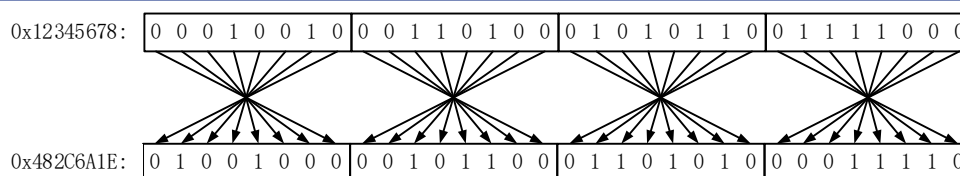


Figure 27. Diagram of byte reverse

## CRC registers

CRC\_DT register can be accessed by bytes (8 bits), half-words (16 bits) or words (32 bits). Other registers have to be accessed by words (32 bits).

**Table 29. CRC register map and reset value**

| Register | Offset | Reset value |
|----------|--------|-------------|
| CRC_DT   | 0x00   | 0xFFFF FFFF |
| CRC_CDT  | 0x04   | 0x0000 0000 |
| CRC_CTRL | 0x08   | 0x0000 0000 |
| CRC_IDT  | 0x10   | 0xFFFF FFFF |
| CRC_POLY | 0x14   | 0x04C1 1DB7 |

### Data register (CRC\_DT)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 31:0 | DT   | 0xFFFF FFFF | rw   | Data value<br>Used as input register when writing new data into the CRC calculator. It returns CRC calculation results when it is read. |

### Common data register (CRC\_CDT)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:8 | Reserved | 0x000000    | resd | Kept at its default value.  |
| Bit 7:0  | CDT      | 0x00        | rw   | Common 8-bit data value<br>This field is used to store one byte data temporarily. This register is not affected by the CRC reset generated by the RST bit in the CRC_CTRL register. |

### Control register (CRC\_CTRL)

| Bit      | Name      | Reset value | Type | Description   |
|----------|-----------|-------------|------|---|
| Bit 31:8 | Reserved  | 0x000000    | resd | Kept at its default value.  |
| Bit 7    | REVOD     | 0           | resd | Reverse output data<br>Set and cleared by software. This bit is used to control whether or not to reverse output data.<br>0: No effect<br>1: Word reverse                                   |
| Bit 6:5  | REVID     | 0x0         | rw   | Reverse input data<br>Set and cleared by software. This bit is used to control how to reverse input data.<br>00: No effect<br>01: Byte reverse<br>10: Half-word reverse<br>11: Word reverse |
| Bit 4:3  | POLY_SIZE | 0x0         | rw   | Polynomial size<br>This field is used to set the size of polynomial. It is used in conjunction with the CRC_POLY register.<br>00: 32 bits<br>01: 16 bits<br>10: 8 bits<br>11: 7 bits        |
| Bit 2:1  | Reserved  | 0x0         | resd | Kept at its default value.  |

| Bit   | Name | Reset value | Type | Description   |
|-------|------|-------------|------|---|
| Bit 0 | RST  | 0           | rw   | Reset CRC calculation unit<br>Set by software. Cleared by hardware. To reset CRC calculation unit, the data register is set as 0xFFFF FFFF.<br>0: No effect<br>1: Reset |

### Initialization register (CRC\_IDT)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 31:0 | IDT  | 0xFFFF FFFF | rw   | Initialization data register<br>When CRC reset is triggered by the RST bit in the CRC_CTRL register, the value in the initialization register is written into the CRC_DT register as an initial value. |

### Polynomial register (CRC\_POLY)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 31:0 | POLY | 0x04C1 1DB7 | rw   | Polynomial coefficient<br>The generated polynomial is a divisor in CRC calculation.<br>Using CRC32 mode, this polynomial coefficient is 0x4C11DB7.<br>Users can also set the polynomial coefficient according to their needs. |



# 11 I<sup>2</sup>C interface

## I<sup>2</sup>C introduction

I<sup>2</sup>C (inter-integrated circuit) bus interface manages the communication between the microcontroller and serial I<sup>2</sup>C bus. It supports master and slave modes, with up to 1 Mbit/s of communication speed (enhanced edition).

## I<sup>2</sup>C main features

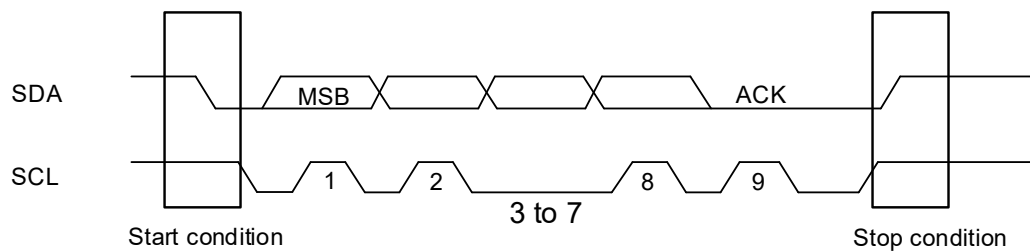
- I<sup>2</sup>C bus
  - Master and slave modes
  - Multimaster capability
  - Stand speed (100 kHz), fast speed (400 kHz) and enhanced fast speed (1 MHz)
  - 7-bit and 10-bit address modes
  - Two 7-bit slave addresses (two addresses, one of them can be masked)
  - Broadcast call mode
  - Programmable data setup and hold time
  - Clock stretching capability
- Support DMA transfer
- Programmable digital noise filter
- Support SMBus2.0 protocol
  - PEC generation and verification
  - Acknowledgement control for command and data
  - ARP(address resolution protocol)
  - Master capability
  - Device capability
  - SMBus reminder capability
  - Timeout detection
  - Idle detection
- PMBus

## I<sup>2</sup>C function overview

I<sup>2</sup>C bus consists of a data line (SDA) and clock line (SCL). It can achieve a maximum of 100 kHz speed in standard mode, up to 400 kHz in fast mode, up to 1 MHz in enhanced fast mode. A frame of data transmission begins with a Start condition and ends with a Stop condition. The bus is kept in busy state after receiving the Start condition, and becomes idle as long as it receives the Stop condition.

Start condition: SDA switches from high to low when SCL is set high.

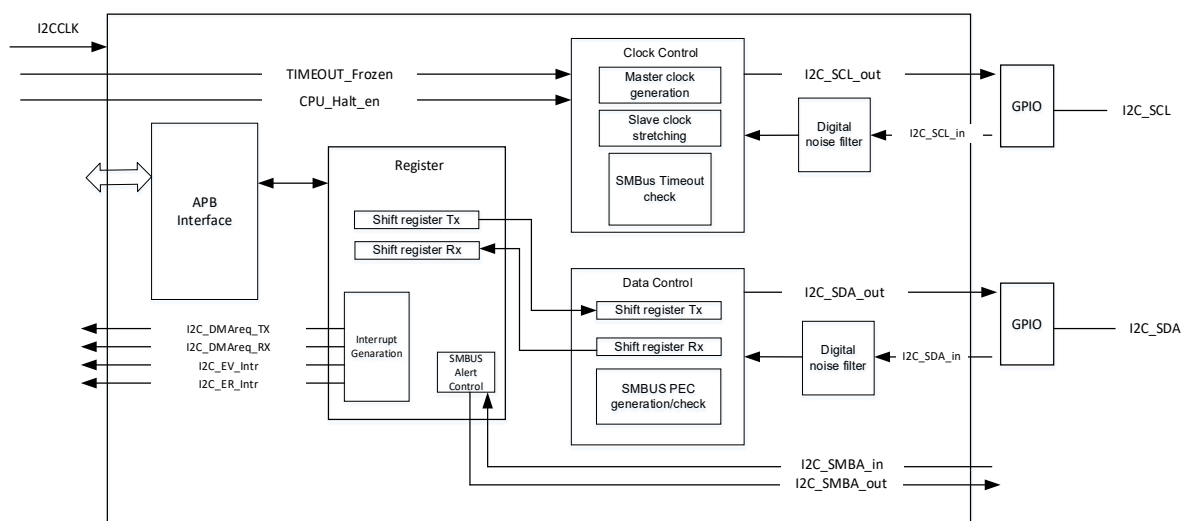
Stop condition: SDA switches from low to high when SCL is set high.



### Figure 28. I<sup>2</sup>C bus protocol

## I<sup>2</sup>C interface

Figure 29 shows the block diagram of I<sup>2</sup>C function.



### Figure 29. I<sup>2</sup>C function block diagram

## Operating mode

I<sup>2</sup>C bus interface can operate both in master mode and slave mode. Switching from master mode to slave mode, vice versa, is supported as well. By default, the interface operates in slave mode. When GENSTART = 1 is set (Start condition is activated), the I<sup>2</sup>C bus interface switches from slave mode to master mode, and returns to slave mode automatically at the end of data transfer (Stop condition is triggered).

## Communication process

■ Master mode communication:

- Start condition generation
- Address transmission
- Data Tx or Rx
- Stop condition generation
- End of communication

■ Slave mode communication:

- Wait until the address is matched.
- Data Tx or Rx
- Wait for the generation of Stop condition
- End of communication

**Digital filter capability**

The digital filter is available on both SCL and SDA lines. It is enabled by setting the DFLT[3:0] bit (0 ~ 15) in the I2C\_CTRL1 register to reduce noise on bus on a large scale. The filter time is  $DLFT \times t_{I2C\_CLK}$ .

The digital filter is not allowed to be altered when the I<sup>2</sup>C is enabled.

**Address control**

Both master and slave support 7-bit and 10-bit addressing modes.

**Slave address mode:**

- In 7-bit mode (ADDR1MODE = 0)
  - ADDR1EN = 1, ADDR2EN = 0 stands for a single address mode: only matches OADDR1
  - ADDR1EN = 1, ADDR2EN = 1 stands for dual address mode: matches OADDR1 and OADDR2
- In 10-bit mode (ADDR1MODE = 1)
  - Only supports a single address mode (ADDR1EN = 1, ADDR2EN = 0), matches OADDR1

**Slave address masking capability**

The Slave address 2 (OADDR2) is maskable, which is done by setting the ADDR2MASK[2:0].

- 0: Address bit [7:1]
- 1: Address bit [7:2]
- 2: Address bit [7:3]
- 3: Address bit [7:4]
- 4: Address bit [7:5]
- 5: Address bit [7:6]
- 6: Address bit [7]
- 7: All addresses, excluding those reserved by I<sup>2</sup>C

**Support special slave address:**

- Broadcast call address (0b0000000x): This address is enabled when GCAEN = 1.
- SMBus device default address (0b1100001x): This address is enabled for SMBus address resolution protocol in SMBus device mode (DEVADDREN = 1).
- SMBus master default address (0b0001000x): This address is enabled for SMBus master notification protocol in SMBus master mode (DEVADDREN = 1).
- SMBus alert address (0b0001100x): This address is enabled for SMBus alert response address protocol in SMBus master mode when SMBALERT = 1.

Refer to SMBus2.0 protocol for more information.

#### Slave address matching procedure:

- Receive a Start condition
- Address matching
- The slave sends an ACK if address is matched.
- ADDR7F is set, with SDR indicating the transmission direction
  - When SDR = 0, slave enters receiver mode, starting receiving data.
  - When SDR = 1, slave enters transmitter mode, starting transmitting data

#### Clock stretching capability

Clock stretching is enabled by default (STRETCH = 0 in the I2C\_CTRL1 register). The slave can hold the SCL line low for software operation. If the clock stretching capability is not supported by master, then the STRETCH must be set in the I2C\_CTRL register. It should be noted that the clock stretching capability of I<sup>2</sup>C slave must be configured before the I<sup>2</sup>C peripherals are enabled.

#### Clock stretching capability enabled

I<sup>2</sup>C slave stretches the SCL clock in one of the following conditions:

- Address reception: When the address received by slave matches the local address enabled (ADDRF = 1 in the I2C\_STS), the SCL line is pulled down until the ADDR is cleared by setting the ADDR in the I2C\_CLR.
- Data reception: When the shift register has received another new byte before the data in the I2C\_RXDT register is read, the I2C will hold the SCL bus low to wait for the software to read I2C\_RXDT register.
- Data transmission: If no data is written when the ADDR is cleared, TDBE = 1 in the I2C\_STS, then the SCL line will be pulled down until the data is written to the I2C\_TXDT.
- Data transmission: If no data is written to the I2C\_TXDT after the completion of the previous data transfer, the SCL line will be pulled down until data is written to the I2C\_TXDT.
- When slave data control mode is selected (SCTRL = 1 in the I2C\_CTRL1) and RLDE = 1 in the I2C\_CTRL2 register, if TCRLD = 1, indicating the completion of the last data transfer, then the TCRLD will be cleared by hardware so as to release the SCL line before a non-zero value is written to the CNT bit in the I2C\_CTRL2 register.

#### Clock stretching capability disabled

The SCL clock is disabled when STRETCH = 1 in the I2C\_CTRL1 register, with the following conditions worth our notice:

- Address reception: The SCL clock is not stretched when the address received by slave matches the local address enabled (ADDRF = 1 in the I2C\_STS).
- Data reception: If there is data to be read in the I2C\_RXDT register before the next ACK signal, an overflow will occur, and the OUF bit will also be set in the I2C\_STS register.
- Data transmission: If no data is written to the I2C\_TXDT register after the completion of the previous data transfer, an underflow will occur, and the OUF will also be set in the I2C\_STS register.

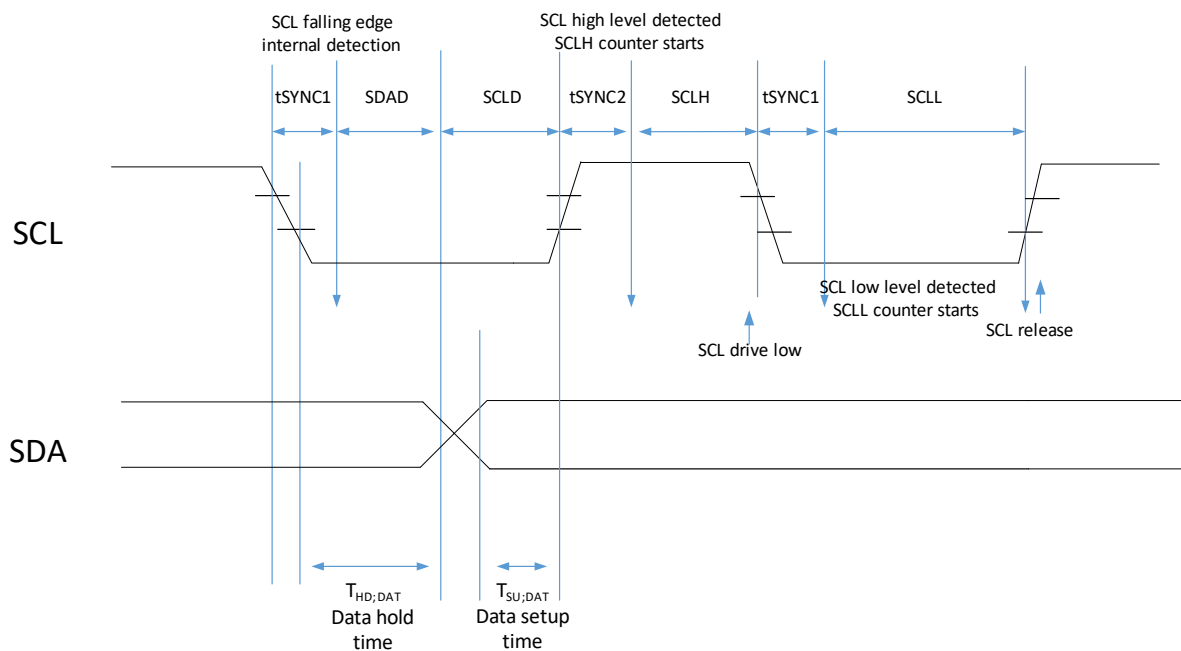
#### I<sup>2</sup>C timing control

I<sup>2</sup>C core is clocked by I2C\_CLK whereas the I2C\_CLK is clocked by PCLK1. The PCLK1 should be set to be less than 4/3 SCL cycles.

The corresponding bits in the I2C\_CLKCTR register are used for timing configuration.

- DIV[7:0]: I<sup>2</sup>C clock divider
- SDAD[3:0]: Data hold time ( $t_{HD;DAT}$ )
- SCLD[3:0]: Data setup time ( $t_{SU;DAT}$ )
- SCLH[7:0]: SCL high
- SCLL[7:0]: SCL low

Note: Timing configuration cannot be modified once the I<sup>2</sup>C is enabled.



**Figure 30. Setup and hold time**

It is possible to configure data hold time ( $t_{HD;DAT}$ ) and data setup time ( $t_{SU;DAT}$ ) freely by setting the DIV[7:0], SDAD[3:0] and SCLD[3:0] in the I2C\_CLKCTRL register.

- Data hold time ( $t_{HD;DAT}$ ): refers to the duration from SCL falling edge to SDA output

$$t_{HD;DAT} = t_{SDAD} + t_{SYNC}$$

$$t_{SDAD} = SDAD \times (DIV + 1) \times t_{I2C\_CLK}$$

$$t_{SYNC} = (DLFT + 3) \times t_{I2C\_CLK} - t_f$$

$t_{SYNC}$  consists of three parts:

- SCL falling edge time  $t_f$
- Digital filter input latency ( $DLFT \times t_{I2C\_CLK}$ )
- Synchronization delay between SCL and I2C\_CLK (2 ~ 3 I2C\_CLK cycles)

- Data setup time ( $t_{SU;DAT}$ ): refers to the duration from SDA output to SCL rising edge

$$t_{SU;DAT} = SCLD \times (DIV + 1) \times t_{I2C\_CLK} - t_f$$

In master mode, the width of SCL signals (high and low) can be configured freely by setting the DIV[7:0], SCLH[7:0] and SCLL[7:0] in the I2C\_CLKCTRL register.

**SCL low:** When the SCL low signal is detected, the internal SCLL counter starts counting until it reaches the SCLL value. At this point, the SCL line is released and become high.

**SCL high:** When the SCL high signal is detected, the internal SCLH counter starts counting. When the counter value reaches the SCLH value, the SCL line is pulled low. In the process of SCL remaining high, if it is pulled low by external bus, the internal SCLH counter will stop counting and start counting in SCL low mode, laying the foundation for clock synchronization.

■ SCL high signal width

$$t_{\text{HIGH}} = (\text{SCLH} + 1) \times (\text{DIV} + 1) \times t_{\text{I2C\_CLK}}$$

■ SCL low signal width

$$T_{\text{Low}} = (\text{SCLL} + 1) \times (\text{DIV} + 1) \times t_{\text{I2C\_CLK}}$$

**Table 30. I<sup>2</sup>C timing specifications**

| Parameter<br>Min.        |                      | Standard mode |      | Fast mode |      | Fast mode plus |      | SMBus |      |
|--------------------------|----------------------|---------------|------|-----------|------|----------------|------|-------|------|
|                          |                      | Max.          | Min. | Max.      | Min. | Max.           | Min. | Max.  |      |
| f <sub>SCL</sub> (kHz)   | SCL clock frequency  |               | 100  |           | 400  |                | 1000 |       | 100  |
| t <sub>LOW</sub> (μs)    | SCL clock low        | 4.7           |      | 1.3       |      | 0.5            |      | 4.7   |      |
| t <sub>HIGH</sub> (μs)   | SCL clock high       | 4.0           |      | 0.6       |      | 0.26           |      | 4.0   | 50   |
| t <sub>HD,DAT</sub> (μs) | Data hold time       | 0             |      | 0         | 0.9  | 0              | 0.45 | 300   |      |
| t <sub>SU,DAT</sub> (ns) | Data setup time      | 250           |      | 100       |      | 50             |      | 250   |      |
| t <sub>r</sub> (ns)      | SCL/SDA rising edge  |               | 1000 |           | 300  |                | 120  |       | 1000 |
| t <sub>f</sub> (ns)      | SCL/SDA falling edge |               | 300  |           | 300  |                | 120  |       | 300  |

## Data transfer management

Data transfer counter is available in the I<sup>2</sup>C interface to control communication flow. It is mainly used for:

- NACK transmission: master reception mode
- STOP transmission: master reception/transmission modes
- RESTART generation: master reception/transmission modes
- ACK control: slave mode (SMBus)
- PEC transmission/reception: master/slave modes

Generally, the data transfer management counter (by setting the CNT[7:0] in the I2C\_CTRL2) is applicable to master mode. It is disabled in slave mode. This counter is used only in SMBus mode for the ACK control and PEC reception of each byte by the slave. In SMBus mode, the slave enables data counter with the SCTRL bit in the I2C\_CTRL2 register.

### Byte control through master

The CNT[7:0] bit in the I2C\_CTRL2 register is used to configure the number of bytes to be transferred, ranging from 1 to 255. If the number of data to be transferred is greater than 255, then the RLDEN bit has to be set in the I2C\_CTRL2 register to enable reload mode. The following configuration processes are described in two aspects:

- ≤255 bytes, for example, the number of data to be transferred is 100 bytes
  - Step 1: Disable reload mode by setting RLDEN = 0
  - Step 2: Set CNT[7:0] = 100
- >255 bytes, for example, the number of data to be transferred is 600 bytes
  - Step 1: Enable reload mode by setting RLDEN = 1
  - Step 2: Set CNT[7:0] = 255, the remaining bytes are 600-255 = 345 bytes
  - Step 3: After the completion of 255-byte data transfer, the TCRLD is set in the I2C\_STS register, and then configure CNT[7:0] = 255 for continuous transfer, the remaining bytes are 345 - 255 = 90
  - Step 4: After the completion of the second 255-byte data transfer, the TCRLD is set in the I2C\_STS register, and then set RLDEN = 0 to disable reload mode before setting CNT[7:0] = 90 for continuous transfer.

There are two ways to stop the last data transfer (RLDEN = 0, reload mode is disabled):

- Stop data transfer automatically (ASTOPEN = 1 in the I2C\_CTRL2)
  - When the number of data programmed in the CNT[7:0] bit has been fully transferred, the master will automatically send a STOP condition.
- Stop data transfer by software (ASTOPEN = 0 in the I2C\_CTRL2)
  - When the number of data programmed in the CNT[7:0] has been fully transferred, the TDC will be set in the I2C\_STS register, and the SCL, at this point, will be pulled low, an interrupt generated if TDCIEN is enabled. In this case, it is possible to send a STOP condition by setting GENSTOP = 1 in the I2C\_CTRL2 register, or send a RESTART condition by setting GENSTART = 1 in the I2C\_CTRL2 register, before clearing TDC flag by software.

### Byte control through slave

This feature is enabled by setting the SCTRL bit in the I2C\_CTRL2 register so that the slave is able to control ACK/NACK signals of each byte independently.

- Proceed as below:
  - Set SCTRL = 1 to enable Byte Control Through Slave.
  - After the slave address is matched (ADDRF = 1), enable reload mode by setting RLDEN = 1, and then set CNT[7:0] = 1.
  - When a byte is received, the TCRLD is set in the I2C\_STS register, and the slave will pull the SCL bus low between the 8th and 9th clock edges. At this point, the user can read the RXDT register and generate an ACK or NACK signal through the NACKEN bit in the I2C\_CTRL2 register.
  - When an NACK signal is generated, it indicates the end of communication.
  - When an ACK signal is generated, the communication flow keeps going on. At this point, set CNT[7:0] = 1, the TCRLD flag is cleared automatically by hardware, and the SCL bus is released for the reception of the next byte.

As we know, the value in the CNT[7:0] bit is not limited to 1. If you want to receive 8 data, for example, but just want to control the ACK/NACK signals of the 8<sup>th</sup> data. Proceed as below: set CNT[7:0] = 8, the slave will receive 7 consecutive data, with ACK signals sent. Once the 8<sup>th</sup> data reception is completed, the SCL bus is pulled low, and then proceed as above to select whether to send an ACK or NACK.

It should be noted that the clock stretching capability must be enabled (STRETCH = 0 in the I2C\_CTRL1 register) before selecting Byte Control Mode Through Slave.

**Table 31. I<sup>2</sup>C configuration table**

| Description  | RLDEN | ASTOPEN | SCTRL |
|--|-------|---------|-------|
| Master transmit/receive RESTART                    | 0     | 0       | ×     |
| Master transmit/receive STOP                       | 0     | 1       | ×     |
| Slave receive (control ACK/NACK of each byte)      | 1     | ×       | 1     |
| Slave transmit/receive (ACK response to all bytes) | ×     | ×       | 0     |

## I<sup>2</sup>C master communication flow

### I<sup>2</sup>C clock initialization (by setting the I2C\_CLKCTRL register)

- I<sup>2</sup>C clock divider: DIV[7:0]
- Data hold time ( $t_{HD,DAT}$ ): SDAD[3:0]
- Data setup time ( $t_{SU,DAT}$ ): SCLD[3:0]
- SCL high duration: SCLH[7:0]
- SCL low duration: SCLL[7:0]

### Set the number of bytes to be transferred

- ≤255 bytes
  - Disable reload mode by setting RLDEN = 0 in the I2C\_CTRL2 register
  - Set CNT[7:0] = N in the I2C\_CTRL2 register
- >255 bytes
  - Enable reload mode by setting RLDEN = 1 in the I2C\_CTRL register
  - Set CNT[7:0] = 255 in the I2C\_CTRL2 register
  - Remaining bytes N = N - 255

### End of data transfer

- ASTOPEN = 0: stop data transfer by software. After the completion of data transfer, the TDC is set in the I2C\_STS register, and GENSTOP = 1 or GENSTART = 1 is written by software to send a STOP or START condition
- ASTOPEN = 1: data transfer is stopped automatically. A STOP condition is sent at the end of data transfer

### Set slave address

- Set slave address value (by setting the SADDR bit in the I2C\_CTRL2 register)
- Set slave address mode (by setting the ADDR10 bit in the I2C\_CTRL2 register)
  - ADDR10 = 0: 7-bit address mode
  - ADDR10 = 1: 10-bit address mode

### Set transfer direction (by setting the DIR bit in the I2C\_CTRL2 register)

- DIR = 0: Master reception
- DIR = 1: Master transmission



### Start data transfer

When GENSTART = 1 in the I2C\_CTRL2 register, the master starts sending a START condition and slave address. After receiving the ACK from the slave, ADDR\_F = 1 is asserted in the I2C\_STS register. The ADDR\_F flag can be cleared by setting ADDR\_C = 1 in the I2C\_CLR register, and then data transfer starts.

### Master transmit

- I2C\_TXDT data register is empty, the shift register is empty, TDIS = 1 in the I2C\_STS register;
- Writing 1 to the TXDT register, and data is immediately moved to the shift register;
- TXDT register becomes empty, TDIS = 1 again;
- Writing 2 to the TXDT register, TDIS is cleared;
- Repeat step 2 and 3 until the data in the CNT[7:0] is sent;
- If TCRLD = 1 (reload mode) in the I2C\_STS register, the following two circumstances should be noted:

Remaining bytes  $N > 255$ : write 255 to the CNT bit,  $N = N - 255$ , TCRLD is cleared, and data transfer continues.

Remaining bytes  $N \leq 255$ : Disable reload mode (RLDEN = 0), write N to the CTN bit, TCRLD is cleared, and data transfer continues.

### Master receive

- After the slave address is matched, ADDR\_F = 1 in the I2C\_STS register, clear ADDR\_F flag by setting ADDR\_C = 1 in the I2C\_CLR register, and then it starts sending data.
  - After the reception of data, RDBF = 1, read the RXDT register will clear the RDBF automatically
  - Repeat step 2 until the reception of data programmed in the CNT[7:0] bit.
  - If TCRLD = 1 (reload mode) in the I2C\_STS, the following two circumstances should be noted:
- Remaining bytes  $N > 255$ : write 255 to the CNT bit,  $N = N - 255$ , TCRLD is cleared, and data transfer continues.
- Remaining bytes  $N \leq 255$ : Disable reload mode (RLDEN = 0), write N to the CTN bit, TCRLD is cleared, and data transfer continues.
- After the reception of the last data, an NACK signal will be sent by master.

### STOP condition

- STOP condition generation:
  - ASTOPEN = 0: TDC = 1 in the I2C\_STS register, set GENSTOP = 1 to generate a STOP condition.
  - ASTOPEN = 1: A STOP condition is generated automatically.
- Wait for the generation of a STOP condition, when a STOP condition is generated, STOPF = 1 is asserted in the I2C\_STS register. The STOPF flag can be cleared by setting STOPC = 1 in the I2C\_CLR register, and then transfer stops.

When the host receives an NACK signal during transmission, then ACKFAIL is set in the I2C\_STS register, and a STOP condition is sent to stop communication, whatever mode (either ASTOPEN = 0 or ASTOPEN = 1).

### Master transmitter

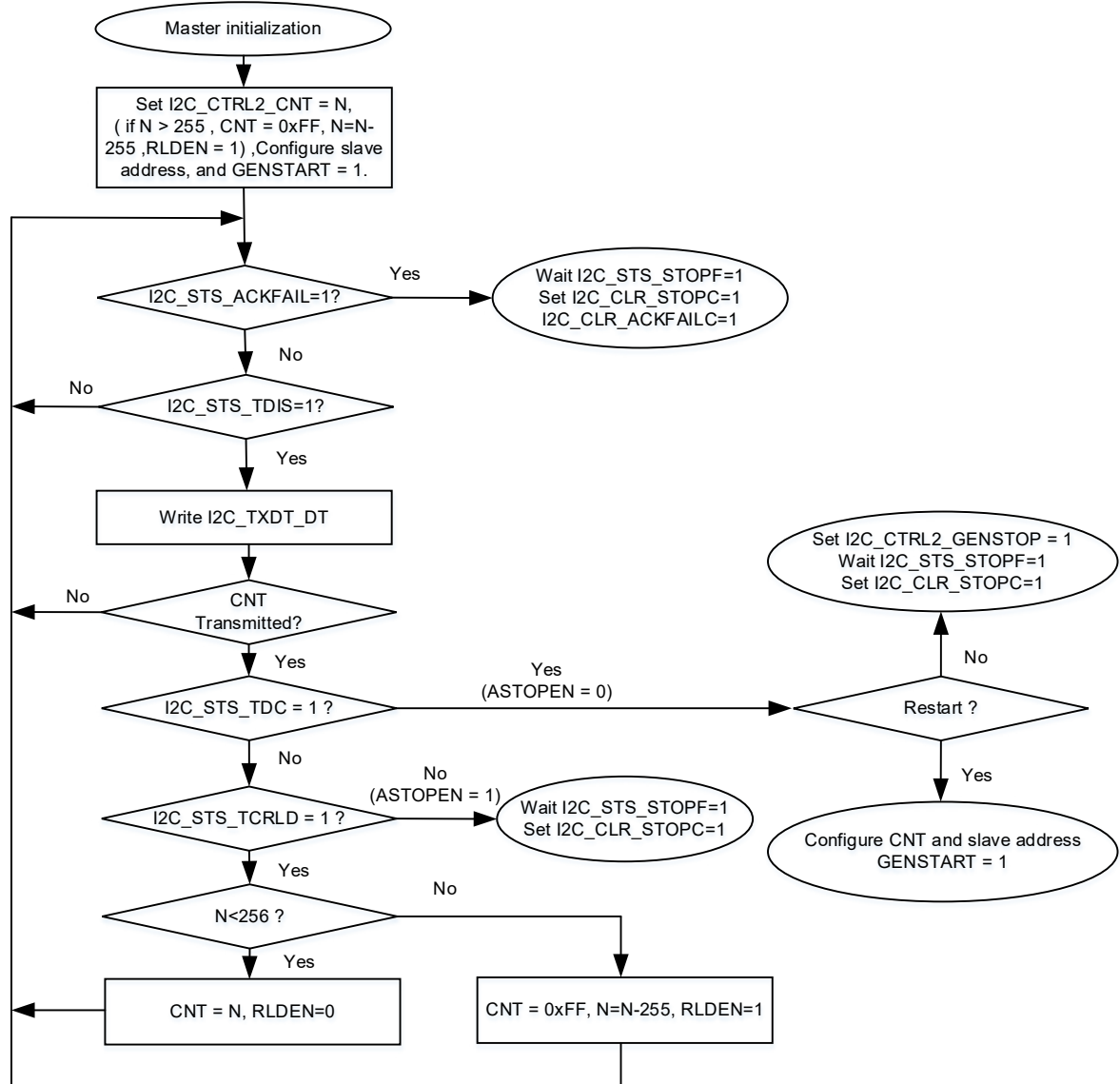


Figure 31. I<sup>2</sup>C master transmission flow

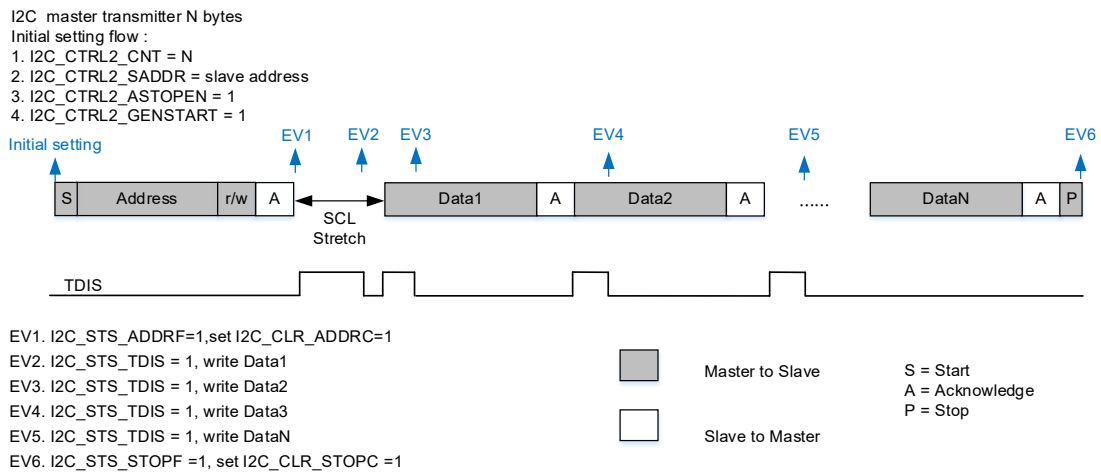


Figure 32. Transfer sequence of I<sup>2</sup>C master transmitter

### Master receiver

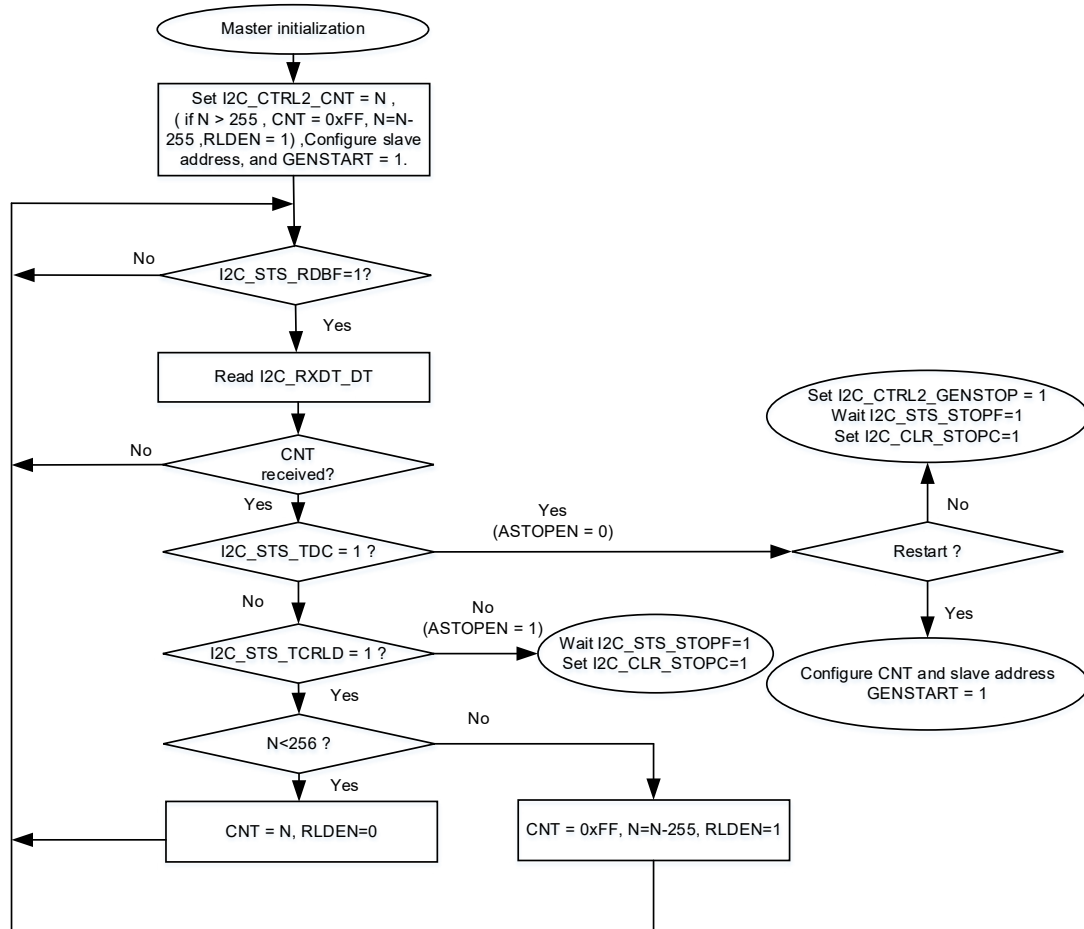
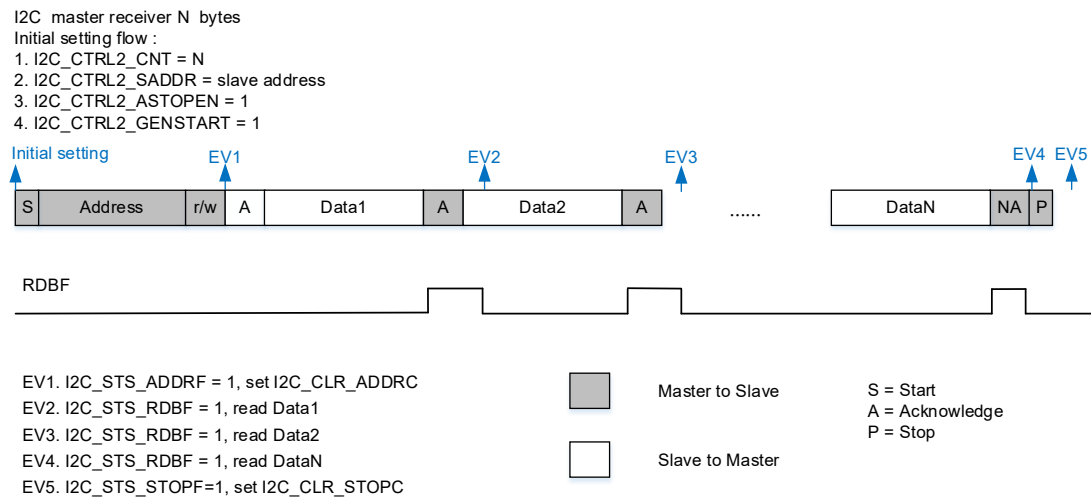


Figure 33. I<sup>2</sup>C master receive flow



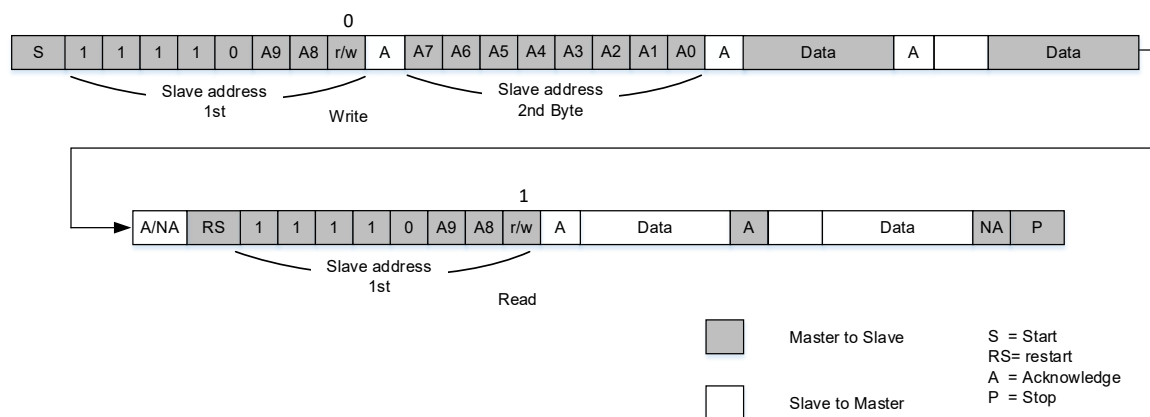
**Figure 34. Transfer sequence of I2C master receiver**

### Master special transfer sequence

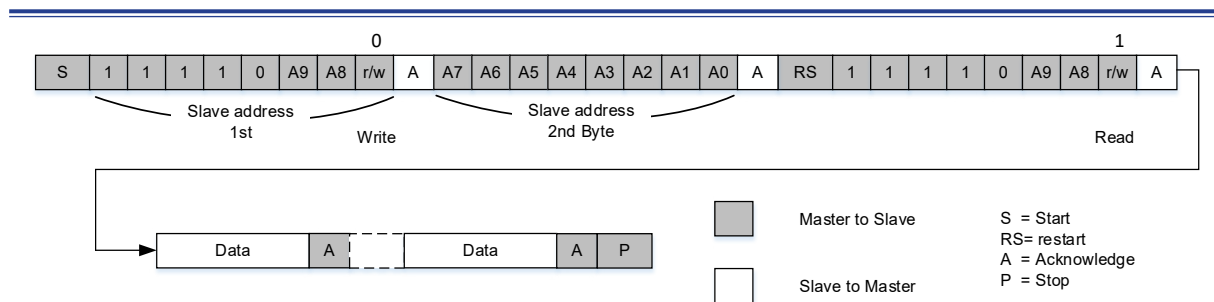
In 10-bit addressing mode, the READH10 bit of the I2C\_CTRL2 register is used to generate a special timing. When READH10 = 1, the master sends data to the slave before read access to the slave, as shown in the figure below:

Operating method:

When ASTOPEN = 0, data is transferred from the master to the slave. At the end of data transfer, READH10 = 0 is asserted, and then the master starts receiving data from the slave.



**Figure 35. 10-bit address read access when READH10 = 1**



**Figure 36. 10-bit address read access when READH10 = 0**

## I<sup>2</sup>C slave communication flow

### I<sup>2</sup>C clock initialization (by setting the I2C\_CLKCTRL register)

- I<sup>2</sup>C clock divider: DIV[7:0]
- Data hold time ( $t_{HD,DAT}$ ): SDAD[3:0]
- Data setup time ( $t_{SU,DAT}$ ): SCLD[3:0]

### Set local address 1

- Set address mode:
  - 7-bit address: by setting ADDR1MODE = 0 in the I2C\_OADDR register
  - 10-bit address: by setting ADDR1MODE = 1 in the I2C\_OADDR register
- Set address 1: by setting the ADDR1 bit in the I2C\_OADDR1 register
- Enable address 1: by setting ADDR1EN = 1 in the I2C\_OADDR1 register

### Set local address 2

- Set address 2: by setting the ADDR2 bit in the I2C\_OADDR2 register.
- Set address 2 mask bit: by setting the ADDR2MASK bit in the I2C\_OADDR2 register.
- Enable address 2: by setting ADDR2EN = 1 in the I2C\_OADDR2 register.

Note: Only 7-bit address mode is available in the address 2 mode. The ADDR2MASK bit is used to mask some address bits freely so that the slave can respond to some specific addresses. Refer to Section 14.2 for more information about the ADDR2MASK bit.

In the case of using only one address, only address 1 needs to be configured, without the need of address 2 mode.

### Wait for address matching

When the local address is received, the ADDRFB bit is set in the I2C\_STS register. The data transfer direction can be obtained by read access to the SDIR bit in the I2C\_STS register. When SDIR = 0, it indicates that the slave is receiving data, whereas SDIR = 1 indicates that the slave is sending data. The ADDR[6:0] bit of the I2C\_STS register indicates what kind of address has been received, which is particularly helpful in the case when the dual address mode is used and the address 2 mode mask bit is set.

Data transfer starts when the ADDRFB is cleared by setting ADDRCL = 1 of the I2C\_CLR register.

### Data transfer (slave transmission, clock stretching enabled, STRETCH = 0)

After address matching:

1. I2C\_TXDT data register becomes empty, the shift register becomes empty, and TDIS = 1 in the I2C\_STS register.
2. Data is then transferred to the shift register after writing 1 to the TXDT register.
3. The TXDT register then becomes empty, and the TDIS is set again.
4. TDIS is cleared by writing 2 to the TXDT register.
5. Repeat step 3 and 4 until the completion of data transfer.
6. Wait for the generation of an NACK signal. Once received, the ACKFAILF is set in the I2C\_STS register. The ACKFAILF flag is cleared by writing 1 to the ACKFAILC.
7. Wait for the generation of a STOP condition. Once received, the STOPF is set in the I2C\_STS register. At the end of data transfer, the STOPF is cleared by writing 1 to the STOPC, transmission ends.

In the case of the clock stretching being disabled (STRETCH = 1), if data has not yet been written to the TXDT register before the transmission of the first bit of the to-be-transferred data (that is, before the generation of SDA edge), an underrun error may occur, and the OUF bit is set in the I2C\_STS register, sending 0xFF to the bus.

In order to write data in time, data must be written to the DT register first before communication, in two different ways:

- Write operation through software: Clear the TXDT register by setting the TDBE bit through software, and then write the first data to the TXDT register, the TDBE is cleared.
- Write operation through interrupts or DMA: Clear the TXDT register by setting the TDBE bit through software, then set the TDIS bit to generate a TDIS event, which generates an interrupt or DMA request. At this point, data is written to the TXDT register using DMA or interrupt functions.

### Data transfer (slave receive, clock stretching enabled, STRETCH = 0)

After address matching:

1. I2C\_RXDT register becomes empty, the shift register becomes empty, and RDBF=0 in the I2C\_STS register.
2. Upon the receipt of data, RDBF=1; The RDBF is cleared by read operation to the RXDT register.
3. Repeat step 2 until the completion of all data transfer.
4. Wait for the generation of a STOP condition. Once received, the STOPF is set in the I2C\_STS register. The STOPF can be cleared by writing 1 to the STOPC bit in the I2C\_CLR register, transfer ends.

In slave receive mode, the slave byte control mode can be used for data reception. This mode allows to control ACK/NACK signals of each byte received. This mode is typically available in SMBus protocol. Refer to Section 11.4.2 for more information about this mode.

Note that the slave must read the received data in the case of the clock stretching being disabled (STRETCH = 1). If one-byte data has been received and data is not read yet before the end of the next data reception, an overrun error occurs, setting the OUF bit in the I2C\_STS register, and sending NCAK.

An interrupt will be generated if the corresponding interrupt enable bit is enabled. For more information about interrupt generation, refer to the interrupt chapter.

### Slave transmission

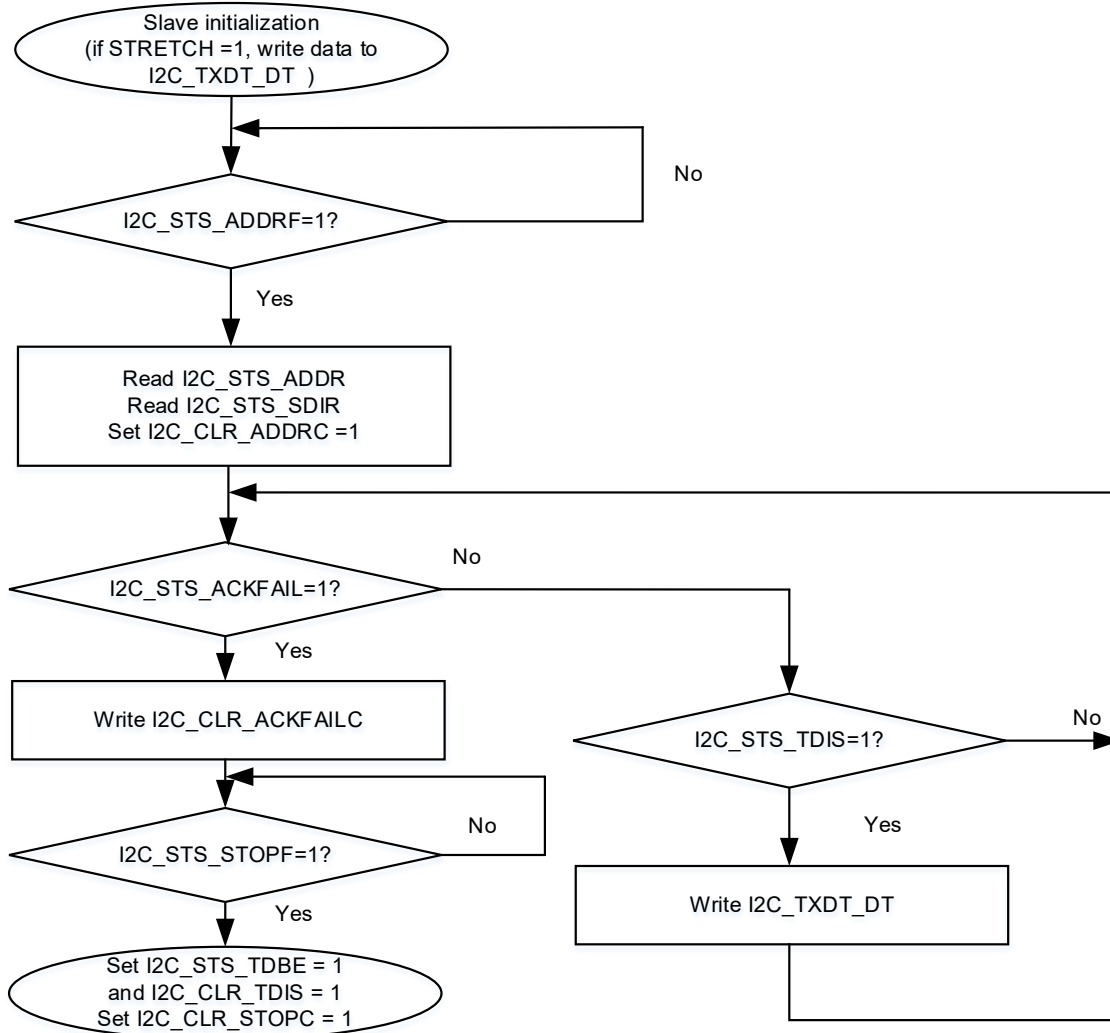
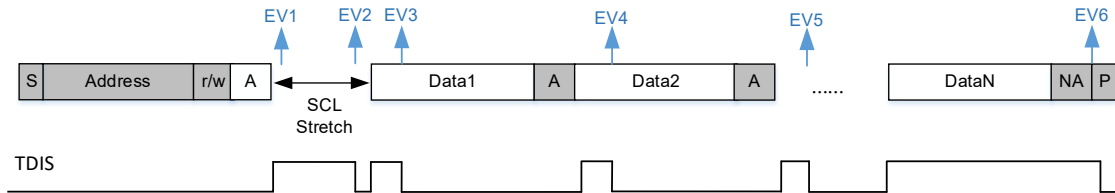


Figure 37. I<sup>2</sup>C slave transmission flow

I2C Slave transfer N bytes to I2C master



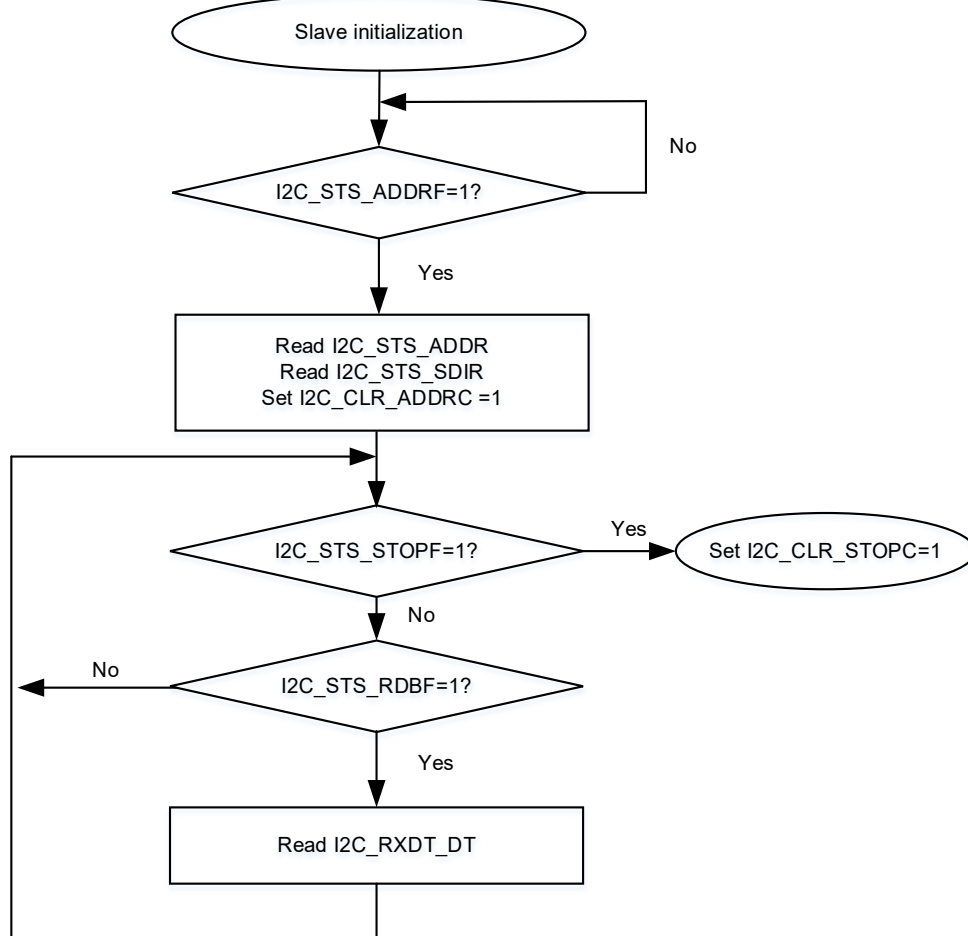
EV1. I2C\_STS1\_ADDRF = 1, set I2C\_CLR\_ADDRF = 1  
 EV2. write Data1, I2C\_STS1\_TDIS = 0  
 EV3. write Data2, I2C\_STS1\_TDIS = 0  
 EV4. write Data3, I2C\_STS1\_TDIS = 0  
 EV5. write DataN, I2C\_STS1\_TDIS = 0  
 EV6. I2C\_STS\_STOPF, set I2C\_CLR\_STOPC

Master to Slave  
 Slave to Master

S = Start  
 A = Acknowledge  
 P = Stop

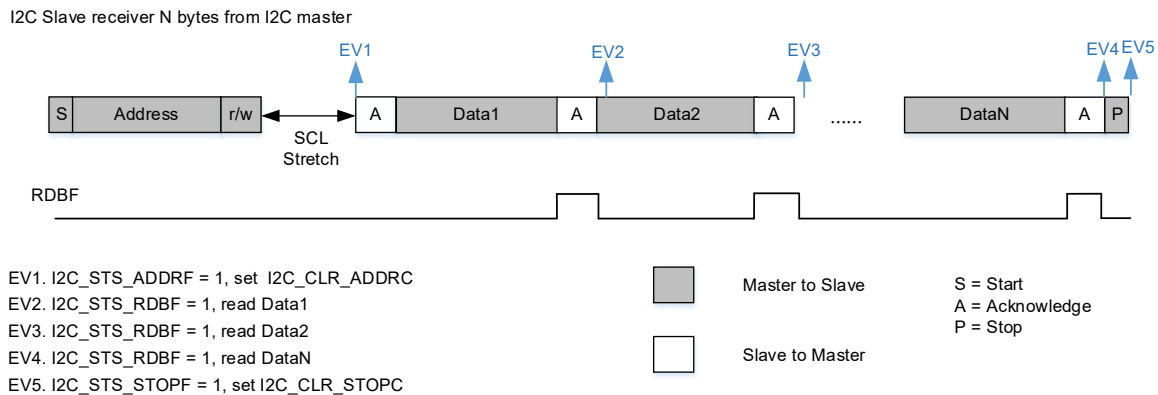
**Figure 38. I2C slave transmission timing**

### Slave receive



**Figure 39. I2C slave receive flow**





**Figure 40. I<sup>2</sup>C slave receive timing**

## SMBus

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other. It is based on I<sup>2</sup>C. With SMBus, the device can provide manufacturer information, tell the system its model/part number, report different types of errors and accept control parameters and so on. For more information, refer to SMBus 2.0 protocol.

### Differences between SMBus and I<sup>2</sup>C

1. SMBus requires a minimum speed of 10 kHz for the purpose of management and monitor. It is quite easy to know whether the bus is in Idle state or not as long as a parameter is input while running on a certain transmission speed, without the need of detecting the STOP signals one after another, or even keeping STOP and other parameter monitor. There is no limit for I<sup>2</sup>C.
2. SMBus transmission speed ranges from 10 kHz to 100 kHz. In contrast, I<sup>2</sup>C has no minimum requirement, and its maximum speed varies from one mode to another, namely, 100 kHz in standard mode and 400 kHz in fast mode.
3. After reset, SMBus needs timeout, but there is no limit for I<sup>2</sup>C in this regard.

### SMBus address resolution protocol (ARP)

SMBus address conflicts can be resolved by dynamically assigning a new unique address to each device. Refer to SMBus 2.0 protocol for more information about ARP.

Setting the DEVADDREN bit in the I2C\_CTRL1 register can enable the I<sup>2</sup>C interface to recognize the default device address (0b1100001x). However, unique device identifier (UDID) and the detailed protocol implementation should be handled by software.

### SMBus host notify protocol

The slave device can send data to the master device through SMBus host notify protocol. For example, the slave can notify the host to implement ARP with this protocol. Refer to SMBus 2.0 protocol for details on SMBus host notify protocol.

In host mode (HADDREN = 1), the I<sup>2</sup>C interface is enabled to recognize the 0b0001000x (default host address).

### SMBus Alert

SMBALERT is an optional signal that connects the ALERT pin between the host and the slave. With this signal, the slave notifies the host to access the slave. SMBALERT is a wired-AND signal. For more information about SMBus Alert, refer to SMBus2.0 protocol.

The detailed sequences are as follows:

SMBus host:

1. Enable SMBus Alert mode by setting SMBALERT = 1;
2. Enable ALERT interrupt if necessary;
3. When an alert event occurs on the ALERT pin (ALERT pin changes from high to low);
4. The host will generate ALERT interrupt if enabled;
5. The host then processes the interrupt and accesses to all devices through ARA (Alert Response Address 0001100x) so as to get the slave addresses. Only the devices with pulled-down SMBALERT can acknowledge ARA.
6. The host then continues to operate based on the slave addresses available.

### SMBus slave:

1. When an alert event occurs and the ALERT pin changes from high to low (SMBALERT = 1), the slave responds to ARA (Alert Response Address) address (0001100x);
2. Wait until the host gets the slave addresses through ARA;
3. Report its own address, but it continues to wait if the arbitration is lost;
4. Address is reported properly, and the ALERT pin is released (SMBALERT = 0).

### Packet error checking (PEC)

Packet error checking (PEC) is used to guarantee the correctness and integrity of data transfer. This is done by using CRC-8 polynomial:

$$C(x) = x^8 + x^2 + x + 1$$

PEC calculation is enabled when PECEN = 1 to check address and data.

PEC transfer:

- Host: PEC transfer is enabled by setting PECTEN = 1 in the I2C\_CTRL2 register. The host sends a PEC as soon as the number of data transfer reaches N-1 (CNT = N).
- Slave: PEC transfer is enabled by setting PECTEN = 1 in the I2C\_CTRL2 register. When the number of data transfer reaches N-1 (CNT = N), the slave will consider the Nth data as a PEC and check it. A NACK will be sent if the PEC checking result is not correct, setting the PECERR flag in the I2C\_STS register. In case of slave transmission mode, a NACK must follow the PEC whatever the checking result.

### SMBus timeout

The SMBus protocol specifies three timeout detection modes:

- Low level timeout ( $t_{\text{TIMEOUT}}$ ): The time duration when the SCL is kept low in a single mode (taking into account master/slave device, however actively or passively pulled low).
- Cumulative timeout for a slave device at low level ( $t_{\text{LOW:SEXT}}$ ): The cumulative time duration when the SCL is pulled low by a slave device during the period from a START condition to a STOP condition.

- Cumulative timeout for a master device at low level ( $t_{\text{LOW:MEX}}$ ): The cumulative time duration when the SCL is pulled low by a master device during the period from the ACK of the last byte to the 8th bit of the next byte (a single byte).

It should be noted that both  $t_{\text{LOW:SEXT}}$  and  $t_{\text{LOW:MEXT}}$  only deal with the time when they set themselves low level, excluding the time when they are pulled low by external sources. In contrast, both of these cases are considered in the calculation of  $t_{\text{TIMEOUT}}$ .

**Table 32. SMBus timeout specification**

| Type of timeout       | Min | Max | Unit |
|-----------------------|-----|-----|------|
| $t_{\text{TIMEOUT}}$  | 25  | 35  | ms   |
| $t_{\text{LOW:SEXT}}$ | —   | 25  | ms   |
| $t_{\text{LOW:MEXT}}$ | —   | 10  | ms   |

The I<sup>2</sup>C peripherals embeds two counters for timeout detection, which can be configured through the I2C\_TIMEOUT register. When a timeout event occurs, the TMOUT is set in the I2C\_STS register. The TMOUT bit can be cleared by writing 1 to the TMOUTC bit in the I2C\_CLR register.

- EXTTIME: This is used to the cumulative timeout detection for master/slave devices at low level  
Timeout duration =  $(\text{EXTTIME} + 1) \times 2048 \times T_{\text{I2C\_CLK}}$
- TOTIME: This is used for clock level timeout detection, selected through the TOMODE bit.  
TOMODE = 0: Low level timeout detection, timeout duration =  $(\text{TOTIME} + 1) \times 2048 \times T_{\text{I2C\_CLK}}$   
TOMODE = 1: High level timeout detection, timeout duration =  $(\text{TOTIME} + 1) \times 4 \times T_{\text{I2C\_CLK}}$

**Table 33. SMBus timeout detection configuration**

| Type of timeout       | Other configurations | Enable bit | Timeout calculation   |
|-----------------------|----------------------|------------|---|
| $t_{\text{TIMEOUT}}$  | TOMODE = 0           | TOEN = 1   | $(\text{TOTIME} + 1) \times 2048 \times T_{\text{I2C\_CLK}}$  |
| $t_{\text{LOW:SEXT}}$ | —                    | EXTEN = 1  | $(\text{EXTTIME} + 1) \times 2048 \times T_{\text{I2C\_CLK}}$ |
| $t_{\text{LOW:MEXT}}$ | —                    | EXTEN = 1  | $(\text{EXTTIME} + 1) \times 2048 \times T_{\text{I2C\_CLK}}$ |

### Slave receive byte control

In slave receive mode, the slave receive byte control mode (SCTRL = 1) can be used to control ACK/NACK signals of each received byte. Refer to section 11.4.2 for more information.

**Table 34. SMBus mode configuration**

| Transfer mode                     | PECEN | PECTEN | RLDEN | ASTOPEN | SCTRL |
|-----------------------------------|-------|--------|-------|---------|-------|
| Master receive/transmit + STOP    | 1     | 1      | 0     | 1       | —     |
| Master receive/transmit + RESTART | 1     | 1      | 0     | 0       | —     |
| Slave receive                     | 1     | 1      | 1     | —       | 1     |
| Slave transmit                    | 1     | 1      | 0     | —       | —     |

### How to use the interface in SMBus mode

- Set SMBus default address acknowledgement:  
HADDREN = 1: Master default address acknowledged (0b0001000x)  
DEVADDREN = 1: Device default address acknowledged (0b1100001x)
- Configure PEC
- Slave receive byte control mode can be enabled (with SCTRL bit in the I2C\_CTRL1) in slave mode, if necessary
- Other configurations follow the I<sup>2</sup>C

However, the detailed SMBus protocol implementation should be handled by software, since the I<sup>2</sup>C interface is only enabled to recognize the addresses of SMBus protocols.

## SMBus master communication flow

The SMBus is similar to the I<sup>2</sup>C in terms of master communication flow.

### I<sup>2</sup>C clock initialization (by setting the I2C\_CLKCTRL register)

- I<sup>2</sup>C clock divider: DIV[7:0]
- Data hold time ( $t_{HD,DAT}$ ): SDAD[3:0]
- Data setup time ( $t_{SU,DAT}$ ): SCLD[3:0]
- SCL high duration: SCLH[7:0]
- SCL low duration: SCLL[7:0]

### SMBus-related initialization

- Select SMBus host: host default address acknowledged (0b0001000x) by setting HADDREN=1
- Enable PEC calculation: Set PECEN = 1 in the I2C\_CTRL1 register
- Enable PEC transfer: Set PECTEN = 1 in the I2C\_CTRL2 register

### Set the number of bytes to be transferred

- Disable reload mode by setting RLDEN = 0 in the I2C\_CTRL2 register
- Set CNT[7:0] = N in the I2C\_CTRL2 register

The number of bytes to be transferred is <255 in SMBus mode at one time.

### End of data transfer

- ASTOPEN = 0: stop data transfer by software. After the completion of data transfer, the TDC is set in the I2C\_STS register, and GENSTOP = 1 or GENSTART=1 is written by software to send a STOP or START condition
- ASTOPEN = 1: data transfer is stopped automatically. A STOP condition is sent at the end of data transfer

### Set slave address

- Set slave address value (by setting the SADDR bit in the I2C\_CTRL2 register)
- Set 7-bit slave address mode (by setting the ADDR10=0 in the I2C\_CTRL2 register)

### Set transfer direction (by setting the DIR bit in the I2C\_CTRL2 register)

- DIR = 0: Master reception
- DIR = 1: Master transmission

### Start data transfer

In case of GENSTART = 1 in the I2C\_CTRL2 register, the master starts sending a START condition and slave address. After receiving the ACK from the slave, ADDR<sub>F</sub> = 1 is asserted in the I2C\_STS register. The ADDR<sub>F</sub> flag can be cleared by setting ADDR<sub>C</sub> = 1 in the I2C\_CLR register, and then data transfer starts.

### Master transmit

- I2C\_TXDT data register is empty, the shift register is empty, TDIS = 1 in the I2C\_STS register;
- Writing 1 to the TXDT register, and data is immediately moved to the shift register;
- TXDT register becomes empty, TDIS = 1 again;
- Writing 2 to the TXDT register, TDIS is cleared;
- Repeat step 2 and 3 until the specified data (N-1) is sent;
- The master will automatically transmit the Nth data, that is, PEC.

### Master receive

- After the reception of data, RDBF = 1, read the RXDT register will clear the RDBF automatically.
- Repeat step 1 until the reception of the specified data (N). The Nth data is set as PEC. A NACK is automatically sent after the receipt of the Nth data (PEC) whatever the PEC result.

### STOP condition

- STOP condition generation:  
ASTOPEN = 0: TDC = 1 in the I2C\_STS register, set GENSTOP = 1 to generate a STOP condition  
ASTOPEN = 1: A STOP condition is generated automatically.
- Wait for the generation of a STOP condition, when a STOP condition is generated, STOPF = 1 is asserted in the I2C\_STS register. The STOPF flag can be cleared by setting STOPC = 1 in the I2C\_CLR register, and then transfer stops.

### SMBus master transmission flow

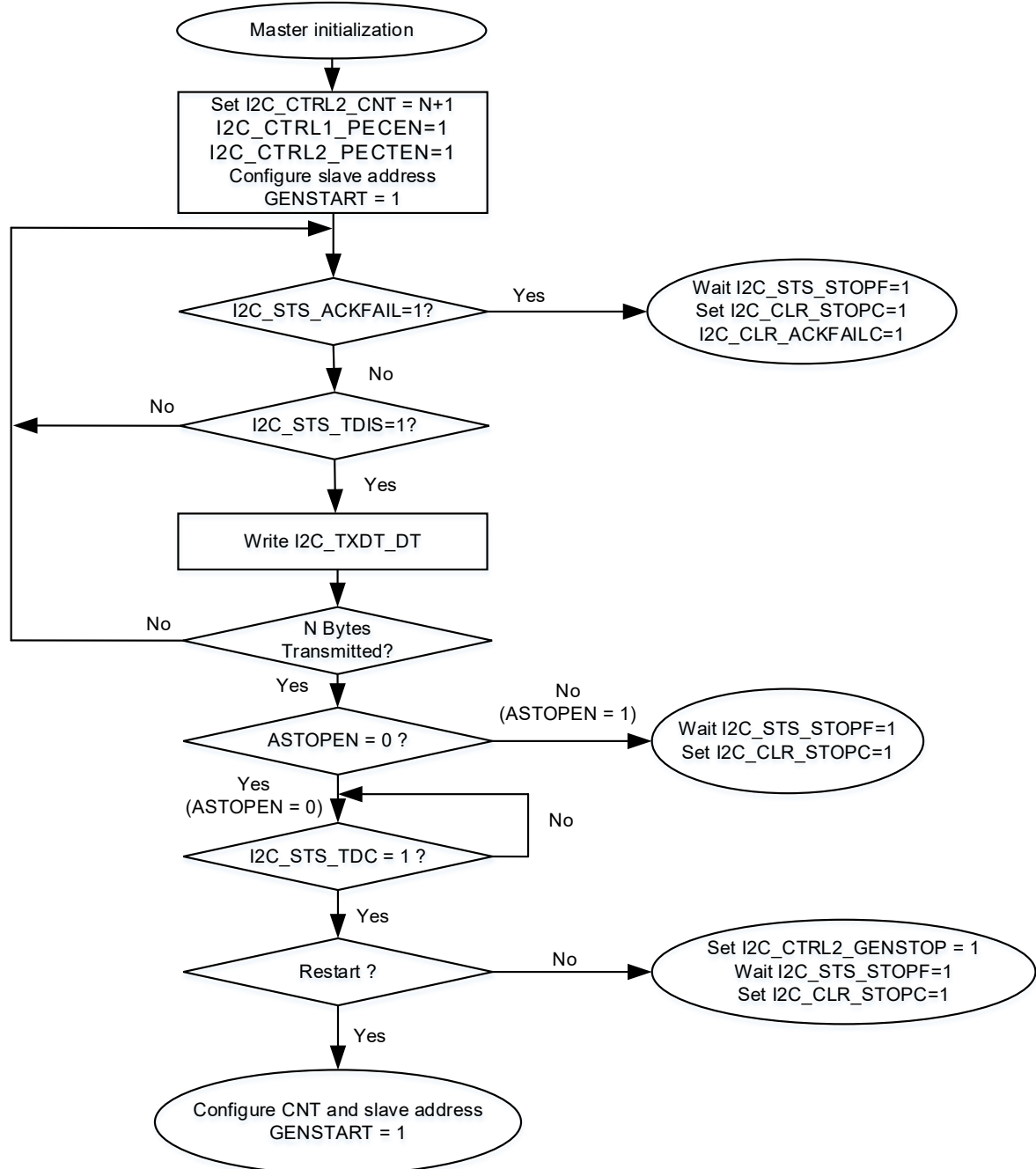
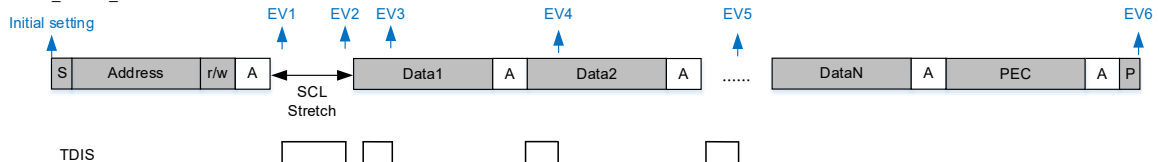


Figure 41. SMBus master transmission flow

SMBus master transmitter N bytes + PEC

Initial setting flow :

1. I2C\_CTRL2\_CNT = N+ 1
2. I2C\_CTRL2\_SADDR = slave address
3. I2C\_CTRL2\_ASTOPEN = 1
4. I2C\_CTRL1\_PECEN = 1
5. I2C\_CTRL2\_PECTEN = 1
6. I2C\_CTRL2\_GENSTART = 1



EV1. I2C\_STS\_ADDRF=1, set I2C\_CLR\_ADDRFC=1

EV2. I2C\_STS\_TDIS = 1 , write Data1

EV3. I2C\_STS\_TDIS = 1 , write Data2

EV4. I2C\_STS\_TDIS = 1 , write Data3

EV5. I2C\_STS\_TDIS = 1 , write DataN

EV6. I2C\_STS\_STOPF =1, set I2C\_CLR\_STOPC =1



Master to Slave

Slave to Master

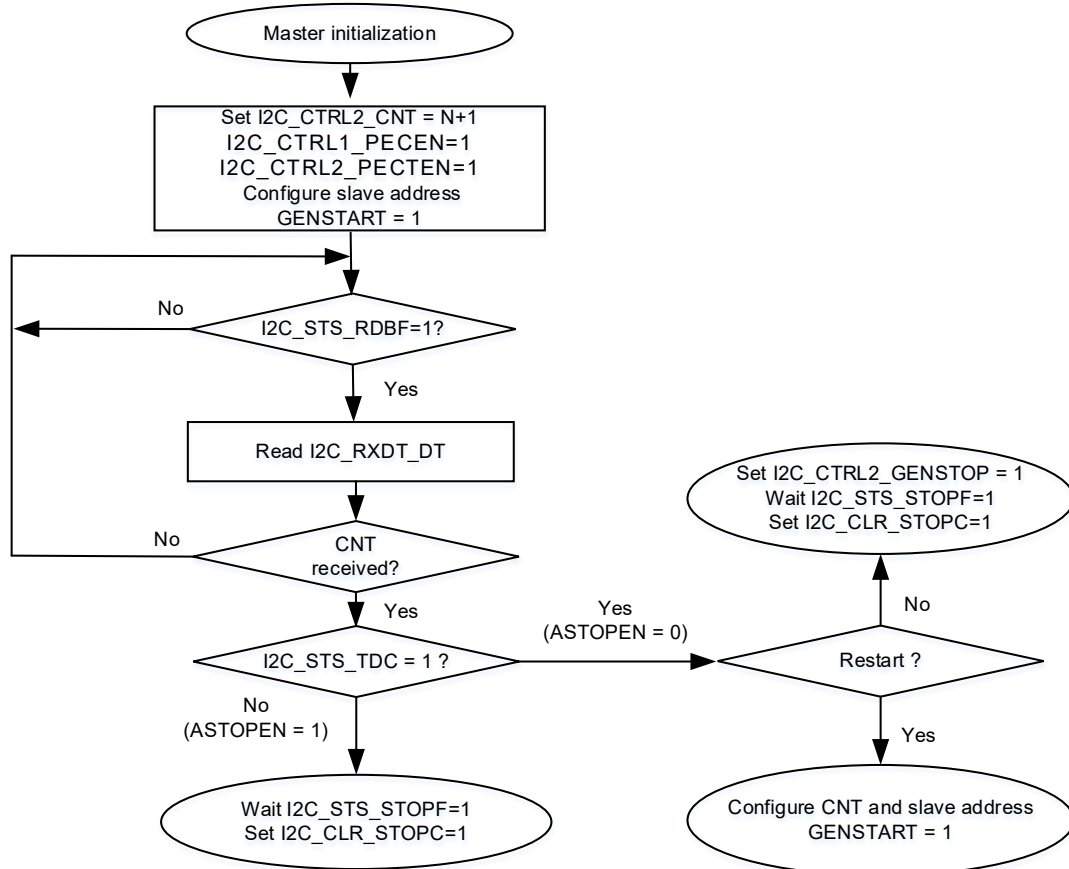
S = Start

A = Acknowledge

P = Stop

**Figure 42. SMBus master transmission timing**

### SMBus master receive flow



**Figure 43. SMBus master receive flow**

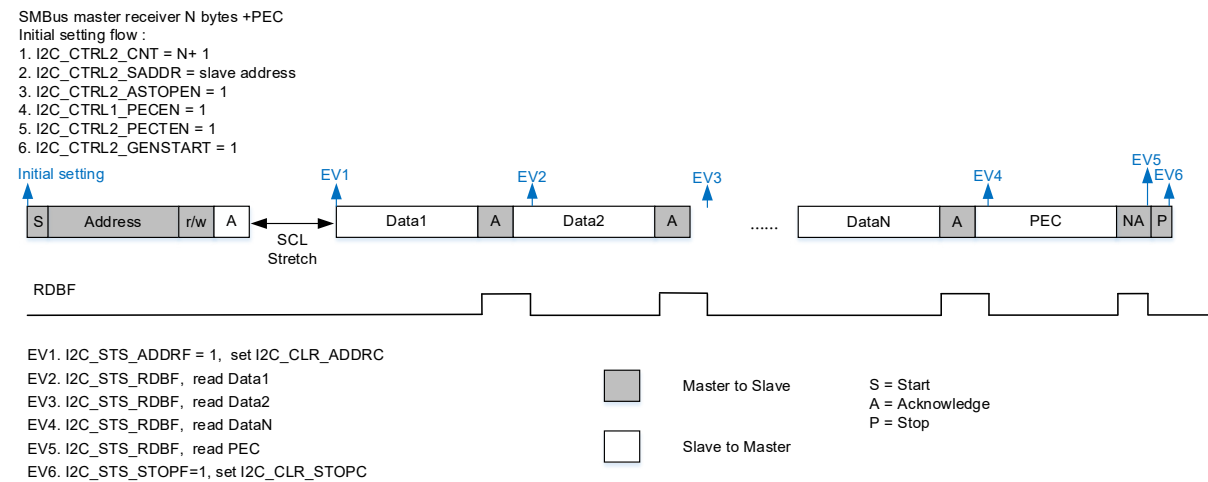


Figure 44. SMBus master receive timing

## SMBus slave communication flow

The SMBus is similar to the I<sup>2</sup>C in terms of slave communication flow.

### I<sup>2</sup>C clock initialization (by setting the I2C\_CLKCTRL register)

- I<sup>2</sup>C clock divider: DIV[7:0]
- Data hold time ( $t_{HD,DAT}$ ): SDAD[3:0]
- Data setup time ( $t_{SU,DAT}$ ): SCLD[3:0]

### Set local address

- Set 7-bit address mode: by setting ADDR1MODE = 0 in the I2C\_OADDR register;
- Set address 1: by setting the ADDR1 bit in the I2C\_OADDR1 register;
- Enable address 1: by setting ADDR1EN = 1 in the I2C\_OADDR1 register.

### SMBus-related initialization

- Select SMBus host: device default address acknowledged (0b1100001x) by setting DEVADDREN = 1;
- Enable PEC calculation: Set PECEN = 1 in the I2C\_CTRL1 register;
- Set slave byte control mode:

Slave transmit: disable byte control mode by setting SCTRL = 0 in the I2C\_CTRL1 register.

Slave receive: enable byte control mode by setting SCTRL = 1 in the I2C\_CTRL1 register.

### Wait for address matching

When the local address is received, the ADDR1F bit is set in the I2C\_STS register. The data transfer direction can be obtained by read access to the SDIR bit in the I2C\_STS register. When SDIR = 0, it indicates that the slave is receiving data, whereas SDIR = 1 indicates that the slave is sending data. The ADDR[6:0] bit of the I2C\_STS register indicates what kind of address has been received, which is particularly helpful in the case when the dual address mode is used and the address 2 mode mask bit is set.



Enable PEC transfer: by setting PECTEN = 1 in the I2C\_CTRL2 register.

Set the number of data to be transferred:

- Slave transmit: by setting CNT = N in the I2C\_CTRL2 register
- Slave receive: by setting CNT = 1 in the I2C\_CTRL2 register

Set reload mode:

- Slave transmit: by setting RLDEN = 0 in the I2C\_CTRL2 register
- Slave receive: by setting RLDEN = 1 in the I2C\_CTRL2 register

The ADDRIF flag can be cleared by setting ADDRC = 1 in the I2C\_CLR register, and then data transfer starts.

#### **Data transfer (slave transmission, clock stretching enabled, STRETCH = 0)**

After address matching:

1. I2C\_TXDT data register becomes empty, the shift register becomes empty, and TDIS = 1 in the I2C\_STS register.
2. Data is then transferred to the shift register after writing 1 to the TXDT register.
3. The TXDT register then becomes empty, and the TDIS is set again.
4. TDIS is cleared by writing 2 to the TXDT register.
5. Repeat step 3 and 4 until data (N-1) is sent.
6. The slave will automatically transmit the Nth data, that is, PEC.
7. Wait for the generation of an NACK signal. Once received, the ACKFAILF is set in the I2C\_STS register. The ACKFAILF flag is cleared by writing 1 to the ACKFAILC.
8. Wait for the generation of a STOP condition. Once received, the STOPF is set in the I2C\_STS register. At the end of data transfer, the STOPF is cleared by writing 1 to the STOPC, transmission ends.

#### **Data transfer (slave receive, clock stretching enabled, STRETCH=0)**

After address matching:

1. I2C\_RXDT register becomes empty, the shift register becomes empty, and RDBF = 0 in the I2C\_STS register.
2. Upon the receipt of one-byte data, RDBF = 1 and TCRLD = 1, then the SCL is pulled low by the slave.
3. The RDBF is cleared by read operation to the RXDT register.
4. NACKEN bit of the I2C\_CTRL register can be configured to generate an ACK or NACK, if needed.

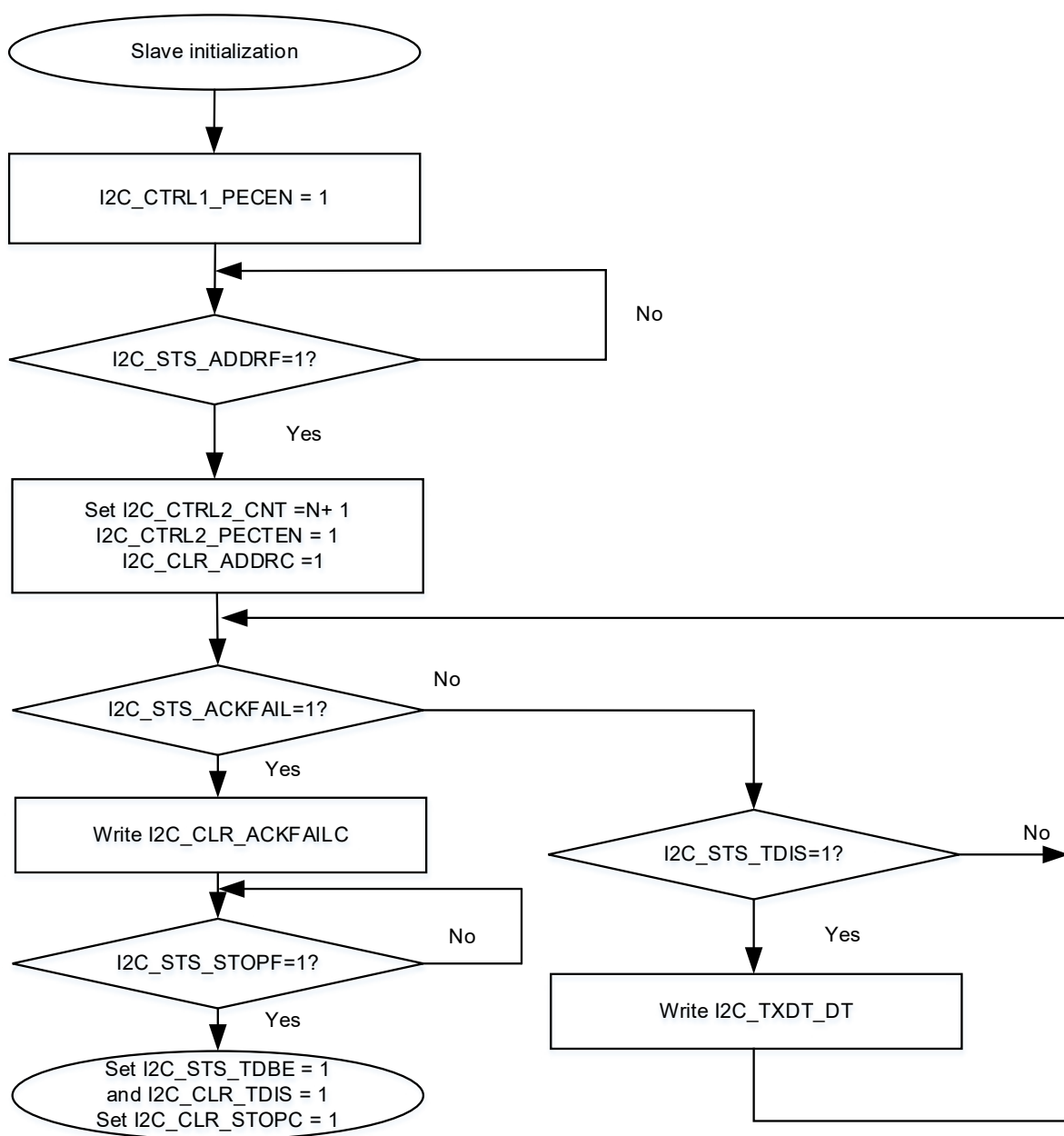
If a NACK is detected, it indicates the completion of communication.

If an ACK is detected, communication continues. Writing CNT = 1 will automatically clear the TCRLD flag by hardware, and the SCL is released by the slave for the reception of the next data.

5. Repeat steps 2/3/4 until the completion of data reception (N-1).
6. Set RLDEN = 0 of the I2C\_CTRL2 register to disable reload mode. Set CNT = 1 to repeat steps 2/3 to receive a PEC. The PECERR bit will be set if a PEC error occurs.

7. Wait for the generation of a STOP condition. Once received, the STOPF is set in the I2C\_STS register. The STOPF can be cleared by writing 1 to the STOPC bit in the I2C\_CLR register, transfer ends.

#### SMBus slave transmission



**Figure 45. SMBus slave transmission flow**

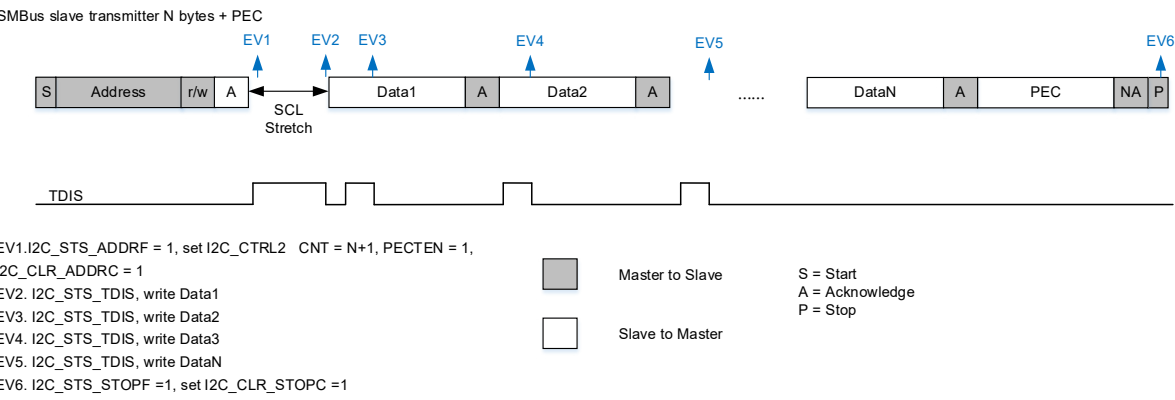


Figure 46. SMBus slave transmission timing

## SMBus slave receive

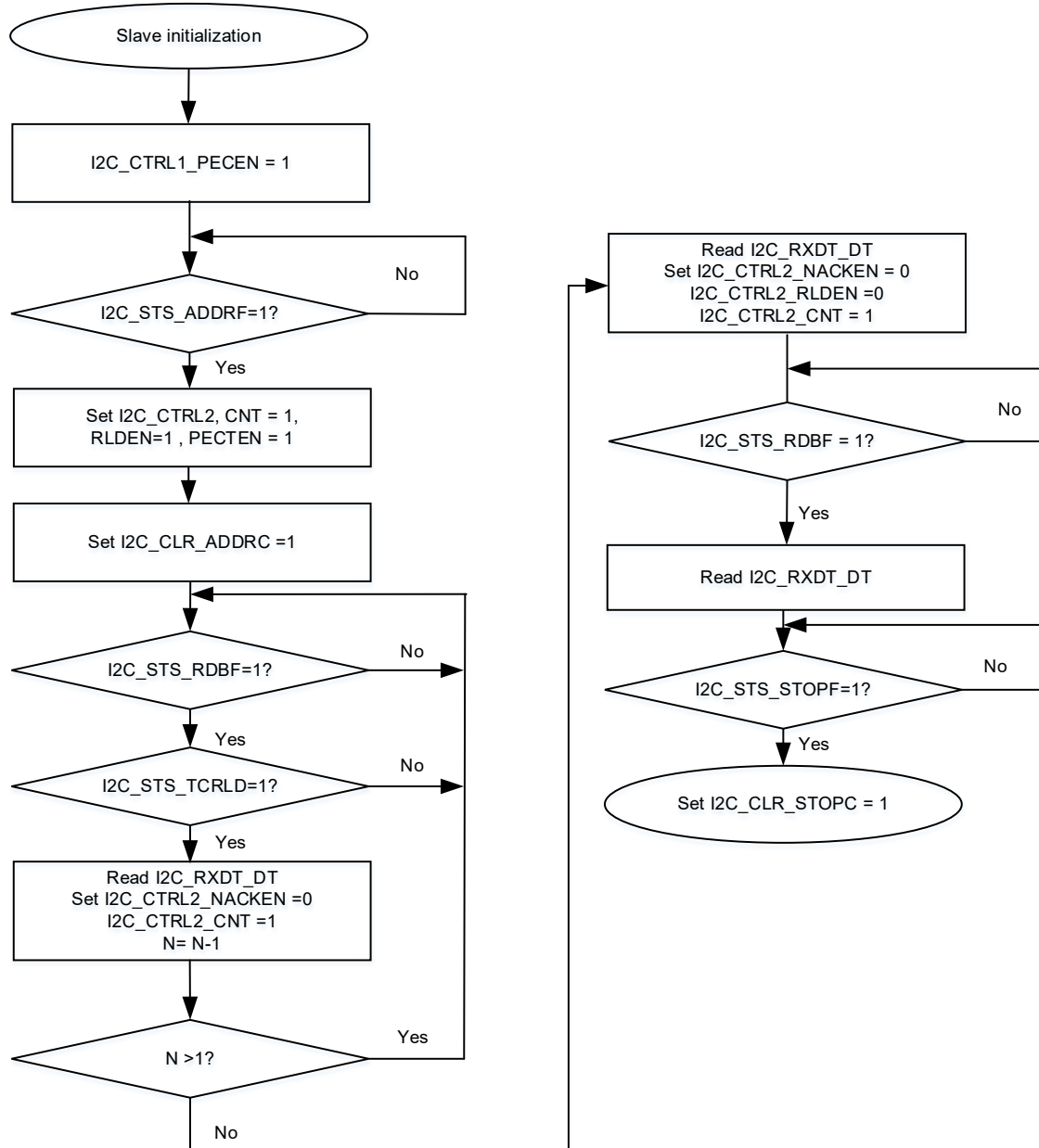
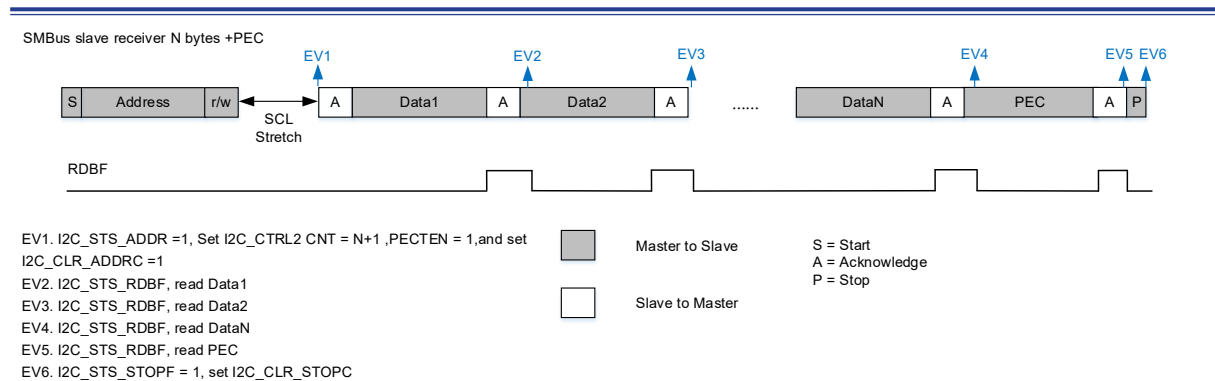


Figure 47. SMBus slave receive flow



**Figure 48. SMBus slave receive timing**

## Data transfer using DMA

I<sup>2</sup>C data transfer can be done using DMA controller so as to reduce the burden on the CPU. The TDLEN and RDLEN must be set 0 when using DMA for data transfer.

### Transmission using DMA (DMATEN = 1)

1. Set the peripheral address (DMA\_CxPADDR = I2C\_TXDT address).
2. Set the memory address (DMA\_CxMADDR = data memory address).
3. The transmission direction is set from memory to peripheral (DTD = 1 in the DMA\_CHCTRL register).
4. Configure the total number of bytes to be transferred in the DMA\_CxDTCNT register.
5. Configure other parameters such as priority, memory data width, peripheral data width, interrupts, etc., in the DMA\_CHCTRL register.
6. Enable the DMA channel by setting CHEN = 1 in the DMA\_CxCTRL register.
7. Enable I<sup>2</sup>C DMA request by setting DMAEN = 1 in the I2C\_CTRL2 register. Once the TDBE bit in the I2C\_STS1 register is set, the data is loaded from the programmed memory to the I2C\_DT register through DMA.
8. When the number of data transfers, programmed in the DMA controller, is reached (DMA\_CxDTCNT = 0), the data transfer is complete (An interrupt is generated if enabled).
9. Master transmitter: Once the TDC flag is set, the STOP condition is generated, indicating that transfer is complete.

Slave transmitter: Once the ACKFAIL flag is set, clear the ACKFAIL flag, transfer is complete.

### Reception using DMA (DMAREN=1)

1. Set the peripheral address (DMA\_CxPADDR = I2C\_RXDT address).
2. Set the memory address (DMA\_CxMADDR = memory address).
3. The transmission directions set from peripheral to memory (DTD = 0 in the DMA\_CHCTRL register).
4. Configure the total number of bytes to be transferred in the DMA\_CxDTCNT register.

5. Configure other parameters such as priority, memory data width, peripheral data width, interrupts, etc., in the DMA\_CHCTRL register.
6. Enable the DMA channel by setting CHEN = 1 in the DMA\_CxCTRL register.
7. Enable I<sup>2</sup>C DMA request by setting DMAEN = 1 in the I2C\_CTRL2 register. Once the RDBE bit in the I2C\_STS1 register is set, the data is loaded from the I2C\_DT register to the programmed memory through DMA.
8. When the number of data transfers, programmed in the DMA controller, is reached (DMA\_CxDTCNT = 0), the data transfer is complete (An interrupt is generated if enabled).
9. Master receiver: Clear the ACKFAIL flag, the STOP condition is generated, indicating that the transfer is complete (when the number of bytes to be transferred is greater  $\geq$  2 and DMAEND = 1, the NACK signal is generated automatically after transfer complete (DMA\_CxDTCNT = 0)).  
Slave receiver: Once the STOPF flag is set, clear the STOPF flag, and the transfer is complete.

## Error management

The error management feature included in the I<sup>2</sup>C provides a guarantee for the reliability of communication. Table 35 presents the manageable error events:

**Table 35. I<sup>2</sup>C error events**

| Error event      | Event flag | Enable control bit | Clear bit |
|------------------|------------|--------------------|-----------|
| SMBus alert      | ALERTF     | ERRIEN             | ALERTC    |
| Timeout error    | TMOUT      | ERRIEN             | TMOUTC    |
| PEC error        | PECERR     | ERRIEN             | PECERRC   |
| Overrun/underrun | OUF        | ERRIEN             | OUFCL     |
| Arbitration lost | ARLOST     | ERRIEN             | ARLOSTC   |
| Bus error        | BUSERR     | ERRIEN             | BUSERRC   |

### Overrun/Underrun (OUF)

In slave mode, an underrun/overrun may appear if the clock stretching feature is disabled (STRETCH = 1 in the I2C\_CTRL1 register).

In slave transmit mode: if data has not yet been written to the TXDT register before the transmission of the first bit of the to-be-transferred data (that is, before the generation of SDA edge), an underrun error may occur, and the OUF bit is set in the I2C\_STS register, sending 0xFF to the bus.

In slave receive mode: The slave must read the received data in the case of the clock stretching being disabled (STRETCH = 1). If one-byte data has been received and data is not read yet before the end of the next data reception, an overrun error occurs, setting the OUF bit in the I2C\_STS register, and sending NACK.

### Arbitration lost (ARLOST)

An arbitration lost may occur when the device controls the SDA line to output high level but the actual bus output is low.

- Master transmit: An arbitration may occur during an address transfer and a data transfer.
- Master receive: An arbitration may occur during an address transfer and an ACK response.
- Slave transmit: An arbitration may occur during a data transfer.
- Slave receive: An arbitration may occur during an ACK response.

Once an arbitration lost is detected, the ARLOST is set by hardware in the I2C\_STS register. The SCL and SDA buses will be released and go automatically back to slave mode.

### Bus error (BUSERR)

The SDA line, during a data transfer, must be kept in a stable state when the SCL is in high level. The SDA can be changed only when the SCL signal becomes low; otherwise, a bus error may appear. When the SCL is high:

- SDA changes from 1 to 0: a misplaced START condition
- SDA changes from 0 to 1: a misplaced STOP condition

Both of these conditions above may trigger a bus error. Once it occurs, the BUSERR is set by hardware in the I2C\_STS register.

### Packet error checking (PECERR)

The PEC is available only in SMBus mode. In master receive and slave receive modes, a PEC error may appear if the received PEC is not equal to the internally calculated PEC. In this case, the PECERR bit is set by hardware in the I2C\_STS register.

In slave receive mode, an NACK is sent when a PEC error is detected.

In master receive mode, an NACK is always sent, whatever the PEC check result.

### SMBus alert (ALERTF)

The SMBus alert feature is present when HADDREN = 1 (SMBus master mode) and SMBALERT = 1 (SMBus alert mode). Once an alert event is detected on the ALERT pin (ALERT pin changes from high to low), the ALERTF bit is set by hardware in the I2C\_STS register.

### Timeout error (TMOUT)

SMBus defines a timeout mechanism for the improvement of the system stability, preventing the bus from being pulled down in the case of a master or slave failure. Once a timeout event (defined in SMBus chapter) is detected, the TMOUT is set by hardware in the I2C\_STS register. If a timeout error occurs in slave mode, the SCL and SDA buses are immediately released; if a timeout error occurs in master mode, a STOP condition is automatically by host to abort the communication.

## I<sup>2</sup>C interrupt requests

The following table lists all the I<sup>2</sup>C interrupt requests.

**Table 36. I<sup>2</sup>C interrupt requests**

| Interrupt event                          | Event flag | Enable control bit |
|--|------------|--------------------|
| Address matched                          | ADDRF      | ADDRIEN            |
| Acknowledge failure                      | ACKFAIL    | ACKFAILIEN         |
| Stop condition received                  | STOPF      | STOPIEN            |
| Transmit interrupt state                 | TDIS       | TDIEN              |
| Receive data buffer full                 | RDBF       | RDIEEN             |
| Transfer complete, wait for loading data | TCRLD      | TDCIEN             |
| Data transfer complete                   | TDC        |                    |
| SMBus alert                              | ALERTF     | ERRIEN             |
| Timeout error                            | TMOUT      |                    |
| PEC error                                | PECERR     |                    |
| Overrun/Underrun                         | OUF        |                    |
| Arbitration lost                         | ARLOST     |                    |
| Bus error                                | BUSERR     |                    |

## I<sup>2</sup>C debug mode

When the microcontroller enters debug mode (Cortex®-M4 halted), the SMBUS timeout either continues to work or stops, depending on the I2Cx\_SMBUS\_TIMEOUT configuration bit in the DEBUG module.

## I<sup>2</sup>C registers

These peripheral registers must be accessed by words (32 bits).

**Table 37. I<sup>2</sup>C register map and reset values**

| Register    | Offset | Reset value |
|-------------|--------|-------------|
| I2C_CTRL1   | 0x00   | 0x00000000  |
| I2C_CTRL2   | 0x04   | 0x00000000  |
| I2C_OADDR1  | 0x08   | 0x00000000  |
| I2C_OADDR2  | 0x0C   | 0x00000000  |
| I2C_CLKCTRL | 0x10   | 0x00000000  |
| I2C_TIMEOUT | 0x14   | 0x00000000  |
| I2C_STS     | 0x18   | 0x00000000  |
| I2C_CLR     | 0x1C   | 0x00000000  |
| I2C_PEC     | 0x20   | 0x00000000  |
| I2C_RXDT    | 0x24   | 0x00000000  |
| I2C_TXDT    | 0x28   | 0x00000000  |

### Control register 1 (I2C\_CTRL1)

| Bit       | Name      | Reset value | Type | Description  |
|-----------|-----------|-------------|------|--|
| Bit 31:24 | Reserved  | 0x00        | res  | Kept at its default value.   |
| Bit 23    | PECEN     | 0x0         | rw   | PEC calculation enable<br>0: PEC calculation disabled<br>1: PEC calculation enabled  |
| Bit 22    | SMBALERT  | 0x0         | rw   | SMBus alert enable / pin set<br>To enable SMBus master alert feature:<br>0: SMBus alert disabled<br>1: SMBus alert enabled<br>To enable SMBus slave alert address:<br>0: Pin high<br>1: Pin low, response address 0001100x |
| Bit 21    | DEVADDREN | 0x0         | rw   | SMBus device default address enable<br>0: SMBus device default address disabled<br>1: SMBus device default address enabled, response device default address 1100001x   |
| Bit 20    | HADDREN   | 0x0         | rw   | SMBus host address enable<br>0: SMBus host address disabled<br>1: SMBus host address enabled, response host address 0001000x   |
| Bit 19    | GCAEN     | 0x0         | rw   | General call address enable<br>0: General call address disabled<br>1: General call address enabled, response 0000000x  |
| Bit 18    | Reserved  | 0x0         | res  | Kept at its default value.   |



| Bit       | Name       | Reset value | Type | Description  |
|-----------|------------|-------------|------|--|
| Bit 17    | STRETCH    | 0x0         | rw   | Clock stretching mode<br>0: Clock stretching mode enabled<br>1: Clock stretching mode disabled<br>Note: This feature applies to slave mode only.                 |
| Bit 16    | SCTRL      | 0x0         | rw   | Slave receive data control<br>0: Slave receive data disabled<br>1: Slave receive data enabled  |
| Bit 15    | DMAREN     | 0x0         | rw   | DMA receive data request enable<br>0: DMA receive data request disabled<br>1: DMA receive data request enabled   |
| Bit 14    | DMATEN     | 0x0         | rw   | DMA Transmit data request enable<br>0: DMA Transmit data request disabled<br>1: DMA Transmit data request enabled  |
| Bit 13:12 | Reserved   | 0x0         | resd | Kept at its default value.   |
| Bit 11:8  | DELT       | 0x0         | rw   | Digital filter value<br>$\text{Filter time} = \text{DFLT} \times T_{\text{I2C\_CLK}}$<br>The glitches less than the filter time on the SCL bus will be filtered. |
| Bit 7     | ERRIEN     | 0x0         | rw   | Error interrupt enable<br>0: Error interrupt disabled<br>1: Error interrupt enabled  |
| Bit 6     | TDCIEN     | 0x0         | rw   | Data transfer complete interrupt enable<br>0: Data transfer complete interrupt disabled<br>1: Data transfer complete interrupt enabled                           |
| Bit 5     | STOPIEN    | 0x0         | rw   | Stop generation complete interrupt enable<br>0: Stop generation complete interrupt disabled<br>1: Stop generation complete interrupt enabled                     |
| Bit 4     | ACKFAILIEN | 0x0         | rw   | Acknowledge fail interrupt enable<br>0: Acknowledge fail interrupt disabled<br>1: Acknowledge fail interrupt enabled   |
| Bit 3     | ADDRIEN    | 0x0         | rw   | Address match interrupt enable<br>0: Address match interrupt disabled<br>1: Address match interrupt enabled  |
| Bit 2     | RDIEN      | 0x0         | resd | Data receive interrupt enable<br>0: Data receive interrupt disabled<br>1: Data receive interrupt enabled   |
| Bit 1     | TDIEN      | 0x0         | rw   | Data transmit interrupt enable<br>0: Data transmit interrupt disabled<br>1: Data transmit interrupt enabled  |
| Bit 0     | I2CEN      | 0x0         | rw   | I <sup>2</sup> C peripheral enable<br>0: Disabled<br>1: Enabled  |

### Control register 2 (I2C\_CTRL2)

| Bit       | Name       | Reset value | Type | Description   |
|-----------|------------|-------------|------|---|
| Bit 31:27 | Reserved   | 0x00        | res  | Kept at its default value.  |
| Bit 26    | PECTEN     | 0x0         | rw   | Request PEC transmission enable<br>0: Transmission disabled<br>1: Transmission enabled  |
| Bit 25    | ASTOPEN    | 0x0         | rw   | Automatically send stop condition enable<br>0: Disabled (Software sends STOP condition)<br>1: Enabled (Automatically send STOP condition) |
| Bit 24    | RLDEN      | 0x0         | rw   | Send data reload mode enable<br>0: Send data reload mode disable<br>1: Send data reload mode enabled                                      |
| Bit 23:16 | CNT[7:0]   | 0x00        | rw   | Transmit data counter   |
| Bit 15    | NACKEN     | 0x0         | rw   | Not acknowledge enable<br>0: Acknowledge enabled<br>1: Acknowledge disabled   |
| Bit 14    | GENSTOP    | 0x0         | rw   | Generate stop condition<br>0: No stop generation<br>1: stop generation  |
| Bit 13    | GENSTART   | 0x0         | rw   | Generate start condition<br>0: No start generation<br>1: Start generation   |
| Bit 12    | READH10    | 0x0         | rw   | 10-bit address header read enable<br>0: 10-bit address header read disabled<br>1: 10-bit address header read enabled                      |
| Bit 11    | ADDR10     | 0x0         | rw   | Host sends 10-bit address mode enable<br>0: 7-bit address mode<br>1: 10-bit address mode  |
| Bit 10    | DIR        | 0x0         | rw   | Master data transfer direction<br>0: Receive<br>1: Transmit   |
| Bit 9:0   | SADDR[9:0] | 0x000       | rw   | Slave address sent by the master<br>In 7-bit address mode, BIT0 and BIT[9:8] don't care.  |

### Own address register 1 (I2C\_OADDR1)

| Bit       | Name       | Reset value | Type | Description   |
|-----------|------------|-------------|------|---|
| Bit 31:16 | Reserved   | 0x0000      | rw   | Kept at its default value.  |
| Bit 15    | ADDR1EN    | 0x0         | rw   | Own Address 1 enable<br>0: Own Address 1 disabled<br>1: Own Address 1 enabled |
| Bit 14:11 | Reserved   | 0x0         | res  | Kept at its default value.  |
| Bit 10    | ADDR1MODE  | 0x0         | rw   | Own Address mode<br>0: 7-bit address mode<br>1: 10-bit address mode           |
| Bit 9:0   | ADDR1[9:0] | 0x000       | rw   | Own address 1<br>In 7-bit address mode, bit 0 and bit [9:8] don't care.       |

### Own address register 2 (I2C\_OADDR2)

| Bit       | Name           | Reset value | Type | Description  |
|-----------|----------------|-------------|------|--|
| Bit 31:16 | Reserved       | 0x0000      | res  | Kept at its default value.   |
| Bit 15    | ADDR2EN        | 0x0         | rw   | Own address 2 enable<br>0: Own address 2 disabled<br>1: Own address 2 enabled  |
| Bit 14:11 | Reserved       | 0x0         | res  | Kept at its default value  |
| Bit 10:8  | ADDR2MASK[2:0] | 0x0         | rw   | Own address 2-bit mask<br>000: Match Address bit [7:1]<br>001: Match Address bit [7:2]<br>010: Match address bit [7:3]<br>011: Match address bit [7:4]<br>100: Match address bit [7:5]<br>101: Match address bit [7:6]<br>110: Match address bit [7]<br>111: Response all addresses other than those reserved for I <sup>2</sup> C |
| Bit 7:1   | ADDR2[7:1]     | 0x00        | rw   | Own address 2<br>7-bit address mode  |
| Bit 0     | Reserved       | 0x0         | res  | Kept at its default value.   |

### Timing register (I2C\_CLKCTRL)

| Bit       | Name      | Reset value | Type | Description  |
|-----------|-----------|-------------|------|--|
| Bit 31:28 | DIVL[3:0] | 0x0         | rw   | Low 4 bits of clock divider value  |
| Bit 27:24 | DIVH[7:4] | 0x0         | rw   | High 4 bits of clock divider value<br>$DIV = (DIVH \ll 4) + DIVL$                |
| Bit 23:20 | SCLD[3:0] | 0x0         | rw   | SCL output delay<br>$T_{SCLD} = (SCLD + 1) \times (DIV + 1) \times T_{I2C\_CLK}$ |
| Bit 19:16 | SDAD[3:0] | 0x0         | rw   | SDA output delay<br>$T_{SDAD} = (SDAD + 1) \times (DIV + 1) \times T_{I2C\_CLK}$ |
| Bit 15:8  | SCLH[7:0] | 0x00        | rw   | SCL high level<br>$T_{SCLH} = (SCLH + 1) \times (DIV + 1) \times T_{I2C\_CLK}$   |
| Bit 7:0   | SCLL[7:0] | 0x00        | rw   | SCL low level<br>$T_{SCLL} = (SCLL + 1) \times (DIV + 1) \times T_{I2C\_CLK}$    |

### Timeout register (I2C\_TIMEOUT)

| Bit        | Name          | Reset value | Type | Description  |
|------------|---------------|-------------|------|--|
| Bit 31     | EXTEN         | 0x0         | rw   | Cumulative clock low extend timeout enable<br>0: Cumulative clock low extend timeout disabled<br>1: Cumulative clock low extend timeout enabled<br>Corresponds to $T_{LOW:SEXT} / T_{LOW:MEXT}$ in SMBus |
| Bit 30: 28 | Reserved      | 0x0         | res  | Kept at its default value.   |
| Bit 27: 16 | EXTTIME[11:0] | 0x000       | rw   | Cumulative clock low extend timeout value<br>Timeout duration = $(EXTTIME + 1) \times 2048 \times T_{I2C\_CLK}$  |
| Bit 15     | TOEN          | 0x0         | rw   | Detect clock low/high timeout enable<br>0: Clock low/high timeout detection disabled<br>1: clock low/high timeout detection enabled<br>Corresponds to $T_{TIMEOUT}$ in SMBus.                            |
| Bit 14: 13 | Reserved      | 0x0         | res  | Kept at its default value.   |

| Bit      | Name         | Reset value | Type | Description  |
|----------|--------------|-------------|------|--|
| Bit 12   | TOMODE       | 0x0         | rw   | Clock timeout detection mode<br>0: Clock low level detection<br>1: Clock high level detection  |
| Bit 11:0 | TOTIME[11:0] | 0x000       | rw   | Clock timeout detection time<br>For clock low level detection (TOMODE = 0):<br>Timeout duration = (TOTIME + 1) × 2048 × T <sub>I2C_CLK</sub><br>For clock high level detection (TOMODE = 1):<br>Timeout duration = (TOTIME + 1) × 4 × T <sub>I2C_CLK</sub> |

### Status register (I2C\_STS)

| Bit       | Name      | Reset value | Type | Description  |
|-----------|-----------|-------------|------|--|
| Bit 31:24 | Reserved  | 0x00        | res  | Kept at its default value.   |
| Bit 23:17 | ADDR[6:0] | 0x00        | r    | Slave address matching value<br>In 7-bit address mode: Slave address received<br>In 10-bit address mode: 10-bit slave address header received  |
| Bit 16    | SDIR      | 0x0         | r    | Slave data transfer direction<br>0: Receive data<br>1: Transmit data   |
| Bit 15    | BUSYF     | 0x0         | r    | Bus busy flag transmission mode<br>0: Bus idle<br>1: Bus busy<br>Once a START condition is detected, this bit is set;<br>Once a STOP condition is detected, this bit is automatically cleared. |
| Bit 14    | Reserved  | 0x00        | res  | Kept at its default value.   |
| Bit 13    | ALERTF    | 0x0         | r    | SMBus alert flag<br>SMBus host: This bit indicates the reception of an alert signal (ALERT pin changes from high to low)<br>0: No alert signal received<br>1: Alert signal received            |
| Bit 12    | TMOUT     | 0x0         | r    | SMBus timeout flag<br>0: No timeout<br>1: Timeout  |
| Bit 11    | PECERR    | 0x0         | r    | PEC receive error flag<br>0: No PEC error<br>1: PEC error  |
| Bit 10    | OUF       | 0x0         | r    | Overflow or underflow flag<br>In transmission mode:<br>0: No overflow or underflow<br>1: Underflow<br>In reception mode:<br>0: No overflow or underflow<br>1: Overflow                         |
| Bit 9     | ARLOST    | 0x0         | r    | Arbitration lost flag<br>0: No arbitration lost detected.<br>1: Arbitration lost detected.   |
| Bit 8     | BUSERR    | 0x0         | rw0c | Bus error flag<br>0: No Bus error occurred<br>1: Bus error occurred  |

| Bit   | Name     | Reset value | Type | Description  |
|-------|----------|-------------|------|--|
| Bit 7 | TCRLD    | 0x0         | r    | <p>Data transfer complete, waiting for data load</p> <p>0: Data transfer is not complete yet</p> <p>1: Data transfer is complete</p> <p>This bit is set when data transfer is complete (CNT = 1) and reload mode is enabled (RLDEN = 1). It is automatically cleared when writing a CNT value.</p> <p>This bit is applicable in master mode or when SCTRL = 1 in slave mode.</p>                               |
| Bit 6 | TDC      | 0x0         | r    | <p>Data transfer complete flag</p> <p>0: Data transfer is not completed yet (the shift register still holds data)</p> <p>1: Data transfer is completed (shift register become empty and all data has been sent to the bus)</p> <p>This bit is set when ASTOPEN = 0, RLDEN = 0, CNT = 0.</p> <p>It is automatically cleared after a START or a STOP condition is received.</p>                                  |
| Bit 5 | STOPF    | 0x0         | r    | <p>Stop condition generation complete flag</p> <p>0: No Stop condition detected.</p> <p>1: Stop condition detected.</p>  |
| Bit 4 | ACKFAILF | 0x0         | r    | <p>Acknowledge failure flag</p> <p>0: No acknowledge failure</p> <p>1: Acknowledge failure</p>   |
| Bit 3 | ADDRHF   | 0x0         | r    | <p>0 ~ 7 bit address head match flag</p> <p>0: 0 ~ 7 bit address head mismatch</p> <p>1: 0 ~ 7 bit address head match</p>  |
| Bit 2 | RDBF     | 0x0         | r    | <p>Receive data buffer full flag</p> <p>0: Data register has not received data yet</p> <p>1: Data register has received data</p>   |
| Bit 1 | TDIS     | 0x0         | rw1s | <p>Transmit data interrupt status</p> <p>0: Data has been written to the I2C_TXDT</p> <p>1: Data has been sent from the I2C_TXDT to the shift register. I2C_TXDT become empty, and thus the to-be-transferred data must be written to the I2C_TXDT.</p> <p>When the clock stretching mode is disabled, a TDIS event is generated by writing 1 so that data is written to the I2C_TXDT register in advance.</p> |
| Bit 0 | TDBE     | 0x0         | rw1s | <p>Transmit data buffer empty flag</p> <p>0: I2C_TXDT not empty</p> <p>1: I2C_TXDT empty</p> <p>This bit is only used to indicate the current status of the I2C_TXDT register. The I2C_TXDT register can be cleared by writing 1 through software.</p>   |

### Status clear register (I2C\_CLR)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:14 | Reserved | 0x00000     | res  | Kept at its default value.  |
| Bit 13    | ALERTC   | 0x0         | w    | Clear SMBus alert flag<br>SMBus alert flag is cleared by writing 1.   |
| Bit 12    | TMOUTC   | 0x0         | w    | Clear SMBus timeout flag<br>SMBus timeout flag is cleared by writing 1.   |
| Bit 11    | PECERRC  | 0x0         | w    | Clear PEC receive error flag<br>PEC receive error flag is cleared by writing 1.                                   |
| Bit 10    | OUFC     | 0x0         | w    | Clear overload / underload flag<br>Overload / underload flag is cleared by writing 1.                             |
| Bit 9     | ARLOSTC  | 0x0         | w    | Clear arbitration lost flag<br>Arbitration lost flag is cleared by writing 1.                                     |
| Bit 8     | BUSERRC  | 0x0         | w    | Clear bus error flag<br>Bus error flag is cleared by writing 1  |
| Bit 7:6   | Reserved | 0x0         | res  | Kept at its default value.  |
| Bit 5     | STOPC    | 0x0         | w    | Clear stop condition generation complete flag<br>Stop condition generation complete flag is cleared by writing 1. |
| Bit 4     | ACKFAILC | 0x0         | w    | Clear acknowledge failure flag<br>Acknowledge failure flag is cleared by writing 1.                               |
| Bit 3     | ADDRC    | 0x0         | w    | Clear 0 ~ 7 bit address match flag<br>0 ~ 7 bit address match flag is cleared by writing 1.                       |
| Bit 2:0   | Reserved | 0x0         | res  | Kept at its default value.  |

### PEC register (I2C\_PEC)

| Bit      | Name        | Reset value | Type | Description                |
|----------|-------------|-------------|------|----------------------------|
| Bit 31:8 | Reserved    | 0x000000    | res  | Kept at its default value. |
| Bit 7:0  | PECVAL[7:0] | 0x00        | r    | PEC value                  |

### Receive data register (I2C\_RXDT)

| Bit      | Name     | Reset value | Type | Description                |
|----------|----------|-------------|------|----------------------------|
| Bit 31:8 | Reserved | 0x000000    | res  | Kept at its default value. |
| Bit 7:0  | DT[7:0]  | 0x00        | r    | Receive data register      |

### Transmit data register (I2C\_TXDT)

| Bit      | Name     | Reset value | Type | Description                |
|----------|----------|-------------|------|----------------------------|
| Bit 31:8 | Reserved | 0x000000    | res  | Kept at its default value. |
| Bit 7:0  | DT[7:0]  | 0x00        | rw   | Transmit data register     |

# 12 Universal synchronous/asynchronous receiver/transmitter (USART)

## USART introduction

The universal synchronous/asynchronous receiver/transmitter (USART) serves an interface for communication by means of various configurations and peripherals with different data formats. It supports asynchronous full-duplex and half-duplex as well as synchronous transfer. With a programmable baud rate generator, users can configure the required communication frequency by setting the system frequency and frequency divider.

In addition to standard NRZ asynchronous and synchronous receiver/transmitter communication protocols, USART also supports widely-used serial communication protocols such as LIN (Local Interconnection Network), IrDA (Infrared Data Association) SIRENDEC specification, Asynchronous SmartCard protocol defined in ISO7816-3 standard, and CTS/RTS (Clear To Send/Request To Send) hardware flow operation.

It also allows multi-processor communication, and supports silent mode waken up by idle frames or ID matching to build up a USART network. Meanwhile, high-speed communication is possible by using DMA.

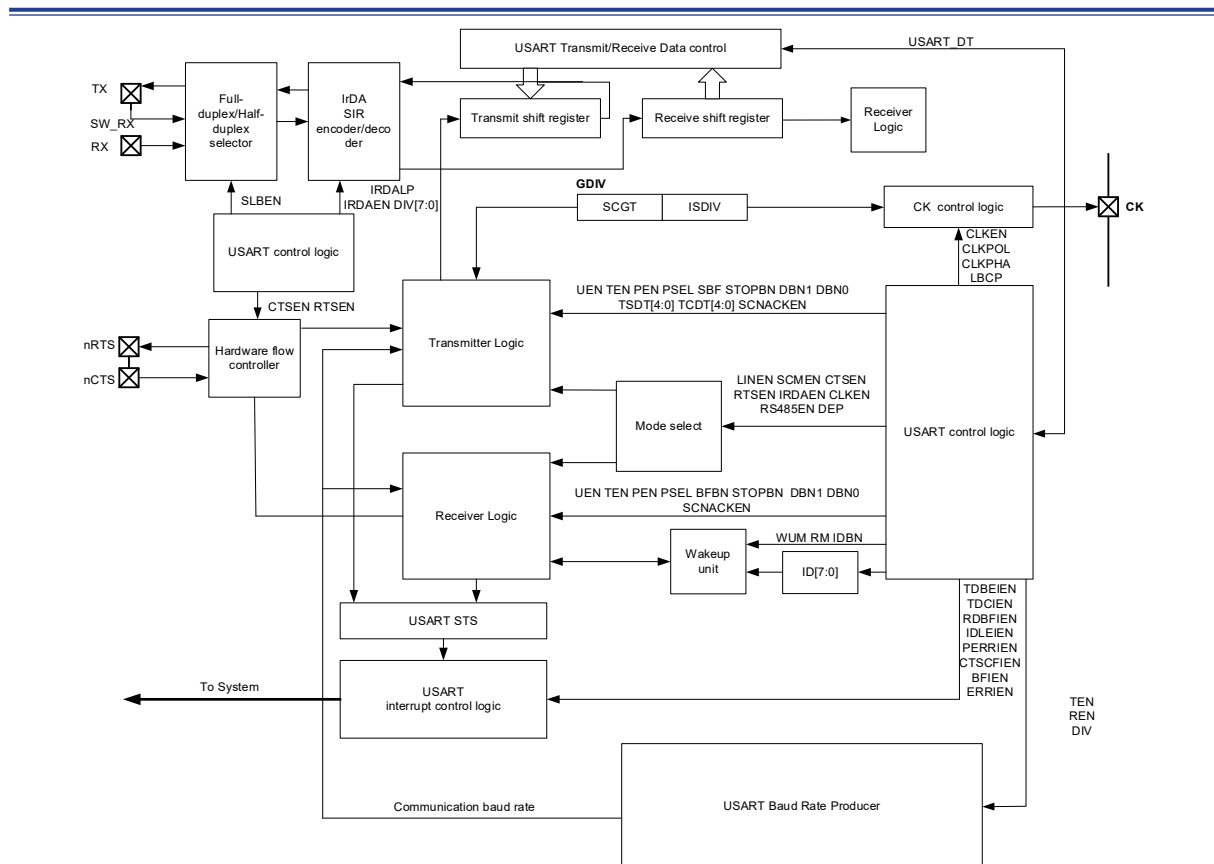


Figure 49. USART block diagram

### USART main features:

- Programmable full-duplex or half-duplex communication
  - Full-duplex, asynchronous communication
  - Half-duplex, single communication
- Programmable communication modes
  - NRZ standard format (Mark/Space)
  - LIN (Local Interconnection Network):
  - IrDA SIR:
  - Asynchronous SmartCard protocol defined in ISO7816-3 standard: Support 0.5 or 1.5 stop bits in Smartcard mode
  - RS-232 CTS/RTS (Clear To Send/Request To Send) hardware flow operation
  - RS-485
  - Multi-processor communication with silent mode (waken up by configuring ID match and bus idle frame)
  - Synchronous mode
- Programmable baud rate generator
  - Shared by transmission and reception
- Programmable frame format
  - Programmable data word length (7 bits, 8 bits or 9 bits)
  - Programmable stop bits-support 1 or 2 stop bits
  - Programmable parity control: transmitter with parity bit transmission capability, and receiver with received data parity check capability
- Programmable DMA multi-processor communication
- Programmable separate enable bits for transmitter and receiver
- Programmable output CLK phase, polarity and frequency
- Detection flags
  - Receive buffer full
  - Transmit buffer empty
  - Transfer complete flag
- Four error detection flags
  - Overrun error
  - Noise error
  - Framing error
  - Parity error
- Programmable 10 interrupt sources with flags
  - CTSF changes
  - LIN break detection



- Transmit data register empty
- Transmission complete
- Receive data register full
- Idle bus detected
- Overrun error
- Framing error
- Noise error
- Parity error

## Full-duplex/half-duplex selector

The full-duplex and half-duplex selector enables USART to perform data exchanges with peripherals in full-duplex or half-duplex mode, which is achieved by setting the corresponding registers.

In two-wire unidirectional full-duplex mode (by default), TX pin is used for data output, while the RX pin is used for data input. Since the transmitter and receiver are independent of each other, USART is allowed to send/receive data at the same time so as to achieve full-duplex communication.

When the HALFSEL is set 1, the single-wire bidirectional half-duplex mode is selected for communication. In this case, the LINEN, CLKEN, SCMEN and IRDAEN bits must be set 0. RX pin is inactive, while TX and SW\_RX are interconnected inside the USART. For the USART part, TX pins is used for data output, and SW\_RX for data input. For the peripheral part, bidirectional data transfer is executed through IO mapped by TX pin.

## Mode selector

### Introduction

USART mode selector allows USART to work in different operation modes through software configuration so as to enable data exchanges between USART and peripherals with different communication protocols.

USART supports NRZ standard format (Mark/Space), by default. It also supports LIN (Local Interconnection Network), IrDA SIR (Serial Infrared), Asynchronous Smartcard protocol in ISO7816-3 standard, RS-232 CTS/RTS (Clear To Send/Request To Send) hardware flow operation, silent mode and synchronous mode, depending on USART mode selection configuration.

### Configuration procedure

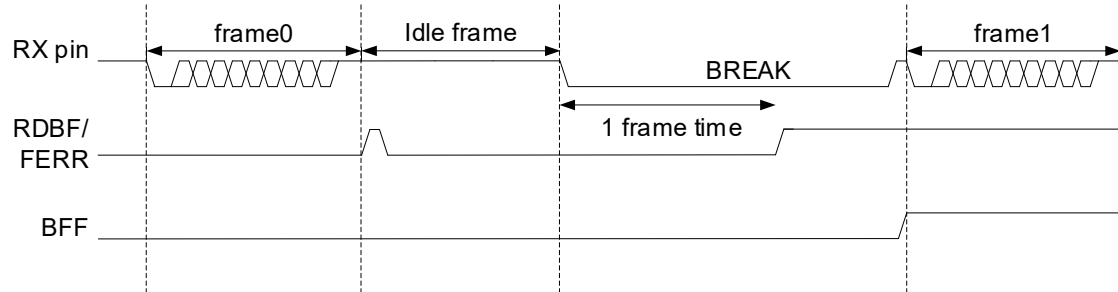
Selection of operation mode is done by following the configuration process listed below. In addition, such configuration method, along with those of receiver and transmitter described in the subsequent sections, are used to make USART initialization configuration.

#### LIN mode:

Parameters configuration: LINEN = 1, CLKEN = 0, STOPBN[1:0] = 0, SCMEN = 0, SLBEN = 0, IRDAEN = 0 and DBN[1:0] = 00. LIN master has break frame transmission capability, and thus it is able to send 13-bit low-level LIN synchronous break frame by setting SBF = 1.

Similarly, LIN slave has break frame detection capability, and thus it is able to detect 11-bit or 10-bit break frame, depending on whether BFBN = 1 or BFBN = 0.

#### CASE1 : BREAK frame occurring after idle frame



#### CASE2 : BREAK frame occurring while a frame is being received

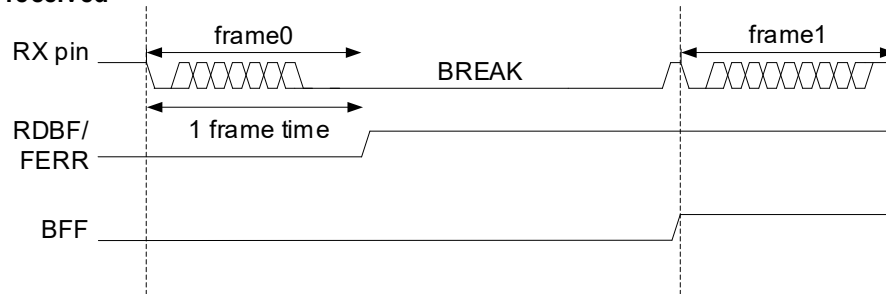


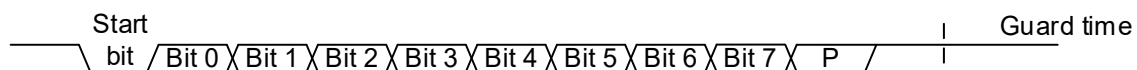
Figure 50. BFF and FERR detection in LIN mode

#### Smartcard mode:

Parameters configuration: SCMEN = 1, LINEN = 0, SLBEN = 0, IRDAEN = 0, CLKEN = 1, DBN = 1, PEN = 1, and STOPBN[1:0] = 11. The polarity, phase and pulse number of the clock can be configured by setting the CLKPOL, CLKPHA and LBCP bits (Refer to Synchronous mode for details). The assertion of the TDC flag can be delayed by setting the SCGT[7:0] bit (guard time bit). The TDF bit can be asserted high after the guard time counter reaches the value programmed in the SCGT[7:0] bit. The Smartcard is a single-wire half duplex communication protocol. The SCNACKEN bit is used to select whether to send NACK when a parity error occurs. This is to indicate to the Smarcard that the data has not been correctly received

Note: It is recommended to use 1.5 stop bits for both transmitting and receiving. The guard time for smart card must meet 1.5-bit time to receive back-to-back data.

#### Without Parity error :



#### With Parity error :

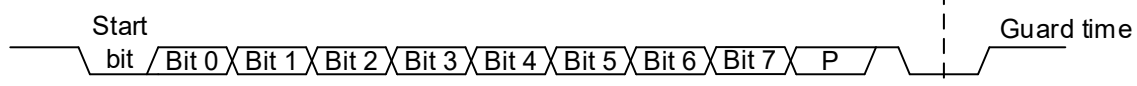
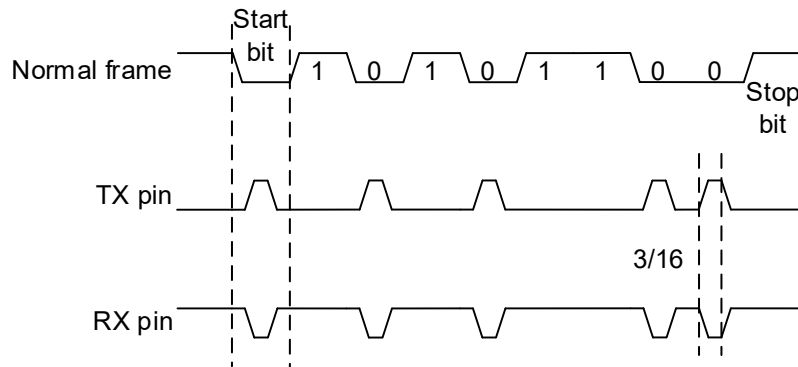


Figure 51. Smartcard frame format

### Infrared mode:

Parameters configuration: IRDAEN = 1, CLKEN = 0, STOPBN[1:0] = 0, SCMEN = 0 and SLBEN = 0. The infrared low-power mode can be enabled by setting IRDALP = 1. In normal mode the transmitted pulse width is specified as 3/16 bit. In infrared low-power mode, the pulse width can be configurable. And the ISDIV[7:0] bit can be used to achieve the desired low-power frequency.



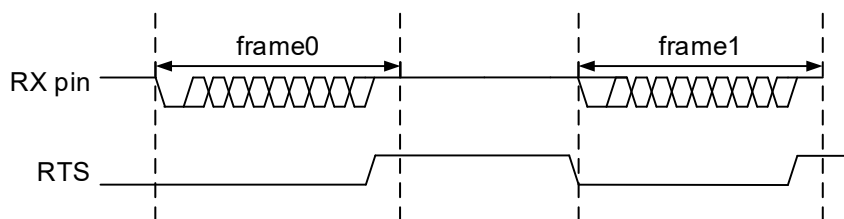
**Figure 52. IrDA DATA(3/16) – normal mode**

### Hardware flow control mode:

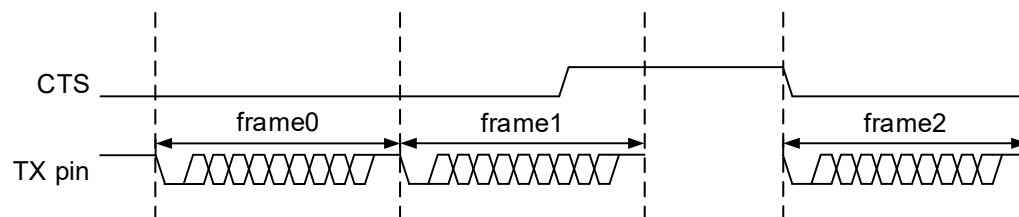
RTS and CTS flow control can be enabled by setting RTSEN = 1 and CTSEN = 1, respectively. RTS: the RTS becomes active (pull-down means low) as soon as the USART receiver is ready to receive a data. When the data has arrived (starts at each STOP bit) in the receive register, the RTS bit is set, indicating request to stop data transfer at the end of current frame.

CTS: the USART transmitter checks the CTS input before sending next frame. The next data is sent if CTS is active (when low); if CTS becomes inactive (when high) during transmission, it stops sending at the end of current transfer.

#### RTS follow control :



#### CTS follow control :



**Figure 53. Hardware flow control**

### RS485 mode:

This mode is enabled by setting RS485EN = 1. The enable signal is output on the RTS pin. The DEP bit is used to select the polarity of the DE signal. The TSDT[4:0] bit is used to define the latency before the transmission of the start bit on the transmitter side, while the TCDT[4:0] is used to define the latency before the TC flag is set following the stop bit at the end of the last data.

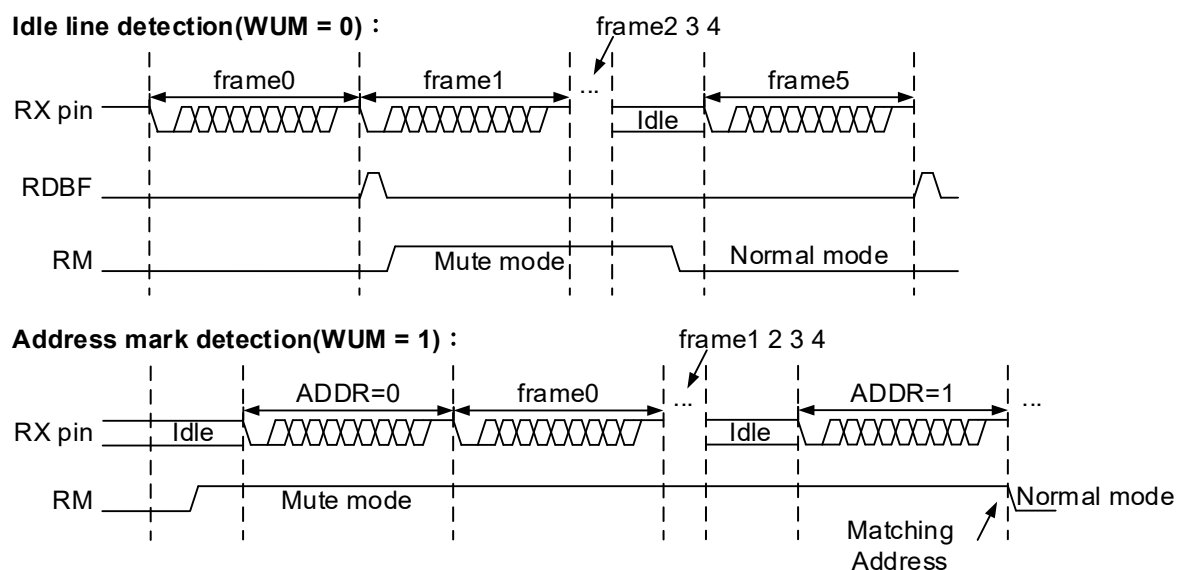
### Silent mode:

Silent mode is entered by setting RM = 1. It is possible to wake up from silent mode by setting WUM = 1 (ID match) and WUM = 0 (idle bus), respectively. The ID[7:0] is configurable. Select ID[7:0] or ID[3:0] by setting the IDBN bit. When ID match is selected, if the MSB of data bit is set, it indicates that the current data stands for ID.

When parity check is disabled, if DBN[1:0] = 10, the MSB is USART\_DT[6]; if DBN[1:0] = 00, the MSB is USART\_DT[7]; if DBN[1:0] = 01, the MSB stands for USART\_DT[8].

When parity check is enabled, if DBN[1:0] = 10, the MSB stands for USART\_DT[5]; if DBN[1:0] = 00, the MSB stands for USART\_DT[6]; if DBN[1:0] = 01, the MSB stands for USART\_DT[7].

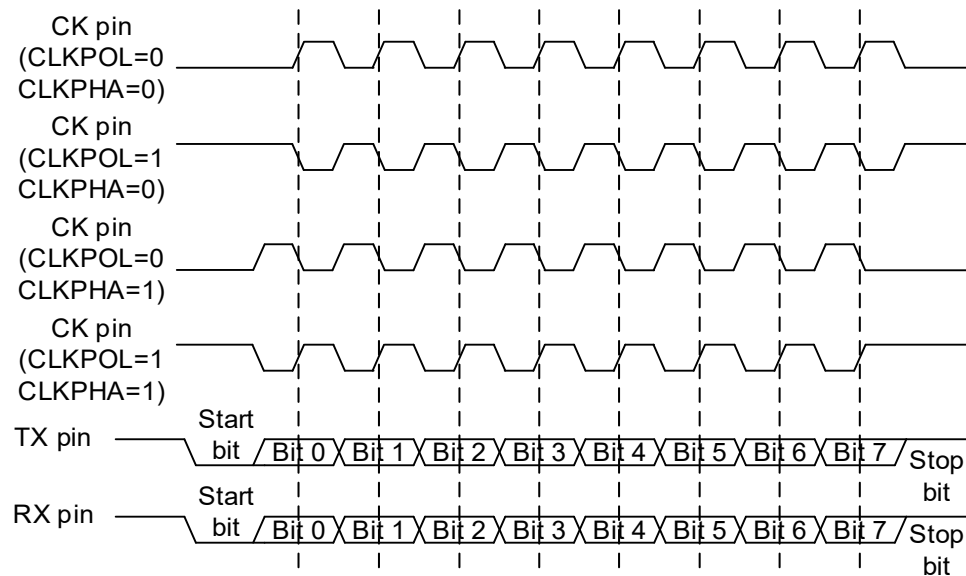
When the ID[3:0] bit is selected, the four LSB bits indicate the ID value; When the ID[7:0] bit is selected, all of the LSB bits indicates the ID value, except for the above parity check bits and MSB bits.



**Figure 54. Mute mode using Idle line or Address mark detection**

### Synchronous mode:

Set the CLKEN bit enables synchronous mode and clock pin output. Select CK pin high or low in idle state by setting the CLKPOL bit (1 or 0). Whether to sample data on the second or first edge of the clock depends on the CLKPHA bit (1 or 0). The LBCP bit (1 or 0) is used to select whether to output clock on the last data bit. And the ISDIV[4:0] is used to select the required clock output frequency.



**Figure 55. 8-bit format USART synchronous mode**

## USART frame format and configuration

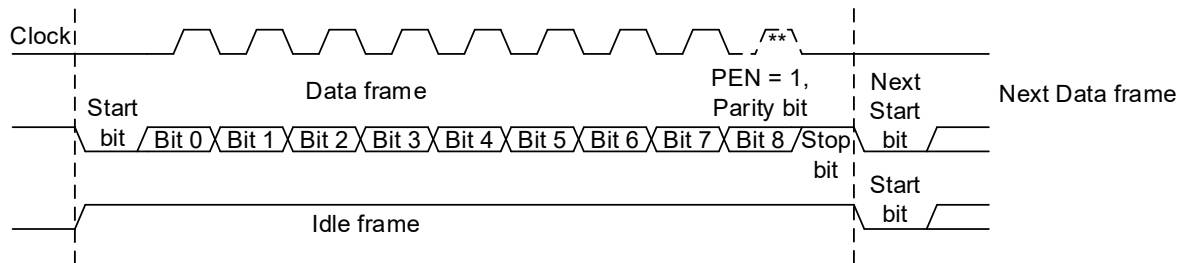
USART data frame consists of start bit, data bit and stop bit, with the last data bit being as a parity bit.

USART idle frame size is equal to that of the data frame under current configuration, but all bits are 1.

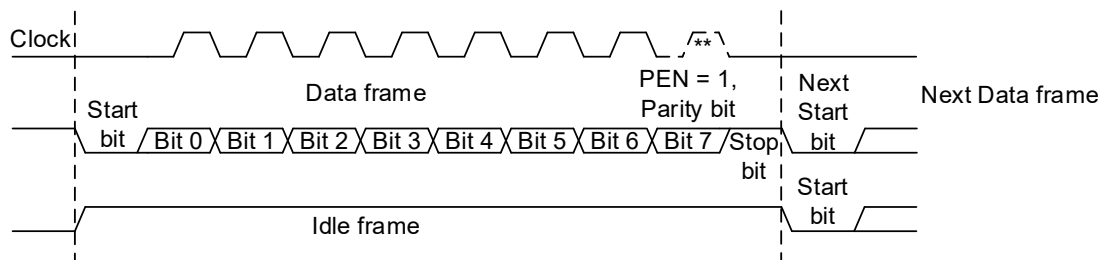
USART break frame size is the current data frame size plus its stop bit. All bits before the stop bit are 0. In non-LIN mode, a break frame transmission and detection must be in line with this rule. For instance, if  $DBN[1:0] = 00$ , the break frame size for transmission and detection should be 10-bit low level plus its stop bit. In LIN mode, refer to Mode selector and configuration process for more details.

The  $DBN1$  and  $DBN0$  bits are used to program 7-bit ( $DBN[1:0] = 10$ ), 8-bit ( $DBN[1:0] = 00$ ) or 9-bit ( $DBN[1:0] = 01$ ) data bits.

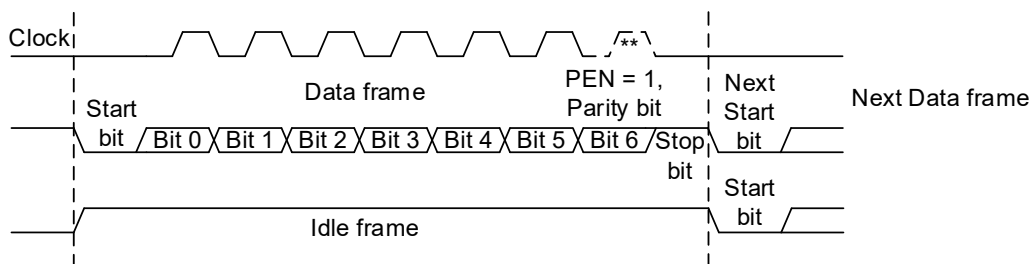
**9-bit word length (DBN1, DBN0 = 01) :**



**8-bit word length (DBN1, DBN0 = 00) :**



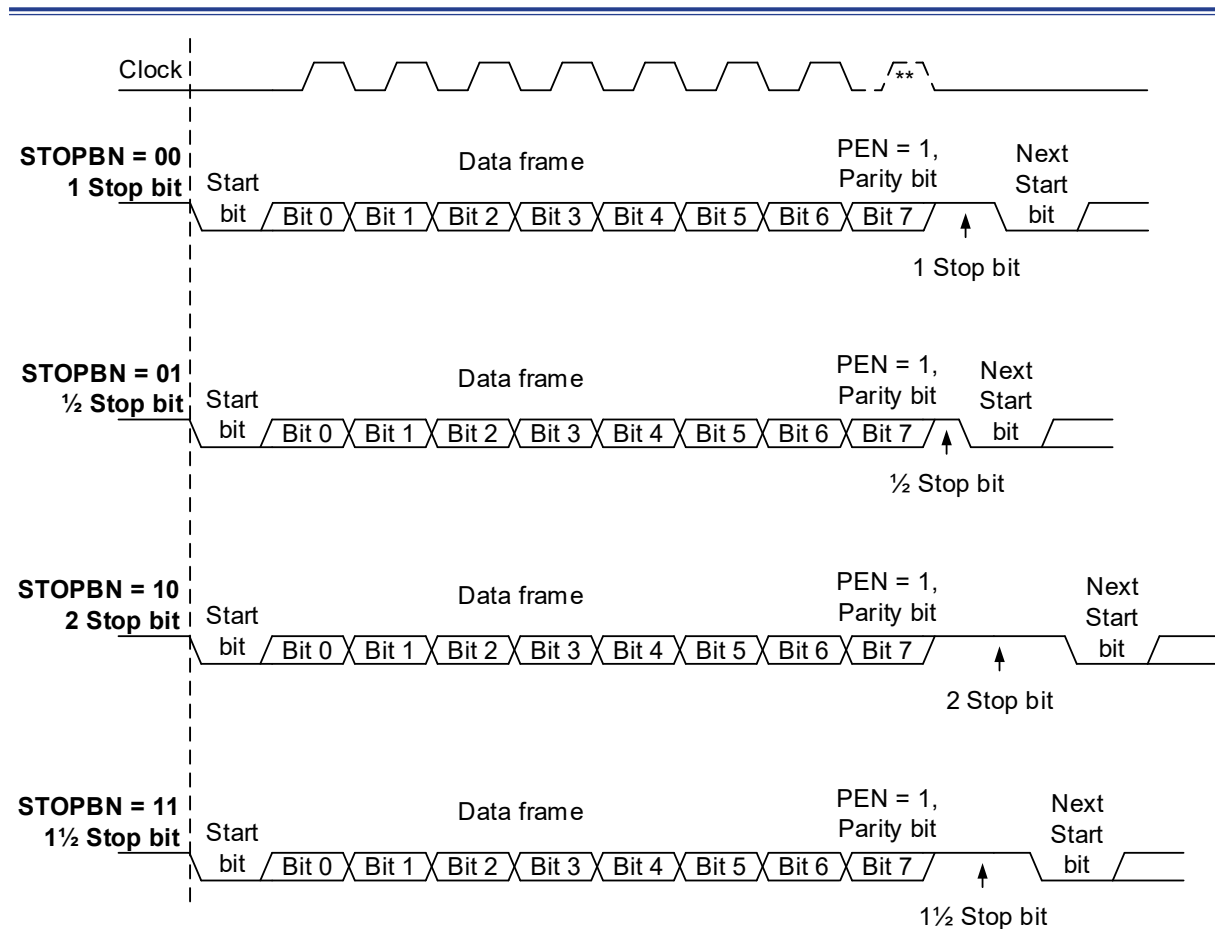
**7-bit word length (DBN1, DBN0 = 10) :**



**Figure 56. Word length**

The STOPBN bit is used to program one bit (STOPBN = 00), 0.5-bit (STOPBN = 01), two-bit (STOPBN = 10) and 1.5-bit (STOPBN = 11) stop bits.

Set the PEN bit will enable parity control. PSEL = 1 indicates Odd parity, while PSEL = 0 for Even parity. Once the parity control is enabled, the MSB of the data bit will be replaced with parity bit, that is, the valid data bits are reduced by one bit.



**Figure 57. Stop bit configuration**

## DMA transfer introduction

Enable transmit data buffer and receive data buffer using DMA to achieve continuous high-speed transmission for USART, which is detailed in subsequent sections. For more information on specific DMA configuration, refer to DMA chapter.

### Transmission using DMA

1. Select a DMA channel: Select a DMA channel from DMA channel map table described in DMA chapter.
2. Configure the destination of DMA transfer: Configure the USART\_DT register address as the destination address bit of DMA transfer in the DMA control register. Data will be sent to this address after transmit request is received by DMA.
3. Configure the source of DMA transfer: Configure the memory address as the source of DMA transfer in the DMA control register. Data will be loaded into the USART\_DT register from the memory address after transmit request is received by DMA.
4. Configure the total number of bytes to be transferred in the DMA control register.
5. Configure the channel priority of DMA transfer in the DMA control register.

6. Configure DMA interrupt generation after half or full transfer in the DMA control register.
7. Enable DMA transfer channel in the DMA control register.

## Reception using DMA

1. Select a DMA transfer channel: Select a DMA channel from DMA channel map table described in DMA chapter.
2. Configure the destination of DMA transfer: Configure the memory address as the destination of DMA transfer in the DMA control register. Data will be loaded from the USART\_DT register to the programmed destination after reception request is received by DMA.
3. Configure the source of DMA transfer: Configure the USART\_DT register address as the source of DMA transfer in the DMA control register. Data will be loaded from the USART\_DT register to the programmed destination after reception request is received by DMA.
4. Configure the total number of bytes to be transferred in the DMA control register.
5. Configure the channel priority of DMA transfer in the DMA control register.
6. Configure DMA interrupt generation after half or full transfer in the DMA control register.
7. Enable a DMA transfer channel in the DMA control register.

## Baud rate generation

### Introduction

USART baud rate generator uses an internal counter based on PCLK. The DIV (USART\_BAUDR [15:0] register) represents the overflow value of the counter. Each time the counter is full, it denotes one-bit data. Thus each data bit width refers to PCLK cycles x DIV.

The receiver and transmitter of USART share the same baud rate generator, and the receiver splits each data bit into 16 equal parts to achieve oversampling, so the data bit width should not be less than 16 PCLK periods, that is, the DIV value must be equal to or be greater than 16.

### Configuration

User can program the desired baud rate by setting different system clocks and writing different values into the USART\_BAUDR register. The calculation format is as follows:

$$\frac{TX}{RX} \text{ baud rate} = \frac{f_{CK}}{DIV}$$

Where,  $f_{CK}$  refers to the system clock of USART (i.e. PCLK1/PCLK2)

- Note: 1. Write access to the USART\_BAUDR register before UEN. The baud rate register value should not be altered when UEN = 1.
2. When USART receiver or transmitter is disabled, the internal counter will be reset, and baud rate interrupt will occur.

**Table 38. Error calculation for programmed baud rate**

| Baud |      | $f_{PCLK} = 36 \text{ MHz}$ |                                       |         | $f_{PCLK} = 72 \text{ MHz}$ |                                       |         |
|------|------|-----------------------------|---------------------------------------|---------|-----------------------------|---------------------------------------|---------|
| No.  | Kbps | Actual                      | Value programmed in the baud register | Error % | Actual                      | Value programmed in the baud register | Error % |
| 1    | 2.4  | 2.4                         | 15000                                 | 0 %     | 2.4                         | 30000                                 | 0 %     |
| 2    | 9.6  | 9.6                         | 3750                                  | 0 %     | 9.6                         | 7500                                  | 0 %     |
| 3    | 19.2 | 19.2                        | 1875                                  | 0 %     | 19.2                        | 3750                                  | 0 %     |



| Baud |       | f <sub>PCLK</sub> = 36 MHz |                                       |         | f <sub>PCLK</sub> = 72 MHz |                                       |         |
|------|-------|----------------------------|---------------------------------------|---------|----------------------------|---------------------------------------|---------|
| No.  | Kbps  | Actual                     | Value programmed in the baud register | Error % | Actual                     | Value programmed in the baud register | Error % |
| 4    | 57.6  | 57.6                       | 625                                   | 0 %     | 57.6                       | 1250                                  | 0 %     |
| 5    | 115.2 | 115.384                    | 312                                   | 0.15 %  | 115.2                      | 625                                   | 0 %     |
| 6    | 230.4 | 230.769                    | 156                                   | 0.16 %  | 230.769                    | 312                                   | 0.16 %  |
| 7    | 460.8 | 461.538                    | 78                                    | 0.16 %  | 461.538                    | 156                                   | 0.16 %  |
| 8    | 921.6 | 923.076                    | 39                                    | 0.16 %  | 923.076                    | 78                                    | 0.16 %  |
| 9    | 2250  | 2250                       | 16                                    | 0 %     | 2250                       | 32                                    | 0 %     |
| 10   | 4500  | NA                         | NA                                    | NA      | 4500                       | 16                                    | 0 %     |

Taking a baud rate of 115.2 Kbps as an example, if  $f_{PCLK} = 36$  MHz, the value in the baud register should be set to 312 (0x38). Based on formula, the calculated baud rate (actual) is  $36000000 / 312 = 115384 = 115.384$  Kbps. The % error between the desired and actual value is calculated based on the formula: (Calculated actual result-Desired)/desired baud rate $\times 100$  %, that is,  $(115.384 - 115.2) / 115.2 \times 100\% = 0.15$  %

## Transmitter

### Transmitter introduction

USART transmitter has its individual TEN control bit. The transmitter and receiver share the same baud rate that is programmable. There is a transmit data buffer (TDR) and a transmit shift register in the USART. The TDBE bit is set whenever the TDR is empty, and an interrupt is generated if the TDBEIEEN is set.

The data written by software is stored in the TDR register. When the shift register is empty, the data will be moved from the TDR register to the shift register so that the data in the transmit shift register is output on the TX pin in LSB mode. The output format depends on the programmed frame format.

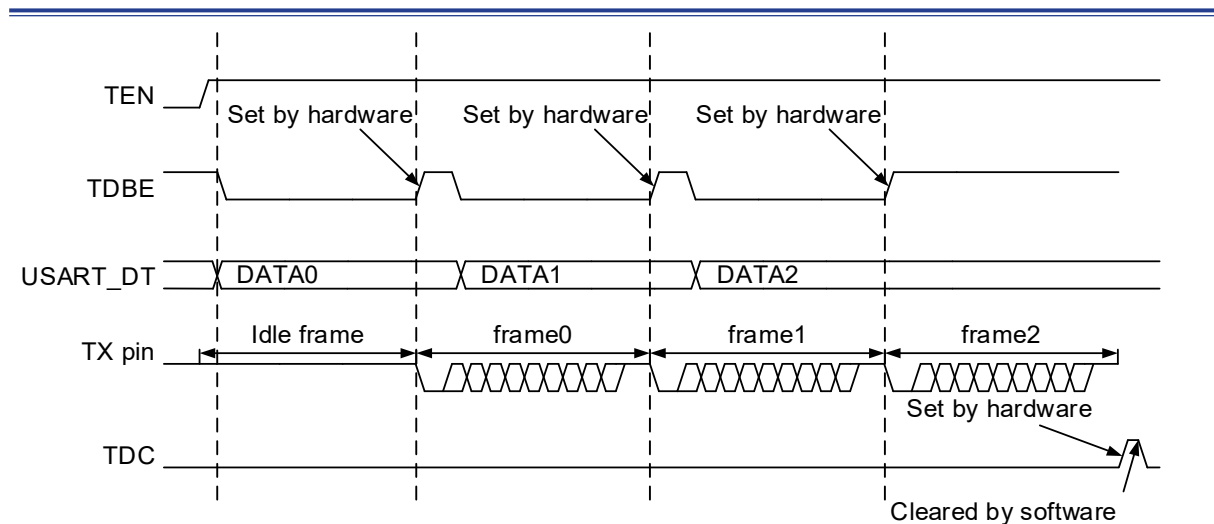
If synchronous transfer or clock output is selected, the clock pulse is output on the CK pin. If the hardware flow control is selected, the control signal is input on the CTS pin.

Note: 1. The TEN bit cannot be reset during data transfer, or the data on the TX pin will be corrupted.  
2. After the TEN bit is enabled, the USART will automatically send an idle frame.

### Transmitter configuration

1. USART enable: Set the UEN bit.
2. Full-duplex/half-duplex configuration: Refer to 12.2 Full-duplex/half-duplex selector.
3. Mode configuration: Refer to 12.3 Mode selector.
4. Frame format configuration: Refer to 12.4 USART frame format and configuration.
5. Interrupt configuration: Refer to 12.10 Interrupt requests.
6. DMA transmission configuration: If the DMA mode is selected, the DMATEN bit (bit 7 in the USART\_CTRL3 register) is set, and configure DMA register accordingly.
7. Baud rate configuration: Refer to 12.6 Baud rate generation.
8. Transmitter enable: When the TEN bit is set, the USART transmitter will send an idle frame.
9. Write operation: Wait until the TDBE bit is set, the data to be transferred will be loaded into the USART\_DT register (This operation will clear the TDBE bit). Repeat this step in non-DMA mode.

10. After the last data expected to be transferred is written, wait until the TDC is set, indicating the end of transfer. The USART cannot be disabled before the flag is set, or transfer error will occur.
11. When  $TDC = 1$ , read access to the USART\_STS register and write access to the USART\_DT register will clear the TDC bit; This bit can also be cleared by writing "0", but this is valid only in DMA mode.



**Figure 58. TDC/TDBE behavior when transmitting**

Note: The interval period between two consecutive data transfers is fixed at 2 PCLK cycles. For example, if APB clock = 72 MHz and clock period = 13.88 ns, after the completion of the stop bit of the last data, the TX pin is able to send the subsequent data only after a delay of  $13.88 \times 2 = 27.76$  ns.

## Receiver

### Receiver introduction

USART receiver has its individual REN control bit (bit 2 in the USART\_CTRL1 register). The transmitter and receiver share the same baud rate that is programmable. There is a receive data buffer (RDR) and a receive shift register in the USART.

The data is input on the RX pin of the USART. When a valid start bit is detected, the receiver ports the data received into the receive shift register in LSB mode. After a full data frame is received, based on the programmed frame format, it will be moved from the receive shift register to the receive data buffer, and the RDBF is set accordingly. An interrupt is generated if the RDBFIEN is set.

If hardware flow control is selected, the control signal is output on the RTS pin.

During data reception, the USART receiver will detect whether there are errors to occur, including framing error, overrun error, parity check error or noise error, depending on software configuration, and whether there are interrupts to generate using the interrupt enable bits.

## Receiver configuration

### Configuration procedure:

1. USART enable: UEN bit is set.
2. Full-duplex/half-duplex configuration: Refer to 12.2 Full-duplex/half-duplex selector.
3. Mode configuration: Refer to 12.3 Mode selector.
4. Frame format configuration: Refer to 12.4 USART frame format and configuration.
5. Interrupt configuration: Refer to 12.10 Interrupt requests.
6. Reception using DMA: If the DMA mode is selected, the DMAREN bit is set, and configure DMA register accordingly.
7. Baud rate configuration: Refer to 12.6 Baud rate generation.
8. Receiver enable: REN bit is set.

### Character reception:

- The RDBF bit is set. It indicates that the content of the shift register is transferred to the RDR (Receiver Data Register). In other words, data is received and can be read (including its associated error flags)
- An interrupt is generated when the RDBFIEN is set.
- The error flag is set when a framing error, noise error or overrun error is detected during reception.
- In DMA mode, the RDNE bit is set after every byte is received, and it is cleared when the data register is read by DMA.
- In non-DMA mode, the RDBF bit is cleared when read access to the USART\_DT register by software. The RDBF flag can also be cleared by writing 0 to it. The RDBF bit must be cleared before the end of next frame reception to avoid overrun error.

### Break frame reception:

- Non-LIN mode: It is handled as a framing error, and the FERR is set. An interrupt is generated if the corresponding interrupt bit is enabled. Refer to framing error described below for details.
- LIN mode: It is handled as a break frame, and the BFF bit is set. An interrupt is generated if the BFIEN is set.

### Idle frame reception:

- It is handled as a data frame, and the IDLEF bit is set. An interrupt is generated if the IDLEIEN is set.

### When a framing error occurs:

- The FERR bit is set.
- The USART receiver moves the invalid data from the receive shift register to the receive data buffer.
- In non-DMA mode, both FERR and RDBF are set at the same time. The latter will generate an interrupt. In DMA mode, an interrupt is generated if the ERRIEN.

#### When an overrun error occurs:

- The ROERR bit is set.
- The data in the receive data buffer is not lost. The previous data is still available when the USART\_DT register is read.
- The content in the receive shift register is overwritten. Afterwards, any data received will be lost.
- An interrupt is generated if the RDBFIEN is set or both ERRIEN and DMAREN are set.
- The ROERR bit is cleared by reading the USART\_STS register and then USART\_DT register in order.

Note: 1. If ROERR is set, it indicates that at least one piece of data is lost, with two possibilities:

If RDBF = 1, it indicates that the last valid data is still stored in the receive data buffer, and can be read.

If RDBF = 0, it indicates that the last valid data in the receive data buffer has already been read.

2. The REN bit cannot be reset during data reception, or the byte that is currently being received will be lost.

### Start bit and noise detection

A start bit detection occurs when the REN bit is set. With the oversampling techniques, the USART receiver samples data on the 3rd, 5th, 7th, 8th, 9th and 10th bits to detect the valid start bit and noise. Table 39 shows the data sampling over start bit and noise detection.

**Table 39. Data sampling over start bit and noise detection**

| Sampled value (3·5·7) | Sampled value (8·9·10) | NERR bit | Start bit validity |
|-----------------------|------------------------|----------|--------------------|
| 000                   | 000                    | 0        | Valid              |
| 001/010/100           | 001/010/100            | 1        | Valid              |
| 001/010/100           | 000                    | 1        | Valid              |
| 000                   | 001/010/100            | 1        | Valid              |
| 111/110/101/011       | Any value              | 0        | Invalid            |
| Any value             | 111/110/101/011        | 0        | Invalid            |

Note: If the sampling values on the 3rd, 5th, 7th, 8th, 9th, and 10th bits do not match the above mentioned requirements, the USART receiver does not think that a correct start bit is received, and thus it will abort the start bit detection and return to idle state waiting for a falling edge.

The USART receiver has the ability to detect noise. In non-synchronous mode, USART receiver samples data on the 7<sup>th</sup>, 8<sup>th</sup> and 9<sup>th</sup> bits, with its oversampling techniques, to distinguish valid data input from noise based on different sampling values, and recover data as well as set NERR (Noise Error Flag) bit.

**Table 40. Data sampling over valid data and noise detection**

| Sampled value | NERR bit | Received bit value | Data validity |
|---------------|----------|--------------------|---------------|
| 000           | 0        | 0                  | Valid         |
| 001           | 1        | 0                  | Invalid       |
| 010           | 1        | 0                  | Invalid       |
| 011           | 1        | 1                  | Invalid       |
| 100           | 1        | 0                  | Invalid       |
| 101           | 1        | 1                  | Invalid       |
| 110           | 1        | 1                  | Invalid       |
| 111           | 0        | 1                  | Valid         |

USART is able to receive data under the maximum allowable deviation condition. Its value depends on the DBN bit of the USART\_CTRL1 register and the DIV[3:0] of the USART\_BAUDR register.

Note: The maximum allowable deviations stated in the table below are calculated based on 115.2 Kbps. The actual deviations may vary with the settings of baud rate. In other words, the greater the baud rate is, the smaller the maximum allowable deviation; in contrast, when the baud rate gets smaller, the maximum allowable deviation will get bigger.

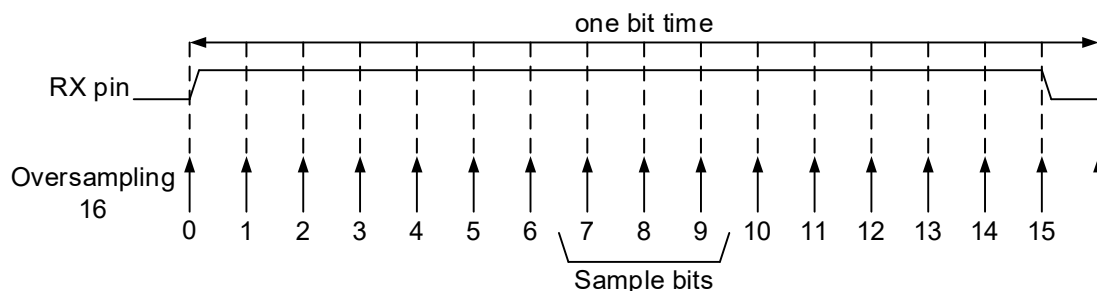
**Table 41. Maximum allowable deviation**

| DBN[1:0] | DIV[3:0] = 0 | DIV[3:0] != 0 |
|----------|--------------|---------------|
| 00       | 3.75 %       | 3.33 %        |
| 01       | 3.41 %       | 3.03 %        |
| 10       | 4.16 %       | 3.7 %         |

When noise is detected in a data frame:

- The NERR bit is set at the same time as the RDBF bit
- The invalid data is transferred from the receive shift register to the receive data buffer.
- No interrupt is generated in non-DMA mode. However, since the NERR bit is set at the same time as the RDBF bit, the RDBF bit will generate an interrupt. In DMA mode, an interrupt will be issued if the ERRIEN is set.

The NERR bit is cleared by read access to USART\_STS register followed by the USART\_DT read operation.

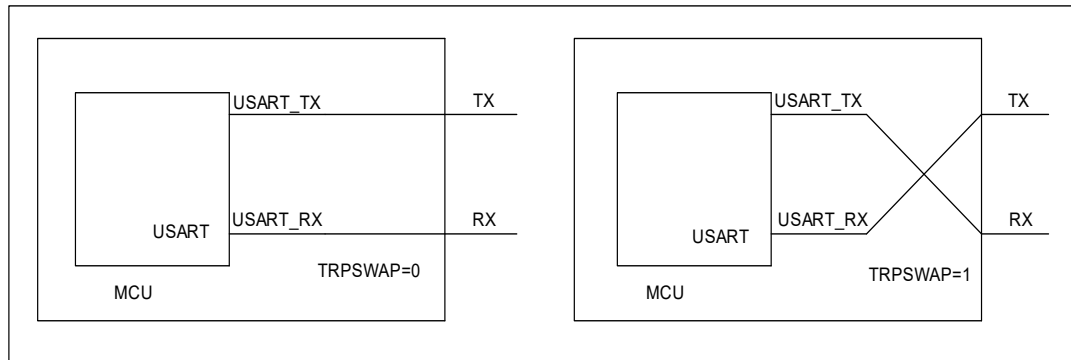


**Figure 59. Data sampling for noise detection**

## Tx/Rx swap

When the TRPSWAP bit (USART\_CTRL2[15]) is set, Tx/Rx pin can be swapped. Two common scenes are listed below:

- If the Tx/Rx were reversed while the user attempts to connect the device externally to a RS-232 chip, they can be swapped through the TRPSWAP bit, without the need of hardware intervention.
- If the user only connected the master Tx to the slave Rx in full-duplex mode, the Tx/Rx can be interchangeable with the TRPSWAP bit, after the master and slave are swapped, without the need of hardware intervention.



**Figure 60. Tx/Rx swap**

Note: The SWAP (USART\_CTRL2[15]) can be modified only when the USART is disabled (UEN = 0).

## Interrupt requests

USART interrupt generator serves as a control center of USART interrupts. It is used to monitor the interrupt source inside the USART in real time and the generation of interrupts according to the programmed interrupt control bits. Table 41 shows the USART interrupt source and interrupt enable control bit. An interrupt will be generated over an event when the corresponding interrupt enable bit is set.

**Table 42. USART interrupt request**

| Interrupt event                              | Event flag            | Enable bit            |
|--|-----------------------|-----------------------|
| Transmit data register empty                 | TDBE                  | TDBEIEN               |
| CTS flag                                     | CTSCF                 | CTSCFIEN              |
| Transmit data complete                       | TDC                   | TDCIEN                |
| Receive data buffer full                     | RDBF                  | RDBFIEN               |
| Receiver overflow error                      | ROERR                 |                       |
| Idle flag                                    | IDLEF                 | IDLEIEN               |
| Parity error                                 | PERR                  | PERRIEN               |
| Break frame flag                             | BFF                   | BFIEN                 |
| Noise error, overflow error or framing error | NERR or ROERR or FERR | ERRIEN <sup>(1)</sup> |

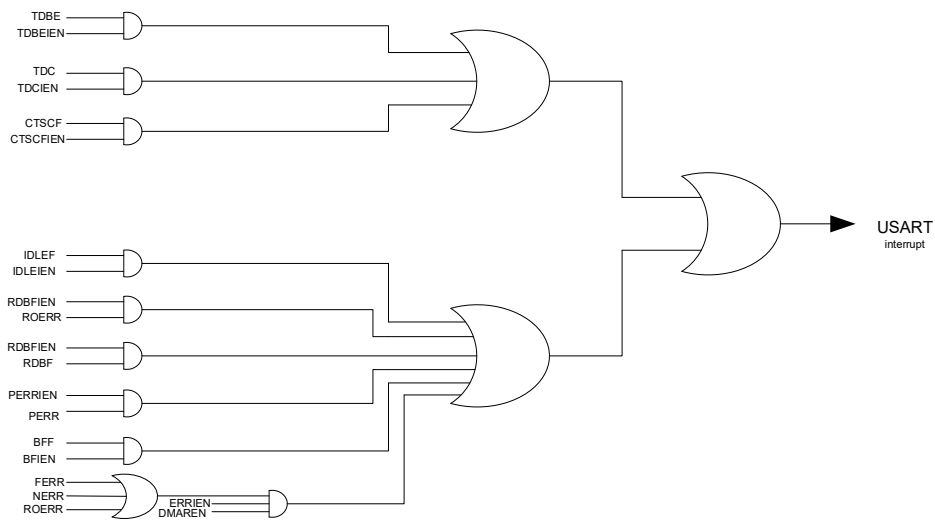


Figure 61. USART interrupt map diagram

## I/O pin control

The following five interfaces are used for USART communication.

RX: Serial data input.

TX: Serial data output. In single-wire half-duplex and Smartcard mode, the TX pin is used as an I/O for data transmission and reception.

CK: Transmitter clock output. The output CLK phase, polarity and frequency can be programmable.

CTS: Transmitter input. Send enable signal in hardware flow control mode.

RTS: Receiver output. Send request signal in hardware flow control mode.

## USART registers

These peripheral registers must be accessed by words (32 bits).

Table 43. USART register map and reset value

| Register    | Offset | Reset value |
|-------------|--------|-------------|
| USART_STS   | 0x00   | 0x0000 00C0 |
| USART_DT    | 0x04   | 0x0000      |
| USART_BAUDR | 0x08   | 0x0000      |
| USART_CTRL1 | 0x0C   | 0x0000      |
| USART_CTRL2 | 0x10   | 0x0000      |
| USART_CTRL3 | 0x14   | 0x0000      |
| USART_GDIV  | 0x18   | 0x0000      |

## Status register (USART\_STS)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:10 | Reserved | 0x000000    | resd | Forced 0 by hardware.  |
| Bit 9     | CTSCF    | 0           | rw0c | CTS change flag<br>This bit is set by hardware when the CTS status line changes. It is cleared by software.<br>0: No change on the CTS status line<br>1: A change occurs on the CTS status line.   |
| Bit 8     | BFF      | 0           | rw0c | Break frame flag<br>This bit is set by hardware when a break frame is detected. It is cleared by software.<br>0: Break frame is not detected.<br>1: Break frame is detected.   |
| Bit 7     | TDBE     | 1           | ro   | Transmit data buffer empty<br>This bit is set by hardware when the transmit data buffer is empty. It is cleared by a USART_DT register write operation.<br>0: Data is not transferred to the shift register.<br>1: Data is transferred to the shift register.  |
| Bit 6     | TDC      | 1           | rw0c | Transmit data complete<br>This bit is set by hardware at the end of transmission. It is cleared by software. (Option 1: read access to USART_STS register followed by a USART_DT write operation; Option 2: Write "0" to this bit.)<br>0: Transmission is not completed.<br>1: Transmission is completed.  |
| Bit 5     | RDBF     | 0           | rw0c | Receive data buffer full<br>This bit is set by hardware when the data is transferred from the shift register to the USART_DT register. It is cleared by software. (Option 1: read USART_DT register; Option 2: write "0" to this bit.)<br>0: Data is not received.<br>1: Data is received.   |
| Bit 4     | IDLEF    | 0           | ro   | Idle flag<br>This bit is set by hardware when an idle line is detected. It is cleared by software. (Read USART_DT register followed by a USART_DT read operation)<br>0: No idle line is detected.<br>1: Idle line is detected.   |
| Bit 3     | ROERR    | 0           | ro   | Receiver overflow error<br>This bit is set by hardware when the data is received while the RDNE is still set. It is cleared by software. (Read USART_STS register followed by a USART_DT read operation)<br>0: No overflow error<br>1: Overflow error is detected.<br>Note: When this bit is set, the DT register content will not be lost, but the subsequent data will be overwritten. |
| Bit 2     | NERR     | 0           | ro   | Noise error<br>This bit is set by hardware when noise is detect on a received frame. It is cleared by software. (Read USART_STS register followed by a USART_DT read operation)<br>0: No noise is detected.<br>1: Noise is detected.   |



| Bit   | Name | Reset value | Type | Description   |
|-------|------|-------------|------|---|
| Bit 1 | FERR | 0           | ro   | Framing error<br>This bit is set by hardware when a stop bit error (low), excessive noise or break frame is detected. It is cleared by software. USART_STS register followed by a USART_DT read operation)<br>0: No framing error is detected.<br>1: Framing error is detected. |
| Bit 0 | PERR | 0           | ro   | Parity error<br>This bit is set by hardware when parity error occurs. It is cleared by software. USART_STS register followed by a USART_DT read operation)<br>0: No parity error occurs.<br>1: Parity error occurs.   |

### Data register (USART\_DT)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31:9 | Reserved | 0x000000    | resd | Kept at its default value.   |
| Bit 8:0  | DT       | 0x000       | rw   | Data value<br>This register provides read and write function. When transmitting with the parity bit enabled, the value written in the MSB bit will be replaced by the parity bit. When receiving with the parity bit enabled, the value in the MSB bit is the received parity bit. |

### Baud rate register (USART\_BAUDR)

Note: If the TEN and REN bits are disabled, the baud counter stops counting.

| Bit       | Name     | Reset value | Type | Description                                     |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.                      |
| Bit 15:0  | DIV      | 0x0000      | rw   | Divider<br>This field define the USART divider. |

### Control register 1 (USART\_CTRL1)

| Bit         | Name     | Reset value | Type | Description  |
|-------------|----------|-------------|------|--|
| Bit 31:29   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 28      | DBN1     | 0x0         | rw   | Data bit num<br>This bit, along with the DBN0 bit, is used to program the number of data bits.<br>10: 7 data bits<br>00: 8 data bits<br>01: 9 data bits<br>11: Write operation forbidden.  |
| Bit 27:26   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 25 : 21 | TSDT     | 0x00        | rw   | Transmit start delay time<br>In RS485 mode, the first data (in sequential transmit mode) is transmitted after a period of time of being written so as to ensure that the transfer direction of the external transmitter/receiver to switch back to transmit. This time depends on the TSDT value, in unit of 1/16 baud rate. |

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 20:16 | TCDT     | 0x00        | rw   | transmit complete delay time<br>In RS485 mode, a period of time (delay) is needed before the last data transfer is complete even if the last STOP bit has been transferred. This time duration allows the transfer direction of the external receiver/transmitter to switch back to receive. This time depends on the TCDT value, in unit of 1/16 baud rate.                            |
| Bit 15:14 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 13    | UEN      | 0           | rw   | USART enable<br>0: USART is disabled.<br>1: USART is enable.  |
| Bit 12    | DBN0     | 0           | rw   | Data bit num<br>This bit, along with DBN1, is used to program the number of data bits.<br>10: 7 data bits<br>00: 8 data bits<br>01: 9 data bits<br>11: Write operation forbidden.   |
| Bit 11    | WUM      | 0           | rw   | Wakeup mode<br>This bit determines the way to wake up silent mode.<br>0: Waken up by idle line<br>1: Waken up by ID match   |
| Bit 10    | PEN      | 0           | rw   | Parity enable<br>This bit is used to enable hardware parity control (generation of parity bit for transmission; detection of parity bit for reception). When this bit is enabled, the MSB bit of the transmitted data is replaced with the parity bit; Check whether the parity bit of the received data is correct.<br>0: Parity control is disabled.<br>1: Parity control is enabled. |
| Bit 9     | PSEL     | 0           | rw   | Parity selection<br>This bit selects the odd or even parity after the parity control is enabled.<br>0: Even parity<br>1: Odd parity   |
| Bit 8     | PERRIEN  | 0           | rw   | PERR interrupt enable<br>0: Interrupt is disabled.<br>1: Interrupt is enabled.  |
| Bit 7     | TDBEIEN  | 0           | rw   | TDBE interrupt enable<br>0: Interrupt is disabled.<br>1: Interrupt is enabled.  |
| Bit 6     | TDCIEN   | 0           | rw   | TDC interrupt enable<br>0: Interrupt is disabled.<br>1: Interrupt is enabled.   |
| Bit 5     | RDBFIEN  | 0           | rw   | RDBF interrupt enable<br>0: Interrupt is disabled.<br>1: Interrupt is enabled.  |
| Bit 4     | IDLEIEN  | 0           | rw   | IDLE interrupt enable<br>0: Interrupt is disabled.<br>1: Interrupt is enabled.  |

| Bit   | Name | Reset value | Type | Description   |
|-------|------|-------------|------|---|
| Bit 3 | TEN  | 0           | rw   | Transmitter enable<br>This bit enables the transmitter.<br>0: Transmitter is disabled.<br>1: Transmitter is enabled.  |
| Bit 2 | REN  | 0           | rw   | Receiver enable<br>This bit enables the receiver.<br>0: Receiver is disabled.<br>1: Receiver is enabled.  |
| Bit 1 | RM   | 0           | rw   | Receiver mute<br>This bit determines if the receiver is in mute mode or not. It is set or cleared by software. When the idle line is used to wake up from mute mode, this bit is cleared by hardware after wake up. When the address match is used to wake up from mute mode, it is cleared by hardware after wake up. When address mismatches, this bit is set by hardware to enter mute mode again.<br>0: Receiver is in active mode.<br>1: Receiver is in mute mode. |
| Bit 0 | SBF  | 0           | rw   | Send break frame<br>This bit is used to send a break frame. It can be set or cleared by software. Generally speaking, it is set by software and cleared by hardware at the end of break frame transmission.<br>0: No break frame is transmitted.<br>1: Break frame is transmitted.  |

### Control register 2 (USART\_CTRL2)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:28 | ID       | 0x0         | rw   | USART identification<br>This field holds the upper four bits of USART ID. It is configurable.   |
| Bit 27:16 | Reserved | 0x000       | resd | Kept at its default value.  |
| Bit 15    | TRPSWAP  | 0           | rw   | Transmit/receive pin swap<br>0: Transmit/receive pin is not swappable<br>1: Transmit/receive pin is swappable   |
| Bit 14    | LINEN    | 0           | rw   | LIN mode enable<br>0: LIN mode is disabled.<br>1: LIN mode is enabled.  |
| Bit 13:12 | STOPBN   | 0           | rw   | STOP bit num<br>These bits are used to program the number of stop bits.<br>00: 1 stop bit<br>01: 0.5 stop bit<br>10: 2 stop bits<br>11: 1.5 stop bits |
| Bit 11    | CLKEN    | 0           | rw   | Clock enable<br>This bit is used to enable the clock pin for synchronous mode or Smartcard mode.<br>0: Clock is disabled.<br>1: Clock is enabled.     |

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 10   | CLKPOL   | 0           | rw   | Clock polarity<br>In synchronous mode or Smartcard mode, this bit is used to select the polarity of the clock output on the clock pin in idle state.<br>0: Clock output low<br>1: Clock output high  |
| Bit 9    | CLKPHA   | 0           | rw   | Clock phase<br>This bit is used to select the phase of the clock output on the clock pin in synchronous mode or Smartcard mode.<br>0: Data capture is done on the first clock edge.<br>1: Data capture is done on the second clock edge.   |
| Bit 8    | LBCP     | 0           | rw   | Last bit clock pulse<br>This bit is used to select whether the clock pulse of the last data bit transmitted is output on the clock pin in synchronous mode.<br>0: The clock pulse of the last data bit is no output on the clock pin.<br>1: The clock pulse of the last data bit is output on the clock pin. |
| Bit 7    | Reserved | 0           | resd | Kept at its default value.   |
| Bit 6    | BFIEN    | 0           | rw   | Break frame interrupt enable<br>0: Disabled<br>1: Enabled  |
| Bit 5    | BFBN     | 0           | rw   | Break frame bit num<br>This bit is used to select 11-bit or 10-bit break frame.<br>0: 10-bit break frame<br>1: 11-bit break frame  |
| Bit 4    | IDBN     | 0           | rw   | Identification bit num<br>This bit is used to select ID bit number.<br>0: 4 bit<br>1: Data bit - 1 bit<br>Note: When this bit is set, in 7, 8 or 8-bit data mode, the ID bit number is the lower 6, 7 or 8 bit (5, 6, or 7 bit when parity check is enabled), respectively.                                  |
| Bit 3: 0 | IDL      | 0x0         | rw   | USART identification<br>This field holds the lower four bits of USART ID. It is configurable.  |

Note: These three bits (CLKPOL, CLKPHA and LBCP) should not be changed while the transmission is enabled.

### Control register 3 (USART\_CTRL3)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Forced 0 by hardware.   |
| Bit 15    | DEP      | 0           | rw   | DE polarity selection<br>0: High level active<br>1: Low level active  |
| Bit 14    | RS485EN  | 0           | rw   | RS485 enable<br>This bit is used to enable RS485 mode. In RS485 mode, the USART controls the transfer direction of the external receiver/transmitter through the DE signal.<br>0: RS485 mode disabled. The control signal DE output is disabled. RTS pin is used in RS232 mode.<br>1: RS485 mode enabled. The control signal DE outputs on the RTS pin. |

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 13:11 | Reserved | 0x0         | resd | Forced 0 by hardware.  |
| Bit 10    | CTSCFIEN | 0           | rw   | CTSCF interrupt enable<br>0: CTSCF interrupt disabled<br>1: CTSCF interrupt enabled  |
| Bit 9     | CTSEN    | 0           | rw   | CTS enable<br>0: CTS is disabled.<br>1: CTS is enabled.  |
| Bit 8     | RTSEN    | 0           | rw   | RTS enable<br>0: RTS is disabled.<br>1: RTS is enabled.  |
| Bit 7     | DMATEN   | 0           | rw   | DMA transmitter enable<br>0: DMA transmitter is disabled.<br>1: DMA transmitter is enabled.  |
| Bit 6     | DMAREN   | 0           | rw   | DMA receiver enable<br>0: DMA receiver is disabled.<br>1: DMA receiver is enabled.   |
| Bit 5     | SCMEN    | 0           | rw   | Smartcard mode enable<br>0: Smartcard mode is disabled.<br>1: Smartcard mode is enabled.   |
| Bit 4     | SCNACKEN | 0           | rw   | Smartcard NACK enable<br>This bit is used to send NACK when parity error occurs.<br>0: NACK is disabled when parity error occurs.<br>1: NACK is enabled when parity error occurs.    |
| Bit 3     | SLBEN    | 0           | rw   | Single-wire bidirectional half-duplex enable<br>0: Single-wire bidirectional half-duplex is disabled.<br>1: Single-wire bidirectional half-duplex is enabled.                        |
| Bit 2     | IRDALP   | 0           | rw   | IrDA low-power mode<br>This bit is used to configure IrDA low-power mode.<br>0: IrDA low-power mode is disabled.<br>1: IrDA low-power mode is enabled.                               |
| Bit 1     | IRDAEN   | 0           | rw   | IrDA enable<br>0: IrDA is disabled.<br>1: IrDA is enabled.   |
| Bit 0     | ERRIEN   | 0           | rw   | Error interrupt enable<br>An interrupt is generated when a framing error, overflow error or noise error occurs.<br>0: Error interrupt is disabled.<br>1: Error interrupt is enabled. |

## Guard time and divider register (USART\_GDIV)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x00        | resd | Forced 0 by hardware.  |
| Bit 15:8  | SCGT     | 0x00        | rw   | Smartcard guard time value<br>This field specifies the guard time value. The transmission complete flag is set after this guard time in smartcard mode.  |
| Bit 7: 0  | ISDIV    | 0x00        | rw   | IrDA/smartcard division<br>In IrDA mode:<br>8 bit [7:0] is valid. It is valid in common mode and must be set to 00000001. In low-power mode, it divides the peripheral clock to serve as the period base of the pulse width;<br>00000000: Reserved. Do not write.<br>00000001: Divided by 1<br>00000010: Divided by 2<br>.....<br>Smartcard mode:<br>The lower 5 bit [4:0] is valid. This division is used to divide the peripheral clock to provide clock for the Smartcard. Configured as follows:<br>00000: Reserved. Do not write.<br>00001: Divided by 2<br>00010: Divided by 4<br>00011: Divided by 6<br>..... |

# 13 Serial peripheral interface (SPI)

## SPI introduction

The SPI interface supports either the SPI protocol or the I<sup>2</sup>S protocol, depending on software configuration. This chapter gives an introduction of the main features and configuration procedure of SPI used as SPI or I<sup>2</sup>S.

## Function overview

### SPI description

The SPI can be configured as host or slave based on software configuration, supporting full-duplex, reception-only full-duplex and transmission-only/reception-only half-duplex modes, DMA transfer, and automatic CRC function of SPI internal hardware. In the meantime, the SPI interface can be compatible with the TI protocol through software configurations.

#### SPI block diagram:

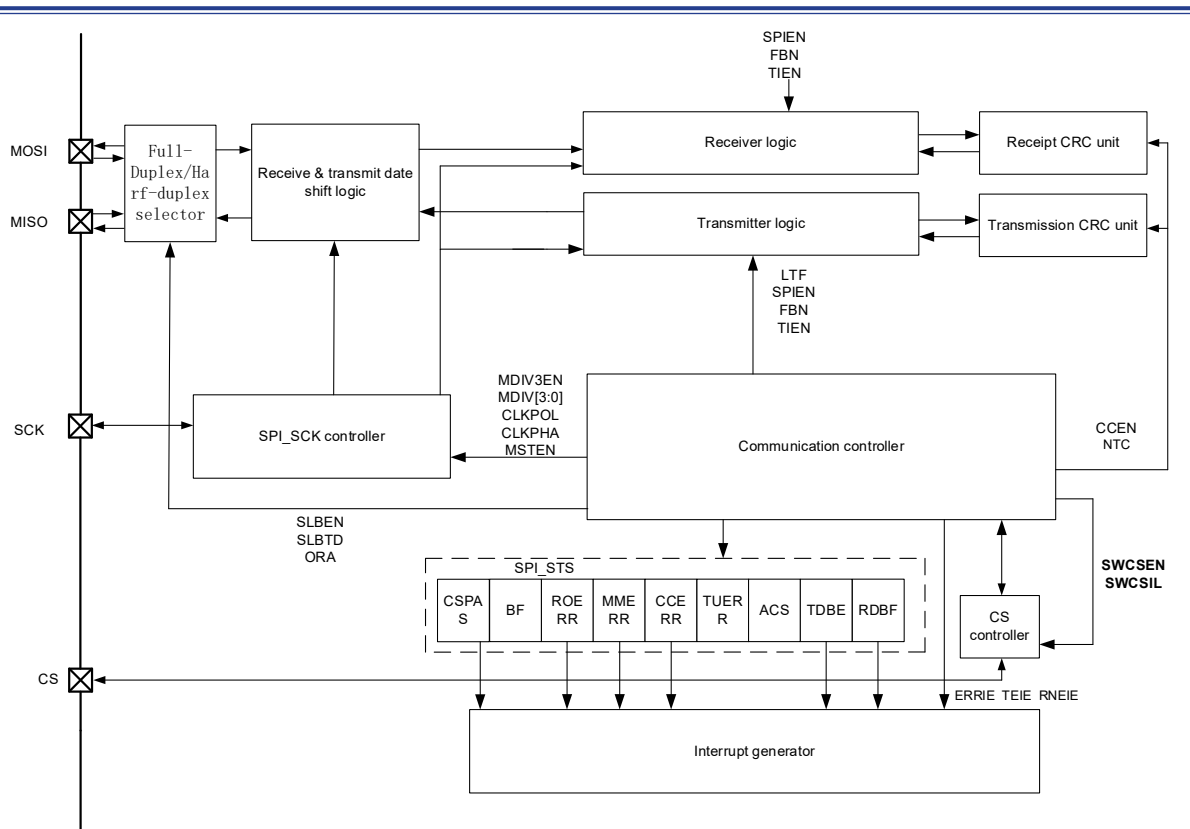


Figure 62. SPI block diagram

### Main features as SPI:

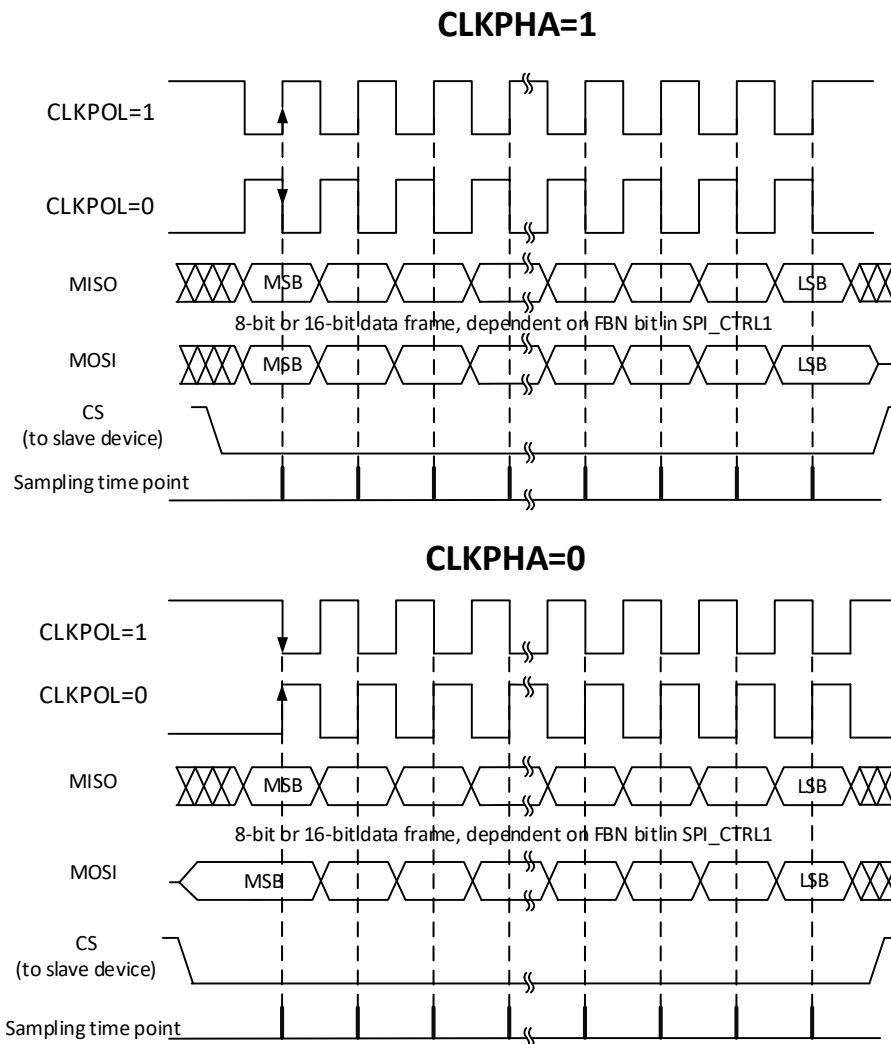
- Full-duplex or half-duplex communication
  - Full-duplex synchronous communication (supporting reception-only mode to release IO for transmission)
  - Half-duplex synchronous communication (transfer direction is configurable: receive or transmit)
- Master or slave mode
- CS signal processing mode
  - CS signal processing by hardware
  - CS signal processing by software
- 8-bit or 16-bit frame format
- Communication frequency and prescalers (Frequency up to 48M, and prescalers up to  $f_{CLK}/2$ )
- Programmable clock polarity and phase
- Programmable data transfer order (MSB-first or LSB-first)
- Programmable error interrupt flags (CS pulse error, receiver overflow error, master mode error and CRC error)
- Programmable transmit data buffer empty interrupt and receive data buffer full interrupt
- Support transmission and reception using DMA
- Support hardware CRC transmission and error checking
- Busy status flag
- Compatible with the TI protocol

### Clock signal phase and polarity

The CLKPOL and CLKPHA bits in the SPI\_CTRL1 register can be used to combine four possible timings. The CLKPOL (clock polarity) bit is used to indicate the polarity of clock output in idle state when there is no data transmission. This bit is valid for devices in master mode and slave mode. When the CLKPOL bit is cleared, the SCK pin remains low in idle state; when the CLKPOL bit is set, the SCK pin remains high in idle state.

When the CLKPHA (clock phase) bit is set, data bit sampling starts from the second edge of SCK clock (falling edge when CLKPOL = 0, rising edge when CLKPOL = 1), and data is latched at the second clock edge. When the CLKPHA bit is cleared, data bit sampling starts from the first edge of SCK clock (rising edge when CLKPOL = 0, falling edge when CLKPOL = 1), and data is latched at the first clock edge.





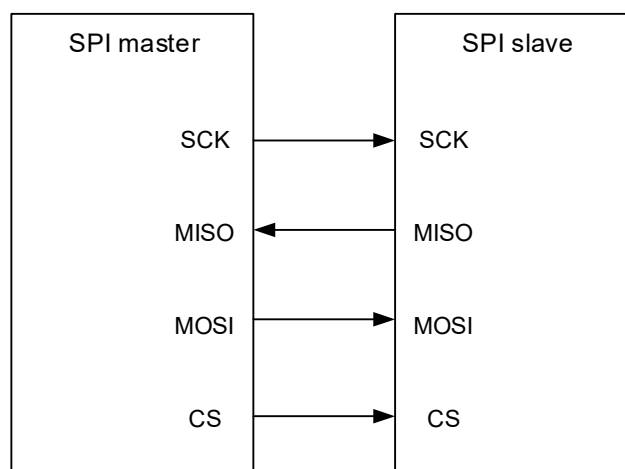
**Figure 63. Data clock timing diagram**

### Full-duplex/half-duplex selector

When used as an SPI interface, it supports four synchronous modes: two-wire unidirectional full-duplex, single-wire unidirectional receive only, single-wire bidirectional half-duplex transmit and single-wire bidirectional half-duplex receive.

Figure 64 shows the two-wire unidirectional full-duplex mode and SPI IO connection:

The SPI operates in two-wire unidirectional full-duplex mode when the SLBEN bit and the ORA bit is both 0. In this case, the SPI supports data transmission and reception at the same time. IO connection is as follows:

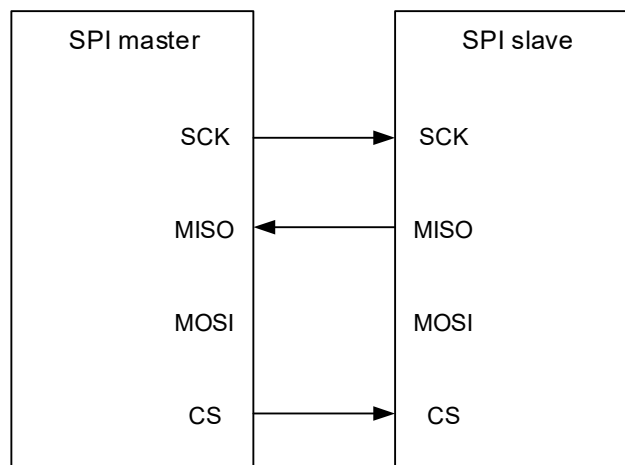


**Figure 64. SPI two-wire unidirectional full-duplex connection**

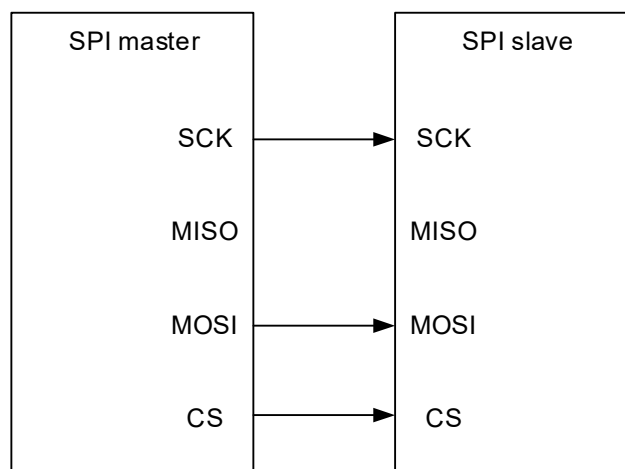
In either master or slave mode, it is required to wait until the RDBF bit and TDBE bit is set, and BF = 0 before disabling the SPI or entering power-saving mode (or disabling SPI system clock).

Figure 65 shows the single-wire unidirectional receive-only mode and SPI IO connection

The SPI operates in single-wire unidirectional receive-only mode when the SLBEN is 0 and the ORA is set. In this case, the SPI can be used only for data reception (transmission is not supported). The MISO pin transmits data in slave mode and receives data in master mode. The MOSI pin transmits data in master mode and receives data in slave mode.



**Figure 65. Single-wire unidirectional receive only in SPI master mode**



**Figure 66. Single-wire unidirectional receive only in SPI slave mode**

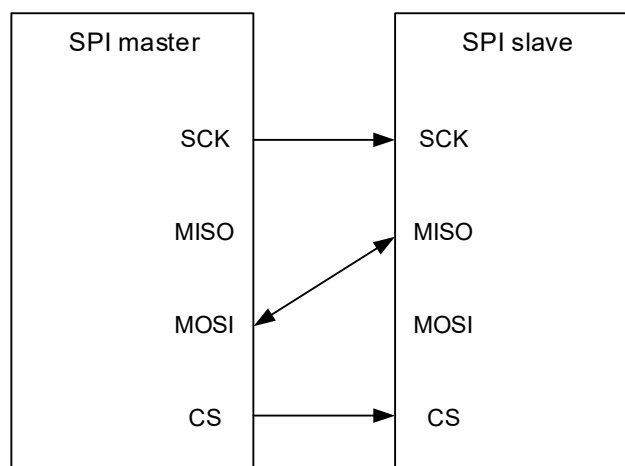
In master mode, it is necessary to wait until the second-to-last RDBF bit is set and then another SPI\_CPK period before disabling the SPI. The last RDBF must be set before entering power-saving mode (or disabling SPI system clock).

In slave mode, there is no need to check any flag before disabling the SPI. However, it is required to wait until the BF becomes 0 before entering power-saving mode.

Figure 67 shows single-wire bidirectional half-duplex mode and SPI IO connection

When the SLBEN is set, the SPI operates in single-wire bidirectional half-duplex mode. In this case, the SPI supports data reception and transmission alternately. In master mode, the MOSI pin transmits or receives data in master mode, while the MISO pin is released. In slave mode, the MISO pin transmits or receives data, but the MOSI pin is released.

The SLBTD bit is used by software to configure transfer direction. When the SLBTD bit is set, the SPI can be used only for data transmission; when the SLBTD bit is 0, the SPI can be used only for data reception.



**Figure 67. Single-wire bidirectional half-duplex mode**

When the SPI is selected for data transmission in single-wire bidirectional half-duplex mode (master or slave), the TDBE bit must be set, and the BF must be 0 before disabling the SPI. The power-saving mode (or disabling SPI system clock) cannot be entered unless the SPI is disabled.

In master mode, when the SPI is selected for data reception in single-wire bidirectional half-duplex mode, it is required to wait until the second-to-last RDBF is set and then another SPI\_SCK period before disabling the SPI. And the last RDBF must be set before entering power-saving mode (or disabling SPI system clock).

In slave mode, when the SPI is selected for data reception in single-wire bidirectional half-duplex mode, there is no need to check any flags before disabling the SPI. However, the BT must be 0 before entering power-saving mode (or disabling SPI system clock).

## Chip select controller

The Chip select controller (CS) is used to enable hardware or software control for chip select signals through software configuration. This controller is used to select master/slave device in multi-processor mode, and to avoid conflicts on the data lines by enabling the SCK signal output followed by CS signal. The hardware and software configuration procedure is detailed as follows, along with their respective input/output in master and slave mode.

### CS hardware configuration procedure:

In master mode with CS being as an output, HWCSOE = 1, SWCSEN = 0, the CS hardware control is enabled. If the SPI is enabled, low level is output on the CS pin. The CS signal is then released after the SPI is disabled and the transmission is complete.

In master mode with CS being as an input, HWCSOE = 0, SWCSEN = 0, the CS hardware control is enabled. At this point, the SPI is automatically disabled by hardware and enters slave mode as soon as the CS pin low is detected by master SPI. The mode error flag (MMERR bit) is set at the same time. An interrupt is generated if ERRIE = 1. When the MMERR is set, the SPIEN and MSTEN cannot be set by software. The MMERR is cleared by read or write access to the SPI\_STS register followed by write operation to the SPI\_CTRL1 register.

In slave mode with CS being as an input, HWCSOE = 0, SWCSEN = 0, the CS hardware control is enabled. The slave selects whether to transmit / receive data based on the level on the CS pin. The slave is selected for data reception and transmission only when the CS pin is low.

### CS software configuration procedure:

In master mode with CS being as an input, SWCSEN = 1, the CS software control is enabled. When SWCSIL = 0, the SPI is automatically disabled by hardware and enters slave mode. The mode error flag (MMERR bit) is set at this time. An interrupt is generated if ERRIE = 1. When the MMERR bit is set, the SPIEN and MSTEN bits cannot be set by software. The MMERR bit is cleared by read or write access to the SPI\_STS register followed by write operation to the SPI\_CTRL1 register.

In slave mode with CS being as an input, SWCSEN = 1, the CS software control is enabled. The SPI judges the CS signal with the SWCSIL bit, instead of CS pin. When SWCSIL = 0, the slave is selected for data reception and transmission.

## SPI\_SCK controller

The SPI protocol adopts synchronous transmission. In master mode with the SPI being used as SPI, it is required to generate a communication clock for data reception and transmission on the SPI, and the communication clock should be output to the slave via IO for data reception and transmission. In slave mode, the communication clock is provided by peripherals, and is input to the SPI via IO. In all, the SPI\_SCK controller is used for the generation and distribution of SPI\_SCK, with the configuration procedure detailed as follows:

#### **SPI\_SCK controller configuration procedure:**

- Clock polarity and clock phase selection: It is selected by setting the CLKPOL and CLKPHA bit.
- Clock prescaler selection: Select the desired PCLK frequency by setting the CRM bit. Select the desired prescaler by setting the MDIV[3:0] bit.
- Master/slave selection: Select SPI as master or slave by setting the MSTEN bit.

Note that the clock output is activated after the SPI is enabled in master reception-only mode, and it remains output until when the SPI is disabled and the reception is complete.

### **CRC**

There is an independent transmission and reception CRC calculation unit in the SPI. When used as SPI through software configuration, the SPI enables CRC calculation and CRC check automatically while the user is reading or writing through DMA or CPU. During the transmission, if the received data is not consistent with, detected by hardware, the data in the SPI\_RCRC register, and such data is exactly the CRC value, then the CCERR bit will be set. An interrupt is generated if ERRIE = 1.

The CRC function and configuration procedure of the SPI are described as follows.

#### **CRC configuration procedure**

- CRC calculation polynomial is configured by setting the SPI\_CPOLY register.
- CRC enable: The CRC calculation is enabled by setting the CCEN bit. This operation will reset the SPI\_RCRC and SPI\_TCRC registers.
- Select if or when the NTC bit is set, depending on DMA or CPU data register. See the following descriptions.

#### **Transmission using DMA**

When DMA is used to write the data to be transmitted, if the CCEN bit is enabled, the hardware calculates the CRC value automatically according to the value in the SPI\_CPOLY register and each transmitted data, and sends the CRC value at the end of the last data transmission. This result is regarded as the value of the SPI\_TCRC register.

#### **Reception using DMA**

When DMA is used to read the data to be received, if the CCEN bit is enabled, the hardware calculates the CRC value automatically according to the value in the SPI\_CPOLY register and each received data, and waits until the completion of CRC data reception at the end of the last data reception before comparing the received CRC value with the value of the SPI\_RCRC register. If check error occurs, the CCERR flag is set. An interrupt is generated if the ERRIE bit is enabled.

#### **Transmission using CPU**

Unlike DMA mode, after writing the last data to be transmitted, the CPU mode requires the NTC bit to be set by software before the end of the last data transmission.

#### **Reception using CPU**

In two-wire unidirectional full-duplex mode, follow CPU transmission mode to operate the NTC bit, the CRC calculation and check in CPU reception mode will be completed automatically.

In single-wire unidirectional reception-only mode and single-wire bidirectional reception-only mode, it is required to set the NTC bit before the software receives the last data when the second-to-last data is received.

## DMA transfer

The SPI supports write and read operations with DMA. Refer to the following configuration procedure.

Special attention should be paid to: when the CRC calculation and check is enabled, the number of data transferred by DMA is configured as the number of the data to be transferred. The number of data read with DMA is configured as the number of the data to be received. In this case, the hardware will send CRC automatically at the end of full transfer, and the receiver will also perform CRC check. Note that the received CRC data will be moved into the SPI\_DT register by hardware, with the RDBF being set, and the DMA read request will be sent if then DAM transfer is enabled. Hence, it is recommended to read the SPI\_DT register to get the CRC value at the end of CRC reception in order to avoid the upcoming transfer error.

### Transmission with DMA

- Select DMA channel: Select a DMA channel for the current SPI from DMA channel map table described in DMA chapter.
- Configure the destination of DMA transfer: Configure the SPI\_DT register address as the destination address bit of DMA transfer in the DMA control register. Data will be sent to this address after transmit request is received by DMA.
- Configure the source of DMA transfer: Configure the memory address as the source of DMA transfer in the DMA control register. Data will be loaded into the SPI\_DT register from the memory address after transmit request is received by DMA.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure the channel priority of DMA transfer in the DMA control register.
- Configure DMA interrupt generation after half or full transfer in the DMA control register.
- Enable DMA transfer channel in the DMA control register.

### Reception with DMA

- Select DMA transfer channel: Select a DMA channel for the current SPI from DMA channel map table described in DMA chapter.
- Configure the destination of DMA transfer: Configure the memory address as the destination of DMA transfer in the DMA control register. Data will be loaded from the SPI\_DT register to the programmed destination after reception request is received by DMA.
- Configure the source of DMA transfer: Configure the SPI\_DT register address as the source of DMA transfer in the DMA control register. Data will be loaded from the SPI\_DT register to the programmed destination after reception request is received by DMA.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure DMA interrupt generation after half or full transfer in the DMA control register
- Enable DMA transfer channel in the DMA control register.

## TI mode

The SPI interface is compatible with the TI protocol. The TI mode is enabled by setting the TIEN bit.

In this mode, the SPI interface will generate a communication clock SPI\_CLK in accordance with the TI protocol. This means that the SPI\_CLK polarity and phase are forced to conform to the TI protocol requirements, without the need of the intervention of CLKPOL and CLKPHA bits. Thus the CLKPOL and CLKPHA bits cannot be used to change the polarity and phase of the SPI\_CLK either.

In this mode, the SPI interface will generate a CS signal in accordance with the TI protocol, meaning that the CS input and output are forced to conform to the TI protocol requirements, without the need of the intervention of SWCSEN, SWCSIL and HWCSE bits. Thus, the SWCSEN, SWCSIL and HWCSE bits cannot be used for CS signal management either.

In slave mode, once the TI mode is enabled, the SPI slave controls the MISO pin only during data transmission, meaning that the MISO pin state remains Hi-Z in idle state.

In slave mode, once the TI mode is enabled, the SPI interface is capable of detecting CS pulse errors during data transmission, and setting the CSPAS bit (It is cleared by reading the SPI\_STS) as soon as a CS pulse error is detected. At this point, the detected pulse error will be discarded by the SPI. However, since there is something wrong with the CS signal, the software should disable the SPI slave and re-configure the SPI master before re-enabling the SPI slave for communication.

## Transmitter

The SPI transmitter is clocked by SPI\_SCK controller. It can output different data frame formats, depending on software configuration. There is a SPI\_DT register available in the SPI that is used to be written with the data to be transmitted. When the transmitter is clocked, the contents in the SPI\_DT register are copied into the data buffer (Unlike SPI\_DT, it is driven by SPI\_SCK, and controlled by hardware, instead of software), and sent out in order based on the programmed frame format.

Both DMA and CPU can be used for write operation. For DMA transfer, refer to DMA transfer section for more details. For CPU transfer, attention should be paid to the TDBE bit. The reset value of this bit is 1, indicating that the SPI\_DT register is empty. If the TDBEIE bit is set, an interrupt is generated. After the data is written, the TDBE is pulled low until the data is moved to the transmit data buffer before the TDBE is set once again. This means that the user can be allowed to write the data to be transmitted only when the TDBE is set.

After the transmitter is configured and the SPI is enabled, the SPI is ready for data transmission. Before going forward, it is necessary for the users to refer to full-duplex / half-duplex chapter to get detailed configuration information, go to the Chip select controller chapter for specific chip select mode, check the SPI\_SCK controller chapter for information on communication clock, and refer to CRC and DMA transfer chapter to configure CRC and DMA (if necessary). The recommended configuration procedure are as follows.

### Transmitter configuration procedure:

- Configure full-duplex/half-duplex selector
- Configure chip select controller
- Configure SPI\_SCK controller
- Configure CRC (if necessary)
- Configure DMA transfer (if necessary)
- If the DMA transfer mode is not used, the software will check whether to enable transmit data

- interrupt (RDBEIE = 1) through the RDBE bit.
- Configure frame format: select MSB/LSB mode with the LTF bit, and select 8/16-bit data with the FBN bit
- Enable SPI by setting the SPIEN

## Receiver

The SPI receiver is clocked by the SPI\_SCK controller. It can output different data frame formats through software configuration. There is a receive data buffer register, driven by the SPI\_SCK, in the SPI receiver.

At the last CLK of each transfer, the data is moved from the shift register to the receive data buffer register. Then the transmitter sets the receive data complete flag to the SPI logic. When the flag is detected by the SPI logic, the data in the receive data buffer is copied into the SPI\_DT register, with the RDBF being set. This means that the data is received, and it is already stored into the SPI\_DT. In this case, read access to the SPI\_DT register will clear the RDBF bit.

Both DMA and CPU can be used for read operation. For DMA transfer, refer to DMA transfer section for more details. For CPU transfer, attention should be paid to the RDBE bit. The reset value of this bit is 0, indicating that the SPI\_DT register is empty. If the data is received and moved into the SPI\_DT, the RDBF is set, meaning that there are some data to be read in the SPI\_DT register. An interrupt is generated if the RDBFIE bit is set.

When the next received data is ready to be moved to the SPI\_DT register, if the previous received data is still not read (RDBF = 1), then the data overflow occurs. The previous receive data is not lost, but the next received data will do. At this point, the ROERR is set. An interrupt is generated if the ERRIE is set. Read SPI\_DT register and then the SPI\_STS register will clear the ROERR bit. The recommended configuration procedure is as follows.

### Receiver configuration procedure:

- Configure full-duplex/half-duplex selector
- Configure chip select controller
- Configure SPI\_SCK controller
- Configure CRC (if necessary)
- Configure DMA transfer (if necessary)
- If the DMA transfer mode is not used, the software will check whether to enable receive data interrupt (RDBEIE = 1) through the RDBE bit.
- Configure frame format: select MSB/LSB mode with the LTF bit, and select 8/16-bit data with the FBN bit
- Enable SPI by setting the SPIEN

## Motorola mode

This section describes the SPI communication timings, which includes full-duplex and half-duplex master/slave timings.



### Full-duplex communication – master mode

Configured as follows:

MSTEN = 1: Master enable

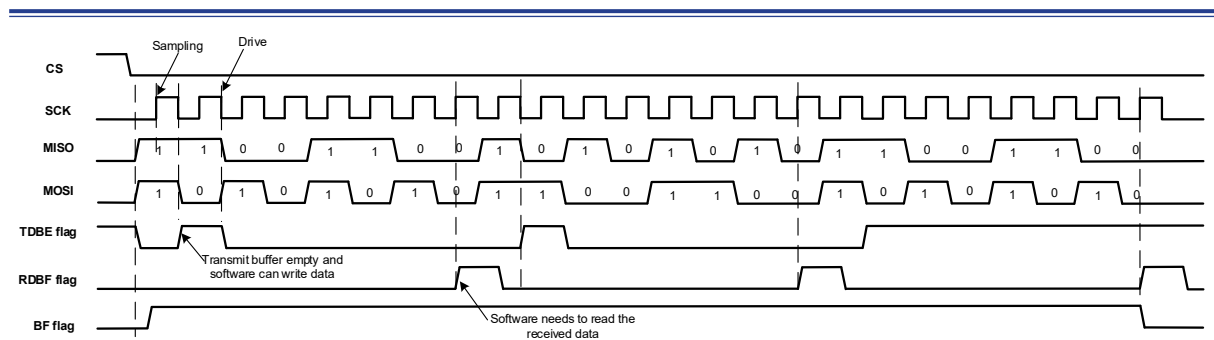
SLBEN = 0: Full-duplex mode

CLKPOL = 0, CLKPHA = 0: SCK idle output low, use the first edge for sampling

FBN = 0: 8-bit frame

Master transmit (MOSI): 0xaa, 0xcc, 0xaa

Slave transmit (MISO): 0xcc, 0xaa, 0xcc



**Figure 68. Master full-duplex communications**

### Full-duplex communication – slave mode

Configured as follows:

MSTEN = 0: Slave enable

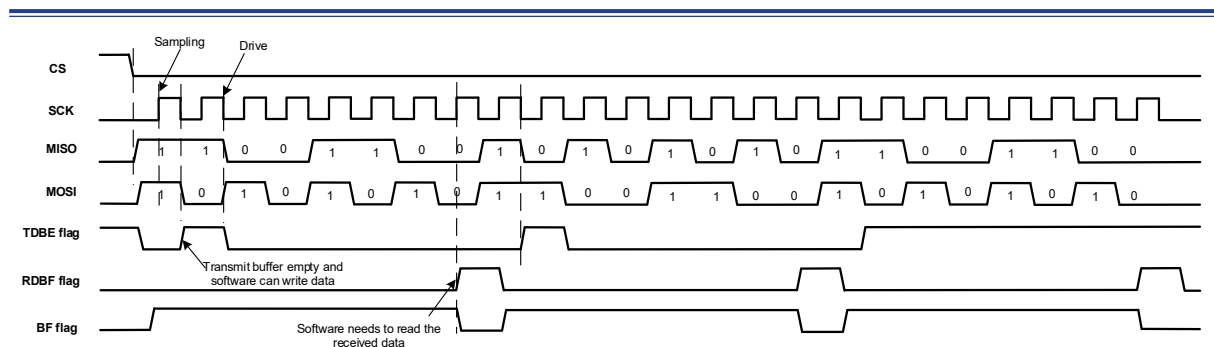
SLBEN = 0: Full-duplex mode

CLKPOL = 0, CLKPHA = 0: SCK idle output low, use the first edge for sampling

FBN = 0: 8-bit frame

Master transmit (MOSI): 0xaa, 0xcc, 0xaa

Slave transmit (MISO): 0xcc, 0xaa, 0xcc



**Figure 69. Slave full-duplex communications**

### Half-duplex communication – master transmit

Configured as follows:

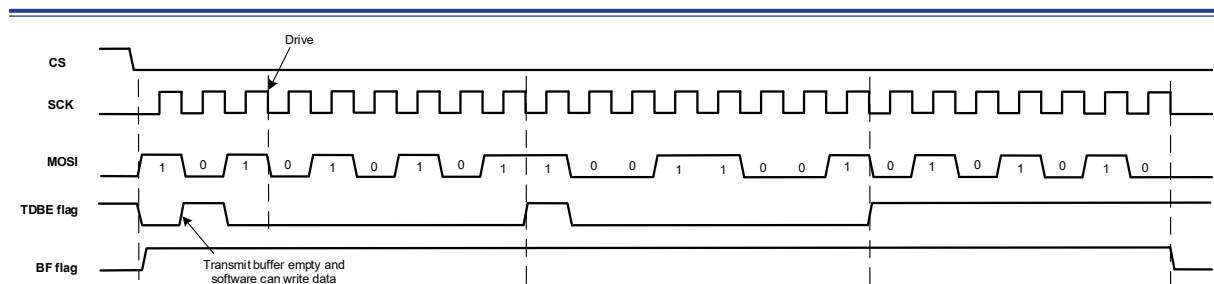
MSTEN = 1: Master enable

SLBEN = 1: Single line bidirectional mode

CLKPOL = 0, CLKPHA = 0: SCK idle output low, use the first edge for sampling

FBN = 0: 8-bit frame

Master transmit (MOSI): 0xaa, 0xcc, 0xaa



**Figure 70. Master half-duplex transmit**

### Half-duplex communication – slave receive

Configured as follows:

MSTEN = 0: Slave enable

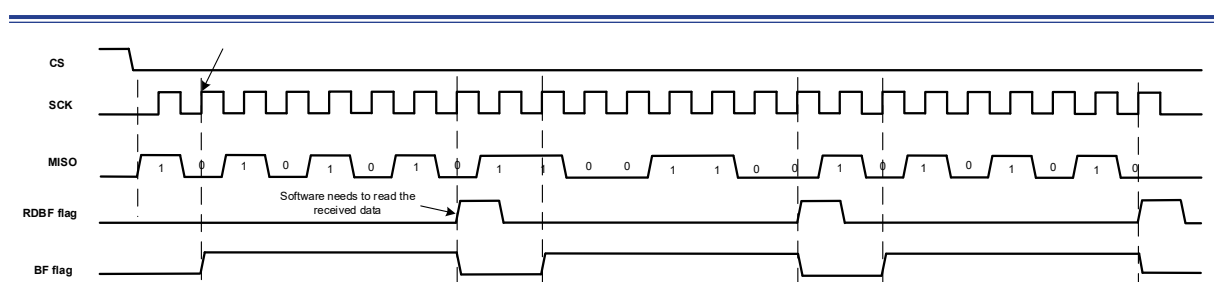
SLBEN = 1: Single line bidirectional mode

SLBTD = 0: Receive mode

CLKPOL = 0, CLKPHA = 0: SCK idle output low, use the first edge for sampling

FBN = 0: 8-bit frame

Slave receive: 0xaa, 0xcc, 0xaa



**Figure 71. Slave half-duplex receive**

### Half-duplex communication – slave transmit

Configured as follows:

MSTEN = 0: Slave enable

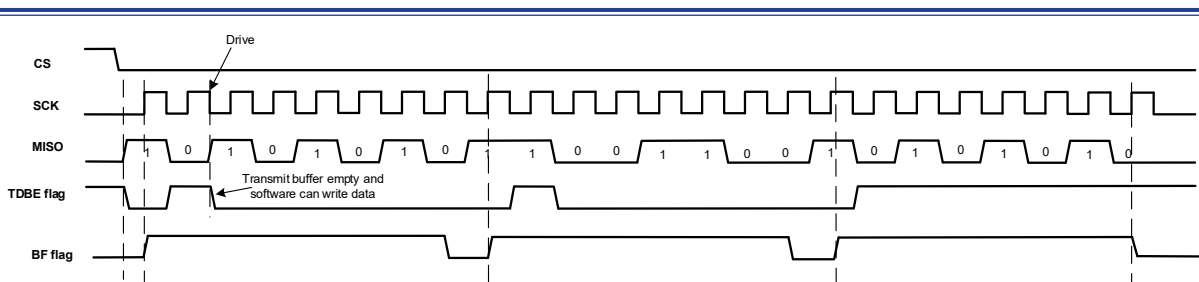
SLBEN = 1: Single line bidirectional mode

SLBTD = 1: Transmit enable

CLKPOL = 0, CLKPHA = 0: SCK idle output low, use the first edge for sampling

FBN = 0: 8-bit frame

Slave transmit: 0xaa, 0xcc, 0xaa



**Figure 72. Slave half-duplex transmit**

### Half-duplex communication – master receive

Configured as follows:

MSTEN = 1: Master enable

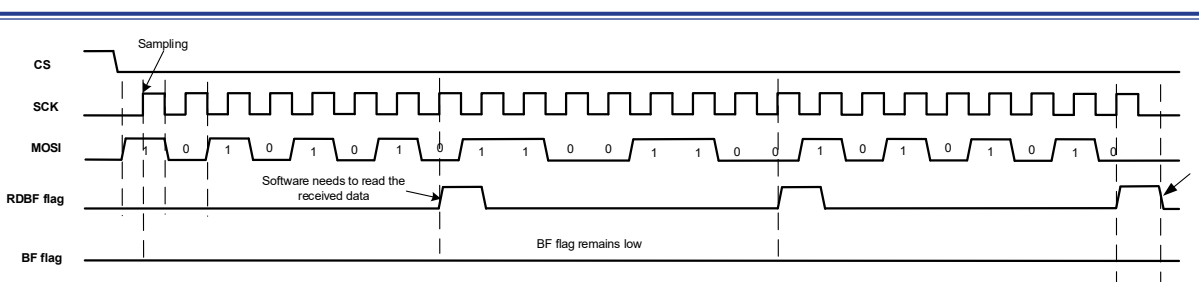
SLBEN = 1: Single line bidirectional mode

SLBTD = 0: Receive enable

CLKPOL = 0, CLKPHA = 0: SCK idle output low, use the first edge for sampling

FBN = 0: 8-bit frame

Master receive: 0xaa, 0xcc, 0xaa

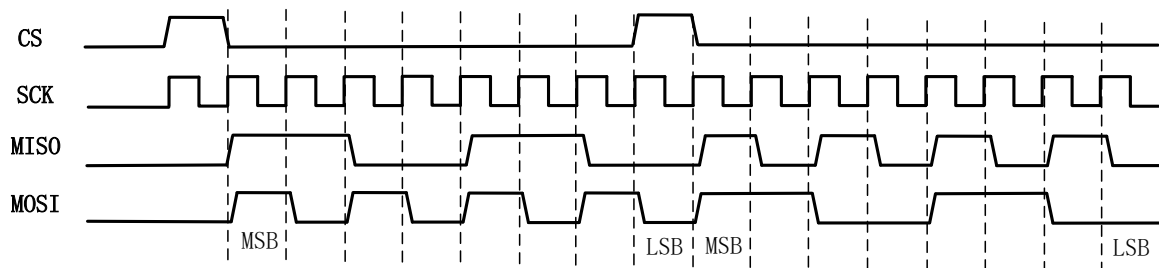


**Figure 73. Master half-duplex receive**

## TI mode

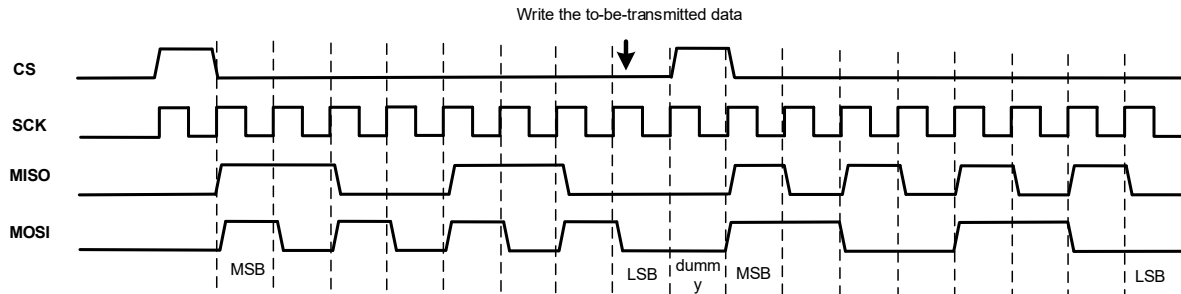
The SPI interface supports TI mode. The TIEN bit can be set to enable SPI TI mode.

In TI mode, a bit of different is present between continuous and discontinuous communication timings. When the to-be transmitted data is written before the rising SCK edge corresponding to the last data of the current transmit frame, it is a continuous communication, without dummy CLK between data, and the host sends a valid CS pulse while transmitting the last data of the current frame.



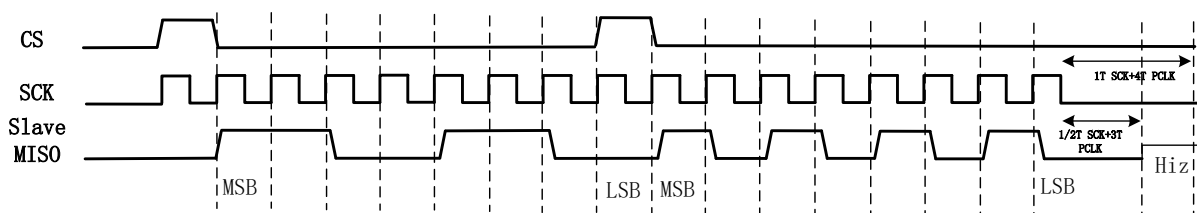
**Figure 74. TI mode continous transfer**

When the to-be-transmitted data is written between the rising and falling SCK edge corresponding to the last data of the current transmit frame, a dummy CLK exists between data.



**Figure 75. TI mode continous transfer with dummy CLK**

When the to-be-transmitted data is written after the falling SCK edge corresponding to the last data of the current transmit frame, the host always issues a valid SCK clock after  $1T_{SCK} + 4T_{PCLK}$ . If the slave still does not detect a valid CS pulse at the end of the current data reception, it disables MISO output after  $1/2T_{SCK} + 3T_{PCLK}$  to control MISO floating.



**Figure 76. TI mode continous transfer with dummy CLK**

## Interrupts

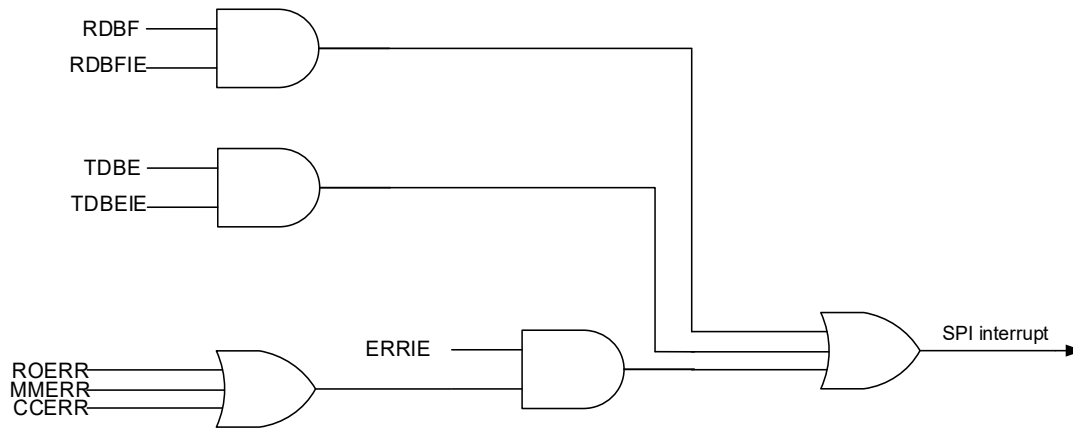


Figure 77. SPI interrupts

## IO pin control

Usually, the SPI is connected to external devices through four pins.

- MISO: Master In/Slave Out. The pin receives data in master mode, and transmits data in slave mode.
- MOSI: Master Out/Slave In. The pin transmits data in master mode, and receives data in slave mode.
- SCK: SPI communication clock. The pin serves as output in master mode, and input in slave mode.
- CS: Chip Select. This is an optional pin which selects master/slave mode.

## Precautions

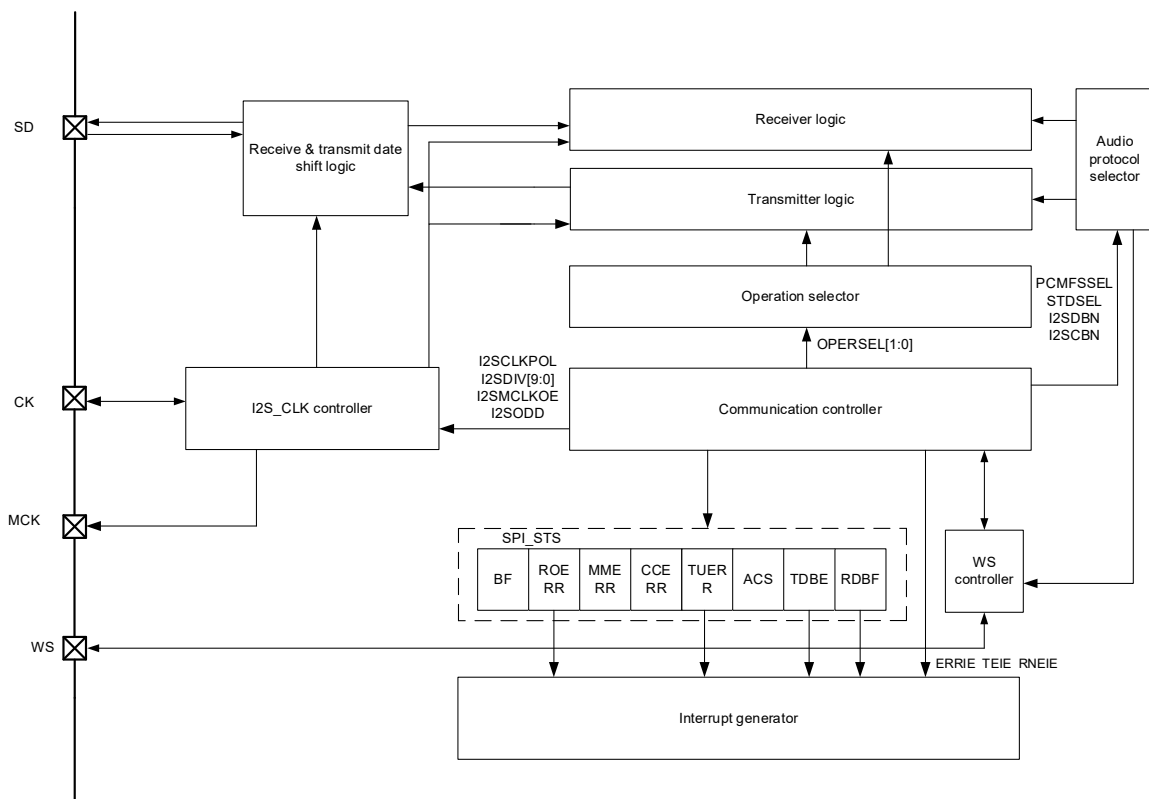
- CRC value is obtained by software reading DT register at the end of CRC reception
- In the case of CPOL = 1 and CPHA = 1, the clock divided by 3 that is generated inside the SPI must be less than 32 MHz. To achieve a greater communication frequency, it is necessary to use a clock divided by 2, and adjust the corresponding HCLK and PCLK frequencies. The SPI frequencies must not exceed the maximum value programmed in the corresponding datasheet.

## I<sup>2</sup>S functional description

### I<sup>2</sup>S introduction

The I<sup>2</sup>S can be configured by software as master reception/transmission, and slave reception/transmission, supporting four kinds of audio protocols including Philips standard, MSB-aligned standard, LSB-aligned standard and PCM standard, respectively. The DMA transfer is also supported.

A single I<sup>2</sup>S supports half-duplex. However, it can work with two additional instantiated I<sup>2</sup>S modules (I<sup>2</sup>S2EXT and I<sup>2</sup>S3EXT) to achieve full-duplex mode. In other words, combining the I<sup>2</sup>S2 with the I<sup>2</sup>S2EXT enables the I<sup>2</sup>S2 to support full-duplex mode. This is true for the I<sup>2</sup>S3 through the combination of the I<sup>2</sup>S3 with the I<sup>2</sup>S3EXT. Refer to I<sup>2</sup>S full-duplex section for more information.



**Figure 78. I²S block diagram**

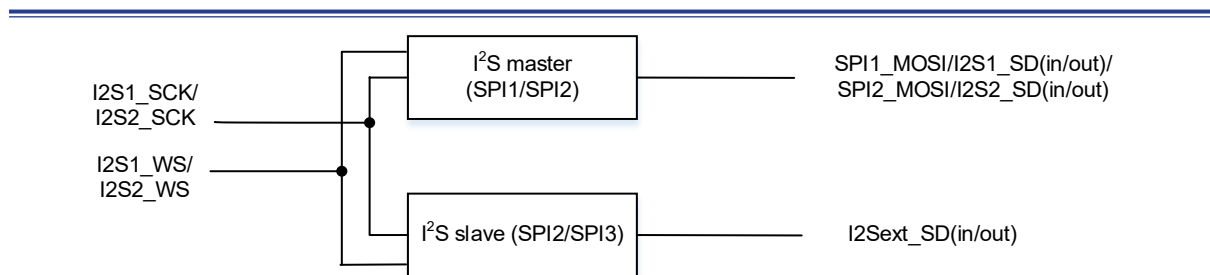
#### Main features when the SPI is used as I²S:

- Programmable operation mode
  - Slave device transmission
  - Slave device reception
  - Master device transmission
  - Master device reception
- Programmable clock polarity
- Programmable clock frequency (8 KHz to 192 KHz)
- Programmable data bits (16 bit, 24 bit, 32 bit)
- Programmable channel bits (16 bit, 32 bit)
- Programmable audio protocol
  - I²S Philips standard
  - MSB-aligned standard (left-aligned)
  - LSB-aligned standard (right-aligned)
  - PCM standard (long or short frame)
- I²S full-duplex

- DMA transfer
- Main peripheral clock with a fixed frequency of 256x Fs (audio sampling frequency)

## I<sup>2</sup>S full-duplex

Two SPIs can be combined to support I<sup>2</sup>S full-duplex mode through the SCFG\_CFG2[31:30] bit in the SCFG register. Of the three SPIs, the SPI1 or SPI2 can be configured as full-duplex master, while the SPI2 or SPI3 can be set as full-duplex slave, which is selected through the SCFG\_CFG2[31:30] bit in the SCFG register. Once selected, the IO remap relations of the master remains unchanged, and the SCK and WS of the slave are connected to the SCK and WS of the master internally, with the SD line of the slave remapped onto the I2S\_SDEXT. The slave's original IO remap relations become invalid, releasing the corresponding IOs.



**Figure 79. I<sup>2</sup>S full-duplex structure**

### I<sup>2</sup>S full-duplex master side:

It supports master or slave mode. It can be programmed as a receiver or transmitter.

- I2Sx\_WS takes part in communication for actual WS signal interaction
- I2Sx\_SCK takes part in communication for actual clock signal interaction
- I2Sx\_SD takes part in communication for data and information interaction of the master side

### I<sup>2</sup>S full-duplex slave side

It supports slave mode only. It can be programmed as a transmitter or receiver.

- I2Sy\_WS does not take part in communication, releasing the corresponding IOs
- I2Sy\_SCK does not take part in communication, releasing the corresponding IOs
- I2Sy\_SD does not take part in communication, releasing the corresponding IOs
- I2S\_SDEXT takes part in communication for data and information interaction of the slave side

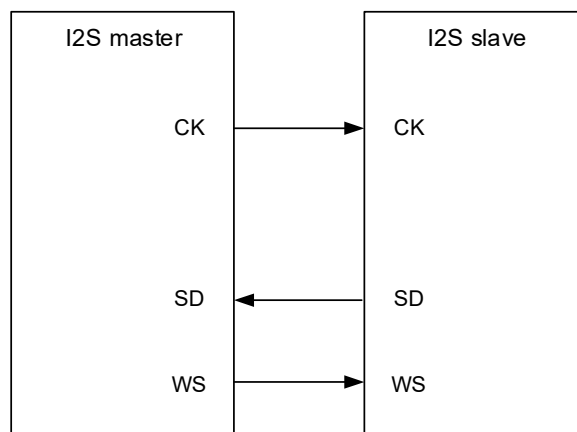
Note: x can be either 1 or 2, whereas y can be either 2 or 3.

## Operating mode selector

The SPI, used as I<sup>2</sup>S selector, offers multiple operating modes for selection, namely, slave device transmission, slave device reception, master device transmission and master device reception. This is done by software configuration.

### Slave device transmission:

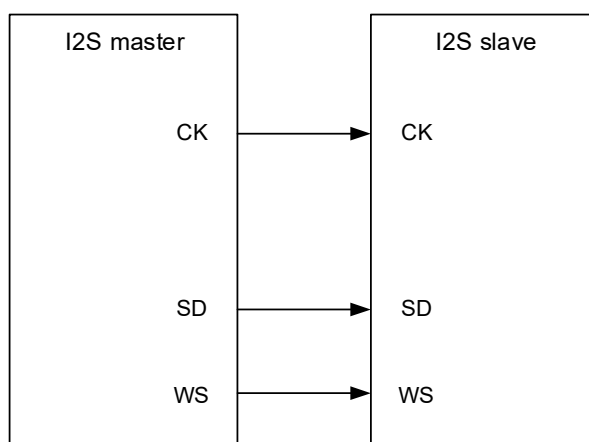
Set the I2SMSEL bit, and OPERSEL[1:0] = 00, the I<sup>2</sup>S will work in slave device transmission mode.



**Figure 80. I<sup>2</sup>S slave device transmission**

### Slave device reception:

Set the I2SMSEL bit, and OPERSEL[1:0] = 01, the I<sup>2</sup>S will work in slave device reception mode.

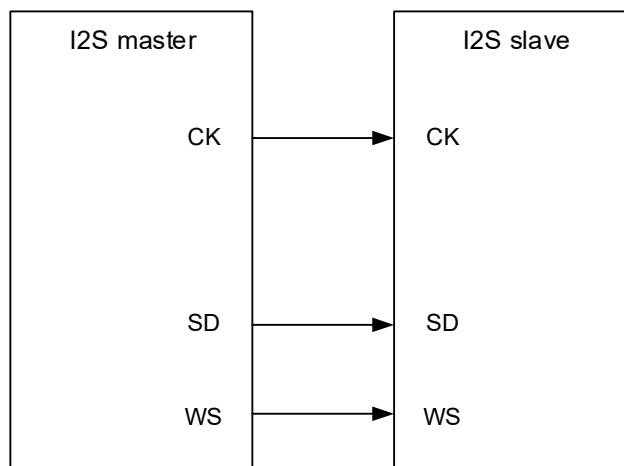


**Figure 81. I<sup>2</sup>S slave device reception**



#### Master device transmission:

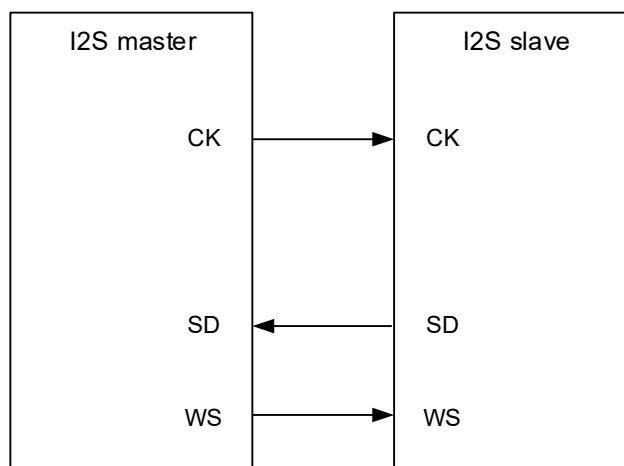
Set the I2SMSEL bit, and OPERSEL[1:0] = 10, the I<sup>2</sup>S will work in master device transmission mode.



**Figure 82. I<sup>2</sup>S master device transmission**

#### Master device reception:

Set the I2SMSEL bit, and OPERSEL[1:0] = 11, the I<sup>2</sup>S will work in master device reception mode.



**Figure 83. I<sup>2</sup>S master device reception**

### Audio protocol selector

While being used as I<sup>2</sup>S, the SPI supports multiple audio protocols. The user can control the audio protocol selector through software configuration to select the desired audio protocol, with the data bits and channel bits being controlled by the audio protocol selector. Besides, the user can also select the data bits and channel bits through software configuration. Meanwhile, the audio protocol selector manages the WS controller, output or detect the WS signal that meets the protocol requirements.

- Select audio protocol by setting the STDSEL bit
  - STDSELE = 00: Philips standard
  - STDSELE = 01: MSB-aligned standard (left-aligned)
  - STDSELE = 10: LSB-aligned standard (right-aligned)
  - STDSELE = 11: PCM standard
- Select PCM frame synchronization format: PCMFSSSEL = 1 for PCM long frame synchronization, PCMFSSSEL = 0 for short frame synchronization (this step is required when selecting PCM protocol)
- Select data bits by setting the I2SDBN bit
  - I2SDBN = 00: 16-bit
  - I2SDBN = 01: 24-bit
  - I2SDBN = 10: 32-bit
- Select channel bits by setting the I2SCBN bit
  - I2SDBN = 0: 16-bit
  - I2SDBN = 1: 32-bit

Note: Read/Write operation mode depends on the selected audio protocols, data bits and channel bits. The following lists all possible configuration combinations and their respective read and write operation mode.
- Philips standard, PCM standard, MSB-aligned or LSB-aligned standard, 16-bit data and 16-bit channel

The data bit is the same as the channel bit. Each channel requires one read/write operation from/to the SPI\_DT register, and the number of DMA transfer is 1.
- Philips standard, PCM standard or MSB-aligned standard, 16-bit data and 32-bit channel
- The data bit is different from the channel bit. Each channel requires one read/write operation from/to the SPI\_DT register, and the number of DMA transfer is 1. The first 16 bits (MSB) are the significant bits, and the 16-bit LSB is forced to 0 by hardware.
- Philips standard, PCM standard or MSB-aligned standard, 24-bit data and 32-bit channel

The data bit is different from the channel bit. Each channel requires two read/write operations from/to the SPI\_DT register, and the number of DMA transfer is 2. The 16-bit MSB transmits and receives the first 16-bit data, the 16-bit LSB transmits and receives the 8-bit MSB data, with 8-bit LSB data being forced to 0 by hardware.
- Philips standard, PCM standard, MSB-aligned or LSB-aligned standard, 32-bit data and 32-bit channel

The data bit is the same as the channel bit. Each channel requires two read/write operations from/to the SPI\_DT register, and the number of DMA transfer is 2. These 32-bit data are proceeded in two times, with 16-bit data each time.
- LSB-aligned standard, 16-bit data and 32-bit channel

The data bit is different from the channel bit. Each channel requires one read/write operation from/to the SPI\_DT register, and the number of DMA transfer is 1. The 16 bits (LSB) are the significant bits while the first 16-bit data (MSB) are forced to 0 by hardware.

■ LSB-aligned standard, 24-bit data and 32-bit channel

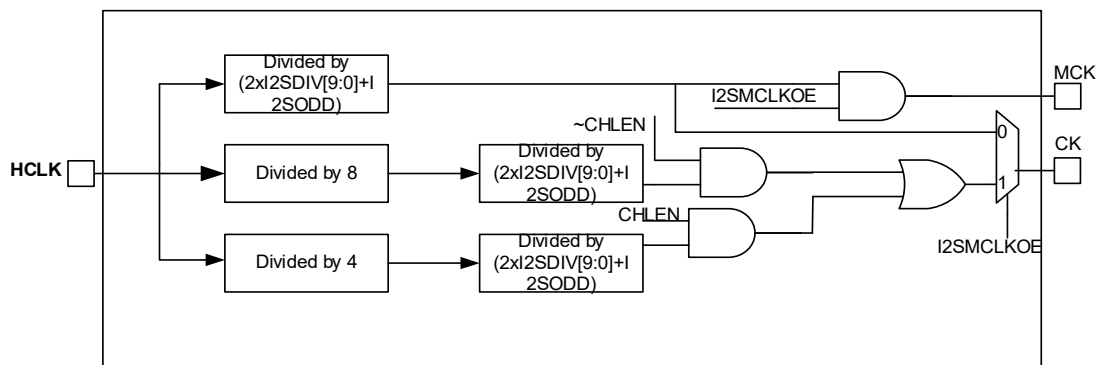
The data bit is different from the channel bit. Each channel requires two read/write operations from/to the SPI\_DT register, and the number of DMA transfer is 2. For the first 16-bit data, its 8-bit LSB are the significant bits, with the 8-bit MSB forced to 0 by hardware; the subsequent 16 bits transmit and receive the second 16-bit data.

## I2S\_CLK controller

The audio protocols the SPI supports adopts synchronous transmission. In master mode, it is required to generate a communication clock for data reception and transmission on the SPI, and the communication clock should be output to the slave via IO for data reception and transmission. In slave mode, the communication clock is provided by master, and is input to the SPI via IO. In all, the I2S\_SCK controller is used for the generation and distribution of I2S\_SCK, with the configuration procedure detailed as follows:

When used as I<sup>2</sup>S master, the SPI can provide communication clock (CK) and main peripheral clock (MCK) shown in Figure 84. The CK and MCK are generated by HCLK divider, with the prescaler of the MCK determined by I2SDIV and I2SODD. The calculation formula is seen in Figure 84.

The prescaler of the CK depends on whether to provide the main clock for peripherals. To ensure that the main clock is always 256 times larger than the audio sampling frequency, The channel bits should be taken into account. When the main clock is needed, the CK should be divided by 8 (I2SCBN = 0) or 4 (I2SCBN = 1), then divided again by the same prescaler as that of the MCK, that is the final communication clock; When the main clock is not needed, the prescaler of the CK is determined by I2SDIV and I2SODD, shown in Figure 84.



**Figure 84. CK & MCK source in master mode**

Apart from the above-mentioned configuration, the following table lists the values of I2SDIV and I2SODD corresponding to some specific frequencies, as well as their respective error for the users to configure the I2SDIV and I2SODD.

**Table 44. Audio frequency precision using system clock**

| SCLK (MHz) | MCLK | TargetFs (Hz) | 16bit  |         |          |       | 32bit  |         |          |       |
|------------|------|---------------|--------|---------|----------|-------|--------|---------|----------|-------|
|            |      |               | I2SDIV | I2S_ODD | RealFs   | Error | I2SDIV | I2S_ODD | RealFs   | Error |
| 72         | No   | 192000        | 6      | 0       | 187500   | 2.34% | 3      | 0       | 187500   | 2.34% |
| 72         | No   | 96000         | 11     | 1       | 97826.09 | 1.90% | 6      | 0       | 93750    | 2.34% |
| 72         | No   | 44100         | 25     | 1       | 44117.65 | 0.04% | 13     | 0       | 43269.23 | 1.88% |
| 72         | No   | 32000         | 35     | 0       | 32142.86 | 0.45% | 17     | 1       | 32142.86 | 0.45% |
| 72         | No   | 22050         | 51     | 0       | 22058.82 | 0.04% | 25     | 1       | 22058.82 | 0.04% |

| SCLK<br>(MHz) | MCLK | TargetFs<br>(Hz) | 16bit  |         |          |       | 32bit  |         |          |       |
|---------------|------|------------------|--------|---------|----------|-------|--------|---------|----------|-------|
|               |      |                  | I2SDIV | I2S_ODD | RealFs   | Error | I2SDIV | I2S_ODD | RealFs   | Error |
| 72            | No   | 16000            | 70     | 1       | 15957.45 | 0.27% | 35     | 0       | 16071.43 | 0.45% |
| 72            | No   | 11025            | 102    | 0       | 11029.41 | 0.04% | 51     | 0       | 11029.41 | 0.04% |
| 72            | No   | 8000             | 140    | 1       | 8007.117 | 0.09% | 70     | 1       | 7978.723 | 0.27% |
| 72            | Yes  | 48000            | 3      | 0       | 46875    | 2.34% | 3      | 0       | 46875    | 2.34% |
| 72            | Yes  | 44100            | 3      | 0       | 46875    | 6.29% | 3      | 0       | 46875    | 6.29% |
| 72            | Yes  | 32000            | 4      | 1       | 31250    | 2.34% | 4      | 1       | 31250    | 2.34% |
| 72            | Yes  | 22050            | 6      | 1       | 21634.62 | 1.88% | 6      | 1       | 21634.62 | 1.88% |
| 72            | Yes  | 16000            | 9      | 0       | 15625    | 2.34% | 9      | 0       | 15625    | 2.34% |
| 72            | Yes  | 11025            | 13     | 0       | 10817.31 | 1.88% | 13     | 0       | 10817.31 | 1.88% |
| 72            | Yes  | 8000             | 17     | 1       | 8035.714 | 0.45% | 17     | 1       | 8035.714 | 0.45% |

## DMA transfer

The SPI supports write and read operations with DMA. Whether used as SPI or I<sup>2</sup>S, read/write request using DMA comes from the same peripheral. As a result, their configuration procedure are the same, described as follows.

### Transmission with DMA

- Select DMA channel: Select a DMA channel for the current SPI from DMA channel map table described in DMA chapter.
- Configure the destination of DMA transfer: Configure the SPI\_DT register address as the destination address bit of DMA transfer in the DMA control register. Data will be sent to this address after transmit request is received by DMA.
- Configure the source of DMA transfer: Configure the memory address as the source of DMA transfer in the DMA control register. Data will be loaded into the SPI\_DT register from the memory address after transmit request is received by DMA.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure the channel priority of DMA transfer in the DMA control register.
- Configure DMA interrupt generation after half or full transfer in the DMA control register.
- Enable DMA transfer channel in the DMA control register.

### Reception with DMA

- Select DMA transfer channel: Select a DMA channel for the current SPI from DMA channel map table described in DMA chapter.
- Configure the destination of DMA transfer: Configure the memory address as the destination of DMA transfer in the DMA control register. Data will be loaded from the SPI\_DT register to the programmed destination after reception request is received by DMA.
- Configure the source of DMA transfer: Configure the SPI\_DT register address as the source of DMA transfer in the DMA control register. Data will be loaded from the SPI\_DT register to the programmed destination after reception request is received by DMA.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure DMA interrupt generation after half or full transfer in the DMA control register
- Enable DMA transfer channel in the DMA control register.

## Transmitter/Receiver

Whether being used as SPI or I<sup>2</sup>S, there is no difference for CPU. The SPI (in whatever mode) shares the same base address, the same SPI\_DT register, the same transmitter and receiver. The SPI transmitter and receiver is responsible for sending and receiving the desired data frame according to the configuration of the communication controller. Thus their status flags such as TDBE, RDBF and ROERR, and their interrupt enable bits including TDBEIE, RDBFIE and ERRIE are identical.

### Special attention must be paid to:

- CRC check is not available on the I<sup>2</sup>S. Any operation linked to CRC, including CCERR flag and the corresponding interrupts, is not supported.
- I<sup>2</sup>S protocol needs decode the current channel status. The ACS bit is used to judge whether the current transfer occurs on the left channel (ACS = 0) or the right channel (ACS = 1).
- TUERR bit indicates whether an underrun occurs. TUERR = 1 means an underrun error occurs on the transmitter. An interrupt is generated when the ERRIE is set.
- Read/write operation to the SPI\_DT register is different under different audio protocols, data bits and channel bits. Refer to the audio protocol selector section for more information.
- Pay more attention to the I<sup>2</sup>S disable operation under different configurations, shown as follows:
  - I2SDBN = 00, I2SCBN = 1, STDSLE = 10: wait for the second-to-last RDBF = 1 and 17 CK periods before disabling the I<sup>2</sup>S.
  - I2SDBN = 00, I2SCBN = 1, STDSLE = 00 or STDSLE = 01 or STDSLE = 11: wait for the last RDBF = 1 and one CK period before the I<sup>2</sup>S.
  - I2SDBN, I2SCBN, STDSLE combination: wait for the second-to-last RDBF = 1 and one CK period before disabling the I<sup>2</sup>S.

### I<sup>2</sup>S transmitter configuration procedure:

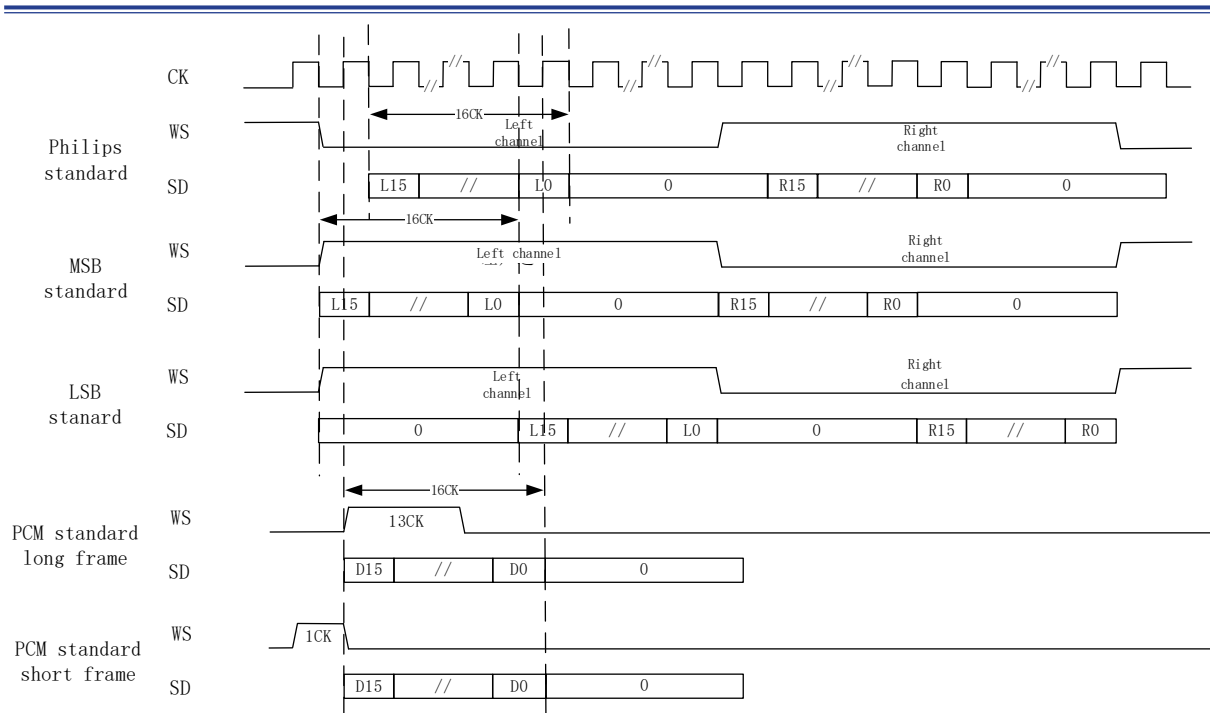
- Configure operation mode selector
- Configure audio protocol selector
- Configure I2S\_SCK controller
- Configure DMA transfer (if necessary)
- Set the I2SEN bit to enable I<sup>2</sup>S
- Follow above steps to configure the I<sup>2</sup>SxEXT (For I<sup>2</sup>S full-duplex mode )

### I<sup>2</sup>S receiver configuration procedure:

- Configure operation mode selector
- Configure audio protocol selector
- Configure I2S\_SCK controller
- Configure DMA transfer (if necessary)
- Set the I2SEN bit to enable I<sup>2</sup>S
- Follow above steps to configure the I<sup>2</sup>SxEXT (For I<sup>2</sup>S full-duplex mode )

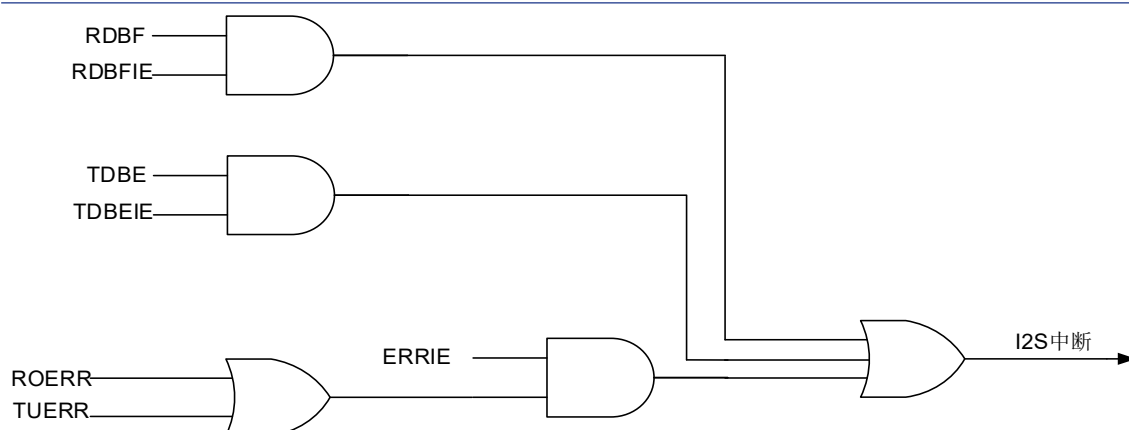
## I<sup>2</sup>S communication timings

I<sup>2</sup>S can address four different audio standards: Philips standard, the most significant byte (left-aligned) and the least significant byte (right-aligned) standards, and the PCM standard. Figure 85 shows their respective timings.



**Figure 85. Audio standard timings**

## Interrupts



**Figure 86. I<sup>2</sup>S interrupts**

## IO pin control

The I<sup>2</sup>S needs three pins for transfer operation, namely, the SD, WS and CK. The MCLK pin is also required if need to provide main clock for peripherals. The I<sup>2</sup>S shares some pins with the SPI, described as follows:

- SD: Serial data (mapped on the MOSI pin) for bidirectional data transmission and reception.
- WS: Word select (mapped on the CS pin) for data control signal output in master mode, and input in slave mode.
- CK: Communication clock (mapped on the SCK pin) as clock signal output in master mode, and input in slave mode.
- MCLK: Master clock (mapped independently) is used to provide main clock for peripherals. The frequency of output clock signal is set to 256x Fs (audio sampling frequency)

## SPI registers

These peripheral registers must be accessed by half-word (16 bits) or word (32 bits).

**Table 45. SPI register map and reset value**

| Register    | Offset | Reset value |
|-------------|--------|-------------|
| SPI_CTRL1   | 0x00   | 0x0000      |
| SPI_CTRL2   | 0x04   | 0x0000      |
| SPI_STS     | 0x08   | 0x0002      |
| SPI_DT      | 0x0C   | 0x0000      |
| SPI_CPOLY   | 0x10   | 0x0007      |
| SPI_RCRC    | 0x14   | 0x0000      |
| SPI_TCRC    | 0x18   | 0x0000      |
| SPI_I2SCTRL | 0x1C   | 0x0000      |
| SPI_I2SCLKP | 0x20   | 0x0002      |

### SPI control register 1 (SPI\_CTRL1) (not used in I<sup>2</sup>S mode)

| Bit    | Name  | Reset value | Type | Description   |
|--------|-------|-------------|------|---|
| Bit 15 | SLBEN | 0x0         | rw   | Single line bidirectional half-duplex enable<br>0: Disabled<br>1: Enabled   |
| Bit 14 | SLBTD | 0x0         | rw   | Single line bidirectional half-duplex transmission direction<br>This bit and the SLBEN bit together determine the data output direction in "Single line bidirectional half-duplex" mode.<br>0: Receive-only mode<br>1: Transmit-only mode |
| Bit 13 | CCEN  | 0x0         | rw   | RC calculation enable<br>0: Disabled<br>1: Enabled  |
| Bit 12 | NTC   | 0x0         | rw   | Transmit CRC next<br>When this bit is set, it indicates that the next data transferred is CRC value.<br>0: Next transmitted data is the normal value<br>1: Next transmitted data is CRC value   |

| Bit     | Name   | Reset value | Type | Description   |
|---------|--------|-------------|------|---|
| Bit 11  | FBN    | 0x0         | rw   | Frame bit num<br>This bit is used to configure the number of data frame bit for transmission/reception.<br>0: 8-bit data frame<br>1: 16-bit data frame  |
| Bit 10  | ORA    | 0x0         | rw   | Receive-only active<br>In two-wire unidirectional mode, when this bit is set, it indicates that Receive-only is active, but the transmit is not allowed.<br>0: Transmission and reception<br>1: Receive-only mode   |
| Bit 9   | SWCSEN | 0x0         | rw   | Software CS enable<br>When this bit is set, the CS pin level is determined by the SWCSIL bit. The status of I/O level on the CK pin is invalid.<br>0: Disabled<br>1: Enabled  |
| Bit 8   | SWCSIL | 0x0         | rw   | Software CS internal level<br>This bit is valid only when the SWCSEN is set. It determines the level on the CS pin.<br>In master mode, this bit must be set.<br>0: Low level<br>1: High level   |
| Bit 7   | LTF    | 0x0         | rw   | LSB transmit first<br>This bit is used to select for MST transfer first or LSB transfer first.<br>0: MSB<br>1: LSB  |
| Bit 6   | SPIEN  | 0x0         | rw   | SPI enable<br>0: Disabled<br>1: Enabled   |
| Bit 5:3 | MDIV   | 0x0         | rw   | Master clock frequency division<br>In master mode, the peripheral clock divided by the prescaler is used as SPI clock. The MDIV[3] bit is in the SPI_CTRL2 register, MDIV[3:0]:<br>0000: Divided by 2<br>0001: Divided by 4<br>0010: Divided by 8<br>0011: Divided by 16<br>0100: Divided by 32<br>0101: Divided by 64<br>0110: Divided by 128<br>0111: Divided by 256<br>1000: Divided by 512<br>1001: Divided by 1024 |
| Bit 2   | MSTEN  | 0x0         | rw   | Master enable<br>0: Disabled (Slave)<br>1: Enabled (Master)   |
| Bit 1   | CLKPOL | 0x0         | rw   | Clock polarity<br>Indicates the polarity of clock output in idle state.<br>0: Low level<br>1: High level  |
| Bit 0   | CLKPHA | 0x0         | rw   | Clock phase<br>0: Data capture starts from the first clock edge<br>1: Data capture starts from the second clock edge  |



Note: The SPI\_CTRL1 register must be 0 in I<sup>2</sup>S mode.

### SPI control register 2 (SPI\_CTRL2)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15:10 | Reserved | 0x00        | resd | Forced 0 by hardware.   |
| Bit 9     | MDIV3EN  | 0x0         | rw   | Master clock frequency divided by 3 enable<br>0: Disabled<br>1: Enabled<br><br>Note: When this bit is set, the MDIV[3:0] becomes invalid, and the SPI clock is forced to be PCLK/3.   |
| Bit 8     | MDIV[3]  | 0x0         | rw   | Master clock frequency division<br>Refer to the MDIV[2:0] of the SPI_CTRL1 register.  |
| Bit 7     | TDBEIE   | 0x0         | rw   | Transmit data buffer empty interrupt enable<br>0: Disabled<br>1: Enabled  |
| Bit 6     | RDBFIE   | 0x0         | rw   | Receive data buffer full interrupt enable<br>0: Disabled<br>1: Enabled  |
| Bit 5     | ERRIE    | 0x0         | rw   | Error interrupt enable<br>This bit controls interrupt generation when errors occur (CCERR, MMERR, ROERR and TUERR)<br>0: Disabled<br>1: Enabled   |
| Bit 4     | TIEN     | 0x0         | rw   | TI mode enable<br>0: TI mode disabled (Motorola mode)<br>1: TI mode enabled (TI mode)<br><br>Note: This mode is not used in I <sup>2</sup> S mode. It must be 0 in I <sup>2</sup> S mode.                                     |
| Bit 3     | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 2     | HWCSOE   | 0x0         | rw   | Hardware CS output enable<br>This bit is valid only in master mode. When this bit is set, the I/O output on the CS pin is low; when this bit is 0, the I/O input on the CS pin must be set high.<br>0: Disabled<br>1: Enabled |
| Bit 1     | DMATEN   | 0x0         | rw   | DMA transmit enable<br>0: Disabled<br>1: Enabled  |
| Bit 0     | DMAREN   | 0x0         | rw   | DMA receive enable<br>0: Disabled<br>1: Enabled   |

## SPI status register (SPI\_STS)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:9 | Reserved | 0x00        | resd | Forced 0 by hardware   |
| Bit 8    | CSPAS    | 0x0         | ro   | CS pulse abnormal setting flag<br>0: CS pulse flag normal<br>1: CS pulse flag is set abnormally<br>Note: This bit is used for TI slave mode. It is cleared by reading the STS register.                                  |
| Bit 7    | BF       | 0x0         | ro   | Busy flag<br>0: SPI is not busy.<br>1: SPI is busy.  |
| Bit 6    | ROERR    | 0x0         | ro   | Receiver overflow error<br>0: No overflow error<br>1: Overflow error occurs.   |
| Bit 5    | MMERR    | 0x0         | ro   | Master mode error<br>This bit is set by hardware and cleared by software (read/write access to the SPI_STS register, followed by write operation to the SPI_CTRL1 register)<br>0: No mode error<br>1: Mode error occurs. |
| Bit 4    | CCERR    | 0x0         | rw0c | CRC error<br>Set by hardware, and cleared by software.<br>0: No CRC error<br>1: CRC error occurs.  |
| Bit 3    | TUERR    | 0x0         | ro   | Transmitter underload error<br>Set by hardware, and cleared by software (read the SPI_STS register).<br>0: No underload error<br>1: Underload error occurs.<br>Note: This bit is only used in I <sup>2</sup> S mode.     |
| Bit 2    | ACS      | 0x0         | ro   | Audio channel state<br>This bit indicates the status of the current audio channel.<br>0: Left channel<br>1: Right channel<br>Note: This bit is only used in I <sup>2</sup> S mode.                                       |
| Bit 1    | TDBE     | 0x1         | ro   | Transmit data buffer empty<br>0: Transmit data buffer is not empty.<br>1: Transmit data buffer is not empty.   |
| Bit 0    | RDBF     | 0x0         | ro   | Receive data buffer full<br>0: Transmit data buffer is not full.<br>1: Transmit data buffer is full.   |

## SPI data register (SPI\_DT)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 15:0 | DT   | 0x0000      | rw   | Data value<br>This register controls read and write operations. When the data bit is set as 8 bit, only the 8-bit LSB [7:0] is valid. |

### SPI CRC register (SPI\_CPOLY) (Not used in I<sup>2</sup>S mode)

| Bit      | Name  | Reset value | Type | Description   |
|----------|-------|-------------|------|---|
| Bit 15:0 | CPOLY | 0x0007      | rw   | CRC polynomial<br>This register contains the polynomial used for CRC calculation.<br>Note: This register is valid only in SPI mode. |

### SPI Rx CRC register (SPI\_RCRC) (not used in I<sup>2</sup>S mode)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | RCRC | 0x0000      | ro   | Receive CRC<br>When CRC calculation is enabled, this register contains the CRC value computed based on the received data. This register is reset when the CCEN bit in the SPI_CTRL1 register is cleared.<br>When the data frame format is set to 8-bit data, only the 8-bit LSB ([7:0]) are calculated based on CRC8 standard; when 16-bit data bit is selected, follow CRC16 standard.<br>Note: This register is only used in SPI mode. |

### SPI Tx CRC register (SPI\_TCRC)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | TCRC | 0x0000      | ro   | Transmit CRC<br>When CRC calculation is enabled, this register contains the CRC value computed based on the transmitted data. This register is reset when the CCEN bit in the SPI_CTRL1 register is cleared.<br>When the data frame format is set to 8-bit data, only the 8-bit LSB ([7:0]) are calculated based on CRC8 standard; when 16-bit data bit is selected, follow CRC16 standard.<br>Note: This register is only used in SPI mode. |

### SPI\_I2S register (SPI\_I2SCTRL)

| Bit       | Name      | Reset value | Type | Description   |
|-----------|-----------|-------------|------|---|
| Bit 15:12 | Reserved  | 0x0         | resd | Forced 0 by hardware.   |
| Bit 11    | I2SMSEL   | 0x0         | rw   | I <sup>2</sup> S mode select<br>0: SPI mode<br>1: I <sup>2</sup> S mode   |
| Bit 10    | I2SEN     | 0x0         | rw   | I <sup>2</sup> S enable<br>0: Disabled<br>1: Enabled  |
| Bit 9:8   | OPERSEL   | 0x0         | rw   | I <sup>2</sup> S operation mode select<br>00: Slave transmission<br>01: Slave reception<br>10: Master transmission<br>11: Master reception            |
| Bit 7     | PCMFSSSEL | 0x0         | rw   | PCM frame synchronization<br>This bit is valid only when the PCM standard is used.<br>0: Short frame synchronization<br>1: Long frame synchronization |
| Bit 6     | Reserved  | 0x0         | resd | Kept at its default value   |

| Bit      | Name      | Reset value | Type | Description  |
|----------|-----------|-------------|------|--|
| Bit 5: 4 | STDSEL    | 0x0         | rw   | I <sup>2</sup> S standard select<br>00: Philips standard<br>01: MSB-aligned standard (left-aligned)<br>10: LSB-aligned standard (right-aligned)<br>11: PCM standard                                    |
| Bit 3    | I2SCLKPOL | 0x0         | rw   | I <sup>2</sup> S clock polarity<br>This bit indicates the clock polarity on the clock pin in idle state.<br>0: Low<br>1: High  |
| Bit 2: 1 | I2SDBN    | 0x0         | rw   | I <sup>2</sup> S data bit num<br>00: 16-bit data length<br>01: 24-bit data length<br>10: 32-bit data length<br>11: Not allowed.  |
| Bit 0    | I2SCBN    | 0x0         | rw   | I <sup>2</sup> S channel bit num<br>This bit can be configured only when the I <sup>2</sup> S is set to 16-bit data; otherwise, it is fixed to 32-bit by hardware.<br>0: 16-bit wide<br>1: 32-bit wide |

### SPI\_I2S prescaler register (SPI\_I2SCLKP)

| Bit                  | Name      | Reset value | Type | Description   |
|----------------------|-----------|-------------|------|---|
| Bit 15:12            | Reserved  | 0x0         | resd | Forced 0 by hardware.   |
| Bit 9                | I2SMCLKOE | 0x0         | rw   | I <sup>2</sup> S Master clock output enable<br>0: Disabled<br>1: Enabled  |
| Bit 8                | I2SODD    | 0x0         | rw   | IOdd factor for I <sup>2</sup> S division<br>0: Actual divider factor = I2SDIV × 2<br>1: Actual divider factor = (I2SDIV × 2) + 1 |
| Bit 11:10<br>Bit 7:0 | I2SDIV    | 0x02        | rw   | I <sup>2</sup> S division<br>It is not allowed to configure I2SDIV[9:0] = 0 or I2SDIV[9:0] = 1                                    |

# 14 Timer

HT32F49041 timers include basic timers, general-purpose timers, and advanced timers.

Please refer to Section 14.1 ~ Section 14.5 for the detailed function modes. All functions of different timers are shown in the following tables.

**Table 46. TMR functional comparison**

| Timer type             | Timer          | Counter bit | Count mode            | Repetition | Prescaler | DMA requests | Capture/compare channel | PWM input mode | EXT input | Brake input |
|------------------------|----------------|-------------|-----------------------|------------|-----------|--------------|-------------------------|----------------|-----------|-------------|
| Advanced-control timer | TMR1           | 16          | Up<br>Down<br>Up/Down | 16-bit     | 1~65535   | O            | 4                       | O              | O         | O           |
| General-purpose timer  | TMR2           | 16/32       | Up<br>Down<br>Up/Down | X          | 1~65535   | O            | 4                       | O              | O         | X           |
|                        | TMR3           | 16          | Up<br>Down<br>Up/Down | X          | 1~65535   | O            | 4                       | O              | O         | X           |
|                        | TMR13<br>TMR14 | 16          | Up                    | X          | 1~65535   | X            | 1                       | X              | X         | X           |
|                        | TMR15          | 16          | Up                    | 8-bit      | 1~65535   | O            | 2                       | O              | X         | O           |
|                        | TMR16<br>TMR17 | 16          | Up                    | 8-bit      | 1~65535   | O            | 1                       | X              | X         | O           |
| Basic timer            | TMR6<br>TMR7   | 16          | Up                    | X          | 1~65535   | O            | X                       | X              | X         | X           |

| Timer type             | Timer          | Counter bit | Count mode            | PWM output | Single pulse output | Complementary output | Dead-time | Encoder interface connection | Interfacing with hall sensors | Linkage peripheral            |
|------------------------|----------------|-------------|-----------------------|------------|---------------------|----------------------|-----------|------------------------------|-------------------------------|-------------------------------|
| Advanced-control timer | TMR1           | 16          | Up<br>Down<br>Up/Down | O          | O                   | O                    | O         | O                            | O                             | Timer synchronization<br>ADC  |
| General-purpose timer  | TMR2           | 16/32       | Up<br>Down<br>Up/Down | O          | O                   | X                    | X         | O                            | O                             | Timer synchronization<br>ADC  |
|                        | TMR3           | 16          | Up<br>Down<br>Up/Down | O          | O                   | X                    | X         | O                            | O                             | Timer synchronization/<br>ADC |
|                        | TMR13<br>TMR14 | 16          | Up                    | O          | O                   | X                    | X         | X                            | X                             | NA                            |
|                        | TMR15          | 16          | Up                    | O          | O                   | O                    | O         | X                            | X                             | Timer synchronization/<br>ADC |
|                        | TMR16<br>TMR17 | 16          | Up                    | O          | O                   | O                    | O         | X                            | X                             | NA                            |
| Basic timer            | TMR6<br>TMR7   | 16          | Up                    | X          | X                   | X                    | X         | X                            | X                             | ADC                           |

## Basic timer (TMR6 and TMR7)

### TMR6 and TMR7 introduction

Each of the basic timers (TMR6 and TMR7) includes a 16-bit upcounter and the corresponding control logic. without being connected to external I/Os, they can be used for a basic timing.

### TMR6 and TMR7 main features

- 16-bit upcounter, auto reload
- 16-bit prescaler used to divide the TMR\_CLK frequency by any factor between 1 and 65536

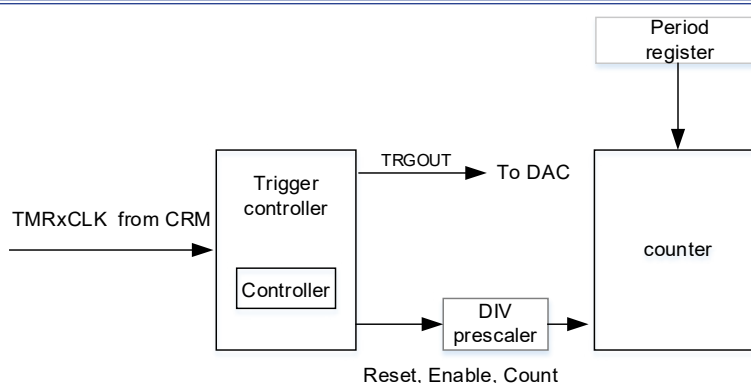


Figure 87. Basic timer block diagram

### TMR6 and TMR7 function overview

#### Counting clock

The counter clock of TMR6 and TMR7 is provided by the internal clock source (CK\_INT) divided by prescaler. When TMR's APB clock prescaler factor is 1, the CK\_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

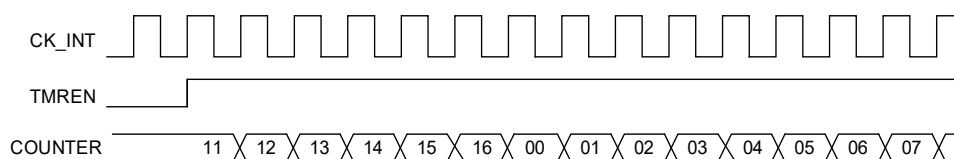


Figure 88. Control circuit with CK\_INT divided by 1

#### Counting mode

The basic timer only supports upcounting mode. It has an internal 16-bit counter in which the value is loaded with the TMRx\_PR register.

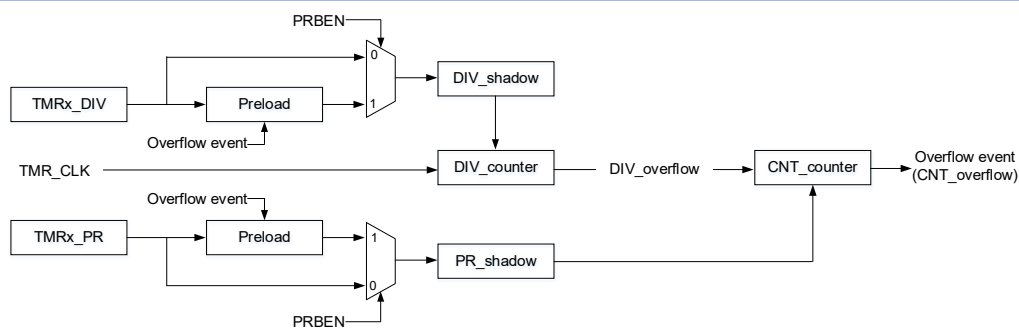
The value in the TMRx\_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN = 1), the value in the TMRx\_PR register is transferred to the shadow register only at an overflow event.

TMRx\_DIV register is used to define the counter frequency of the counter. The counter counts once every  $DIV[15:0] + 1$  clock cycle. Similar to TMRx\_PR register, after enabling periodic buffer, the value of the TMRx\_DIV register are transferred into the shadow register at each overflow event.

Reading the TMRx\_CNT register returns the current counter value. Writing the TMRx\_CNT register will update the current counter value.

An overflow event is enabled by default. It can be disabled by setting OVFE = 1 in the TMRx\_CTRL1 register. The OVFS bit in the TMRx\_CTRL1 register is used to select the source of an overflow event, which is, by default, counter overflow or underflow, setting OVFSWTR, reset signal generated by slave mode timer controller in reset mode. Once the OVFS is set, an overflow event is generated only when overflow or underflow occurs.

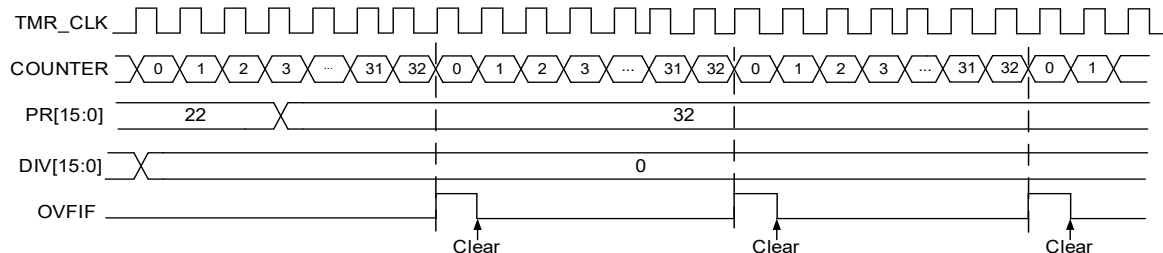
Setting the TMREN bit (TMREN = 1) enables the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR\_EN is set 1 clock cycle after the TMREN is set.



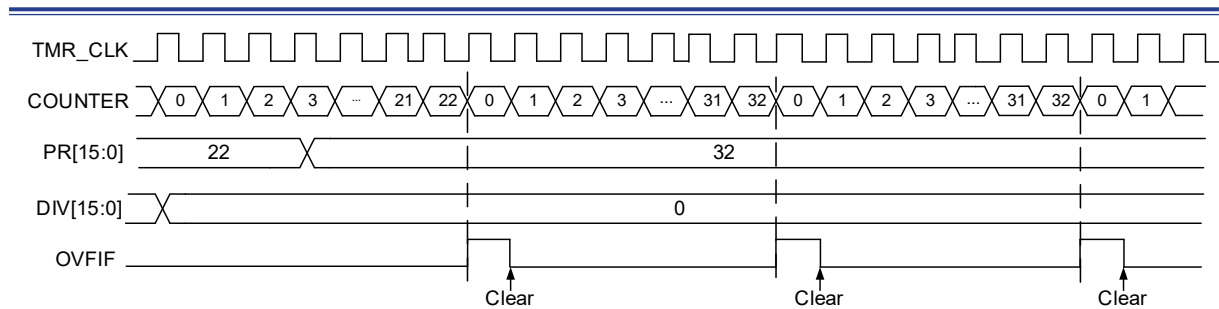
**Figure 89. Basic structure of a counter**

### Upcounting mode

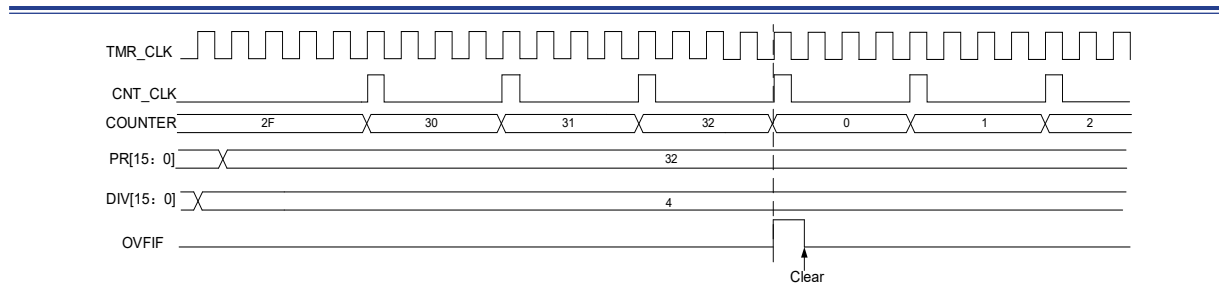
This mode is enabled by setting TWCMSSEL[1:0] = 2'b00 and OWCDIR = 1'b0 in the TMRx\_CTRL register. In upcounting mode, the counter counts from 0 to the value programmed in the TMRx\_PR register, then restarts from 0 and generates a counter overflow event with setting OVFI = 1 at the same time. If the overflow event is disabled, the counter is no longer reloaded with a prescaler value and a periodic value when a counter overflow event occurs; otherwise, the counter is updated with prescaler and periodic values at an overflow event.



**Figure 90. Overflow event when PRBEN = 0**



**Figure 91. Overflow event when PRBEN = 1**



**Figure 92. Counting timing diagram when the prescaler division is 4**

### Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the TMRx counter stops counting when the TMRx\_PAUSE bit is set.

### TMR6 and TMR7 registers

These peripheral registers must be accessed by word (32 bits).

In Table 47, all the TMRx registers are mapped to a 16-bit addressable space.

**Table 47. TMR6 and TMR7 register table and reset value**

| Register   | Offset | Reset value |
|------------|--------|-------------|
| TMRx_CTRL1 | 0x00   | 0x0000      |
| TMRx_CTRL2 | 0x04   | 0x0000      |
| TMRx_IDEN  | 0x0C   | 0x0000      |
| TMRx_ISTS  | 0x10   | 0x0000      |
| TMRx_SWEVT | 0x14   | 0x0000      |
| TMRx_CVAL  | 0x24   | 0x0000      |
| TMRx_DIV   | 0x28   | 0x0000      |
| TMRx_PR    | 0x2C   | 0x0000      |



### TMR6 and TMR7 control register 1 (TMRx\_CTRL1)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:8 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 7    | PRBEN    | 0x0         | rw   | Period buffer enable<br>0: Period buffer is disabled.<br>1: Period buffer is enabled.  |
| Bit 6:4  | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 3    | OCMEN    | 0x0         | rw   | One cycle mode enable<br>This bit is used to select whether to stop the counter at overrun event.<br>0: Disabled<br>1: Enabled   |
| Bit 2    | OVFS     | 0x0         | rw   | Overflow event source<br>This bit is used to configure overflow event or DMA request sources.<br>0: Counter overflow, setting the OVFSWTR bit or overflow event generated from the slave controller<br>1: Only counter overflow generates an overflow event.   |
| Bit 1    | OVFEN    | 0x0         | rw   | Overflow event enable<br>This bit is used to enable or disable OEV event generation.<br>0: OEV event is enabled. An overflow event is generated by any of the following events:<br>– Counter overflow<br>– Setting the OVFSWTR bit<br>– Overflow event generated from the slave controller<br>1: OEV event is disabled.<br>If the OVFSWTR bit is set, or a hardware reset is generated from the slave controller, the counter and the prescaler are reinitialized.<br>Note: This bit is set and cleared by software. |
| Bit 0    | TMREN    | 0x0         | rw   | TMR enable<br>0: Disabled<br>1: Enabled  |

### TMR6 and TMR7 control register 2 (TMRx\_CTRL2)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:7 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 6:4  | PTOS     | 0x0         | rw   | Master TMR output selection<br>This field is used to select the signals in master mode to be sent to slave timers.<br>000: Software overflow<br>001: Enable<br>010: Overflow |
| Bit 3:0  | Reserved | 0x0         | resd | Kept at its default value.   |

#### TMR6 and TMR7 DMA/interrupt enable register (TMRx\_IDEN)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:9 | Reserved | 0x0         | resd | Kept at its default value.                                     |
| Bit 8    | OVFDEN   | 0x0         | rw   | Overflow event DMA request enable<br>0: Disabled<br>1: Enabled |
| Bit 7:1  | Reserved | 0x0         | resd | Kept at its default value.                                     |
| Bit 0    | OVFIEN   | 0x0         | rw   | Overflow interrupt enable<br>0: Disabled<br>1: Enabled         |

#### TMR6 and TMR7 interrupt status register (TMRx\_ISTS)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:1 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 0    | OVFIF    | 0x0         | rw0c | Overflow interrupt flag<br>This bit is set by hardware at an overflow event. It is cleared by software.<br>0: No overflow event occurs.<br>1: Overflow event occurs, and OVFE = 0, and OVFS = 0 in the TMRx_CTRL1 register:<br>– An overflow event occurs when OVFG = 1 in the TMRx_SWEVE register<br>– An overflow event occurs when the counter value (CVAL) is reinitialized by a trigger event. |

#### TMR6 and TMR7 software event register (TMRx\_SWEVT)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:1 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 0    | OVFSWTR  | 0x0         | rw0c | Overflow event triggered by software<br>An overflow event is triggered by software.<br>0: No effect<br>1: Generate an overflow event by software write operation. |

#### TMR6 and TMR7 counter value (TMRx\_CVAL)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---------------|
| Bit 15:0 | CVAL | 0x0         | rw   | Counter value |

#### TMR6 and TMR7 division (TMRx\_DIV)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 15:0 | DIV  | 0x0         | rw   | Divider value<br>The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK} / (DIV[15:0] + 1)$ .<br>At each overflow event, DIV value is sent to the DIV register. |

#### TMR6 and TMR7 period register (TMRx\_PR)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | PR   | 0x0         | rw   | Period value<br>This indicates the period value of the TMRx counter. The timer stops working when the period value is 0. |

## General-purpose timer (TMR2 and TMR3)

### TMR2 and TMR3 introduction

The general-purpose timer (TMR2 and TMR3) consists of a 16-bit counter supporting up, down, up/down (bidirectional) counting modes, four capture/compare registers, and four independent channels to achieve input capture and programmable PWM output.

### TMR2 and TMR3 main features

- Source of count clock is selectable: internal clock, external clock and internal trigger
- 16-bit up, down, up/down and encoder mode counter (TMR2 can be extended to 32-bit)
- 4 independent channels for input capture, output compare, PWM generation and one-pulse mode output
- Synchronization control between master and slave timers
- Interrupt/DMA is generated at overflow event, trigger event and channel event
- Support TMR burst DMA transfer

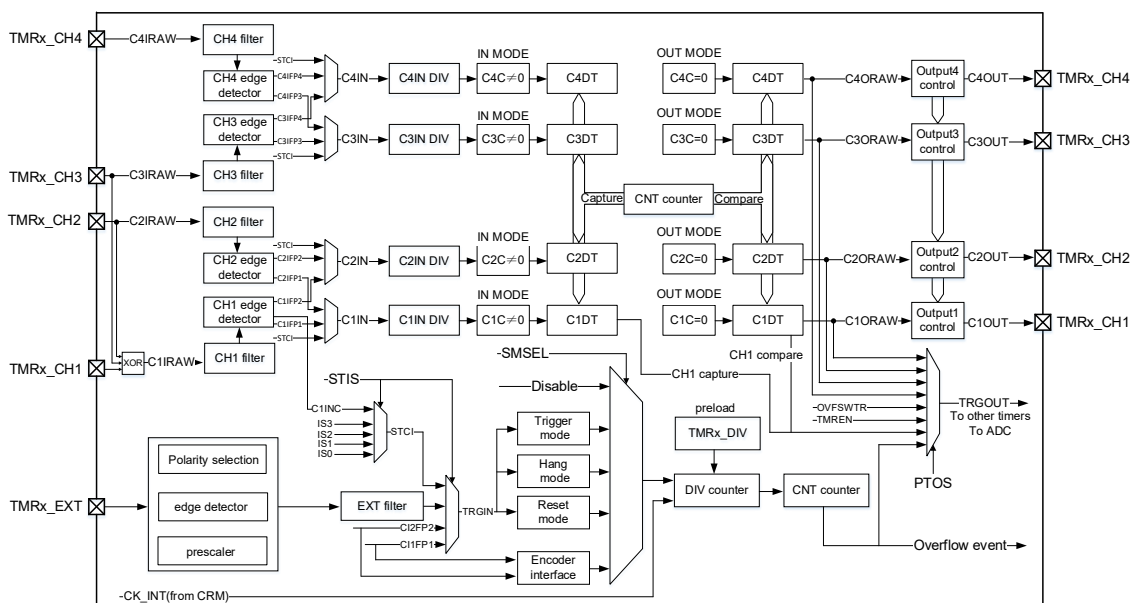
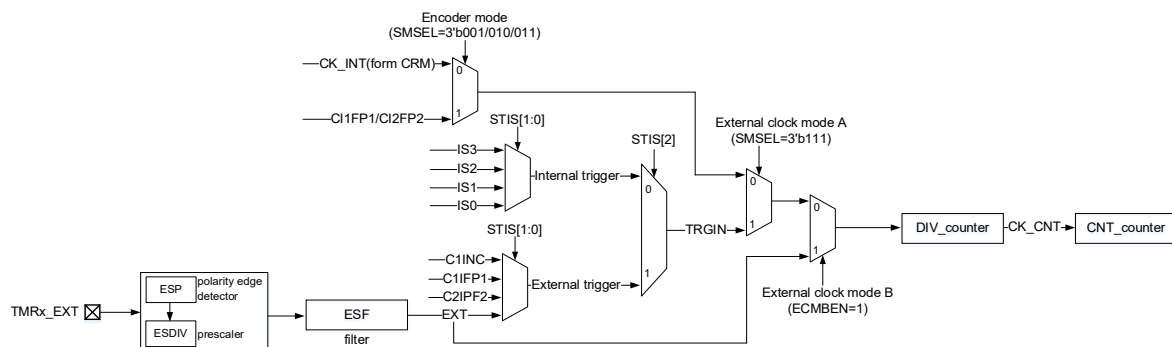


Figure 93. General-purpose timer block diagram

### TMR2 and TMR3 function overview

#### Counting clock

The count clock of TMR2 ~ TMR5 can be provided by the internal clock (CK\_INT), external clock (external clock mode A and B) and internal trigger input (ISx).



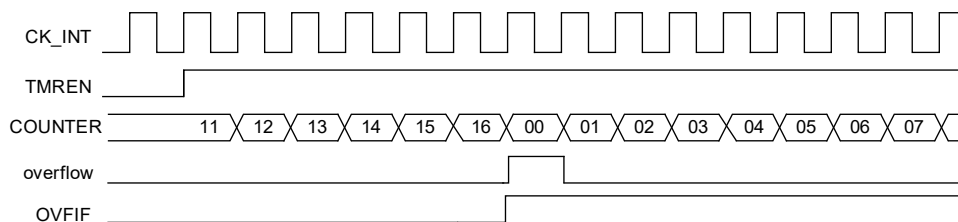
**Figure 94. Counting clock**

### Internal clock (CK\_INT)

By default, the CK\_INT, which is divided by a prescaler, is used to drive the counter to start counting. When TMR's APB clock prescaler factor is 1, the CK\_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

Follow the procedures below:

- Select a counting mode by setting the TWCMSSEL[1:0] in TMRx\_CTRL1 register. If an unidirectional aligned counting mode is selected, it is necessary to select a counting direction through the OWCDIR in TMRx\_CTRL1 register.
- Set counting frequency through TMRx\_DIV register.
- Set counting cycles through TMRx\_PR register.
- Enable a counter by setting the TMREN bit in the TMRx\_CTRL1 register.



**Figure 95. Control circuit with CK\_INT, TMRx\_DIV = 0x0 and TMRx\_PR = 0x16**

### External clock (TRGIN/EXT)

The counter clock can be provided by two external clock sources, namely, TRGIN and EXT signals.

SMSEL = 3'b111: External clock mode A is selected. By setting the STIS[2:0] bit, select an external clock source TRGIN signal to drive the counter to start counting.

The external clock sources include: C1INC (STIS = 3'b100, channel 1 rising edge and falling edge), C11FP1 (STIS = 3'b101, a signal after channel 1 filter and polarity selection), C21FP2 (STIS = 3'b110, a signal after channel 2 filter and polarity selection) and EXT (STIS = 3'b111, external input signal after polarity selection, frequency division and filter).

ECMBEN = 1: External clock mode B is selected. The counter is driven by external input that has gone through polarity selection, frequency division and filtering. The external clock mode B is

equivalent to the external clock mode A, and the EXT signal is used as an external force TRGIN,

To use external clock mode A, follow the steps below:

■ Set external source TRGIN parameters

If the TMRx\_CH1 is used as a source of TRGIN, it is necessary to configure channel 1 input filter (C1DF[3:0] in TMRx\_CM1 register) and channel 1 input polarity (C1P/C1CP in TMRx\_CCTRL register);

If the TMRx\_CH2 is used as source of TRGIN, it is necessary to configure channel 1 input filter (C2DF[3:0] in TMRx\_CM1 register) and channel 2 input polarity (C2P/C2CP in TMRx\_CCTRL register);

If the TMRx\_EXT is used as a source of TRGIN, it is necessary to configure the external signal polarity (ESP in TMRx\_STCTRL register), external signal frequency division (ESDIV[1:0] in TMRx\_STCTRL) and external signal filter (ESF[3:0] in TMRx\_STCTRL register).

■ Set TRGIN signal source through the STIS[1:0] bit in TMRx\_STCTRL register

■ Enable external clock mode A by setting SMSEL = 3'b111 in TMRx\_STCTR register.

■ Set counting frequency through the DIV[15:0] in TMRx\_DIV register.

■ Set counting period through the PR[15:0] in TMRx\_PR register.

■ Enable counter through the TMREN bit in TMRx\_CTRL1 register.

To use external clock mode B, follow the steps below:

■ Set external signal polarity through the ESP bit in TMRx\_STCTRL register.

■ Set external signal frequency division through the ESDIV[1:0] bit in TMRx\_STCTRL register.

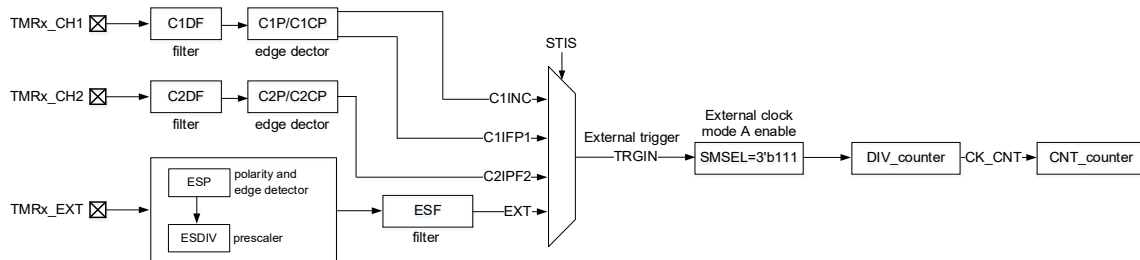
■ Set external signal filter through the ESF[3:0] bit in TMRx\_STCTRL register.

■ Enable external clock mode B through the ECBEN bit in TMRx\_STCTR register.

■ Set counting frequency through the DIV[15:0] bit in TMRx\_DIV register.

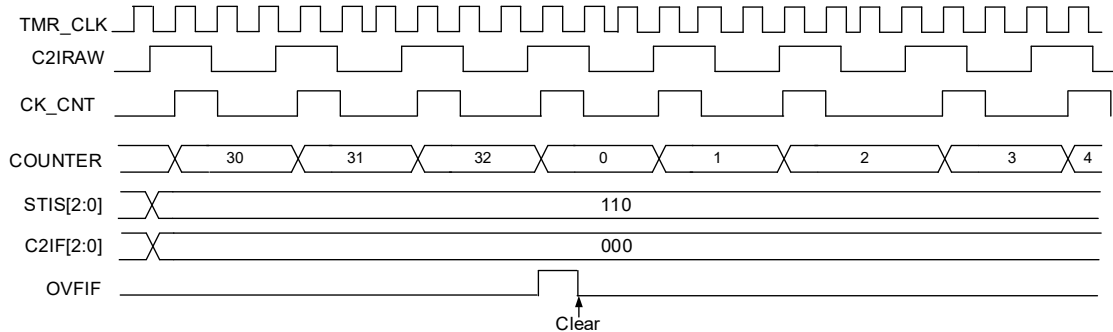
■ Set counting period through the PR[15:0] bit in TMRx\_PR register.

■ Enable counter through the TMREN in TMRx\_CTRL1 register.

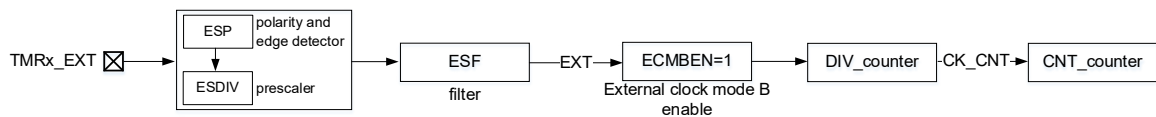


**Figure 96. Block diagram of external clock mode A**

Note: The delay between the signal on the input side and the actual clock of the counter is due to the synchronization circuit.

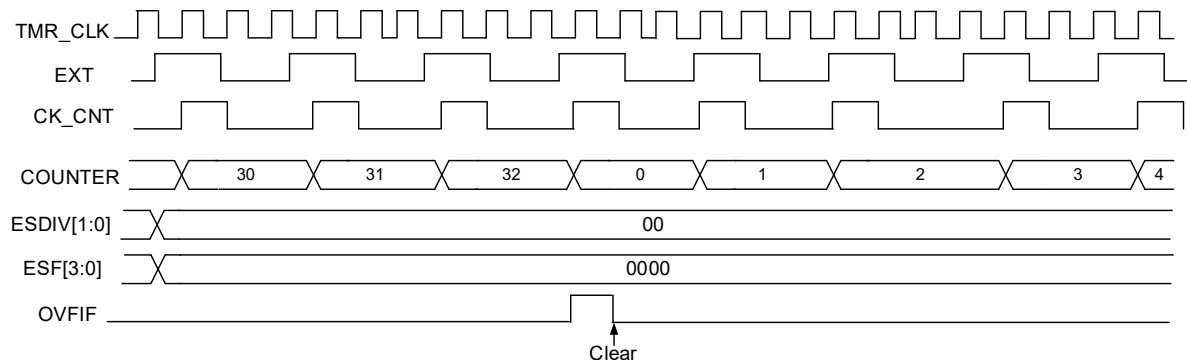


**Figure 97. Counting in external clock mode A, PR = 0x32 and DIV = 0x0**



**Figure 98. Block diagram of external clock mode B**

Note: The delay between the EXT signal on the input side and the actual clock of the counter is due to the synchronization circuit.



**Figure 99. Counting in external clock mode B, PR = 0x32, DIV = 0x0**

### Internal trigger input (ISx)

Timer synchronization allows interconnection between several timers. The TMR\_CLK of one timer can be provided by the TRGOUT signal output by another timer. Set the STIS[2:0] bit to select internal trigger signal to enable counting.

Each timer (TMR2 to TMR5) consists of a 16-bit prescaler, which is used to generate the CK\_CNT that enables the counter to count. The frequency division relationship between the CK\_CNT and TMR\_CLK can be adjusted by setting the value of the TMRx\_DIV register. The prescaler value can be modified at any time, but it takes effect only when the next overflow event occurs.

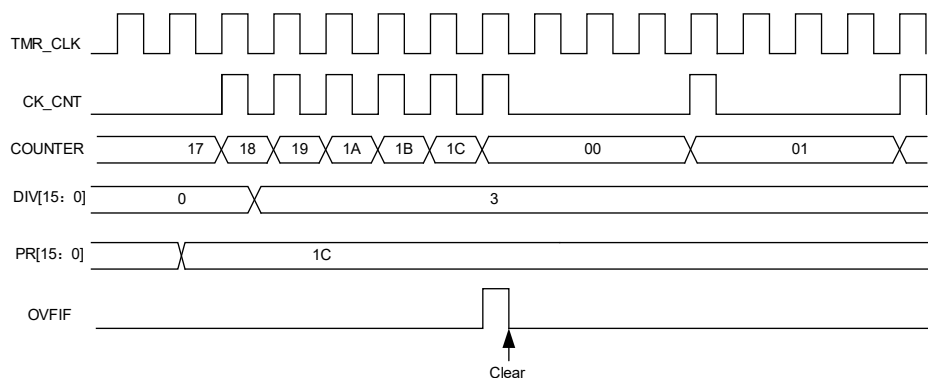
Below is the configuration procedure for internal trigger input:

- Set counting cycles through TMRx\_PR register.
- Set counting frequency through TMRx\_DIV register.
- Set counting modes through the TWCMSEL[1:0] in TMRx\_CTRL1 register.
- Select internal trigger by setting STIS[2:0] = 3'b000 ~ 3'b011 in TMRx\_STCTRL register.
- Select external clock mode A by setting SMSEL[2:0] = 3'b111 in TMRx\_STCTRL register.
- Enable TMRx to start counting through the TMREN in TMRx\_CTRL1 register.

**Table 48. TMRx internal trigger connection**

| Slave controller | IS0<br>(STIS = 000) | IS1<br>(STIS = 001) | IS2<br>(STIS = 010) | IS3<br>(STIS = 011) |
|------------------|---------------------|---------------------|---------------------|---------------------|
| TMR1             | TMR15               | TMR2                | TMR3                | —                   |
| TMR2             | TMR1                | TMR15               | TMR3                | USB_OTG_SOF         |
| TMR3             | TMR1                | TMR2                | TMR15               | —                   |
| TMR15            | TMR2                | TMR3                | TMR16_OC            | TMR17_OC            |

Note: If there is no corresponding timer in a device, the corresponding trigger signal ISx is not present.



**Figure 100. Counter timing with prescaler value changing from 1 to 4**

### Counting mode

The timer (TMR2 and TMR3) supports several counting modes to meet different application scenarios. Each timer has an internal 16-bit up, down, up/down counter. TMR2 can be extended to 32-bit by setting the PMEN bit. The TMRx\_PR register is loaded with the counter value.

The value in the TMRx\_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN = 1), the value in the TMRx\_PR register is transferred to the shadow register only at an overflow event.

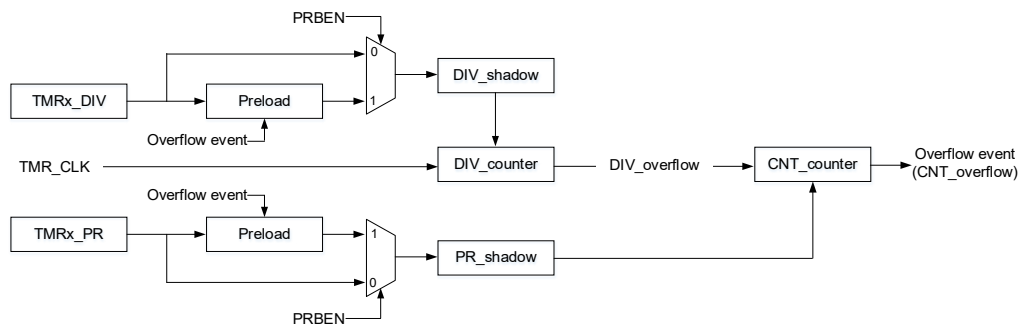
TMRx\_DIV register is used to define the counter frequency of the counter. The counter counts once every DIV[15:0] + 1 clock cycle. Similar to TMRx\_PR register, after enabling periodic buffer, the value of the TMRx\_DIV register are transferred into the shadow register at each overflow event.

Reading the TMRx\_CNT register returns the current counter value. Writing the TMRx\_CNT register will update the current counter value.

An overflow event is enabled by default. It can be disabled by setting OVFEN = 1 in the TMRx\_CTRL1 register. The OVFS bit in the TMRx\_CTRL1 register is used to select the source of an overflow event, which is, by default, counter overflow or underflow, setting OVFSWTR, reset signal

generated by slave mode timer controller in reset mode. Once the OVFS is set, an overflow event is generated only when overflow or underflow occurs.

Setting the TMREN bit (TMREN = 1) enables the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR\_EN is set 1 clock cycle after the TMREN is set.

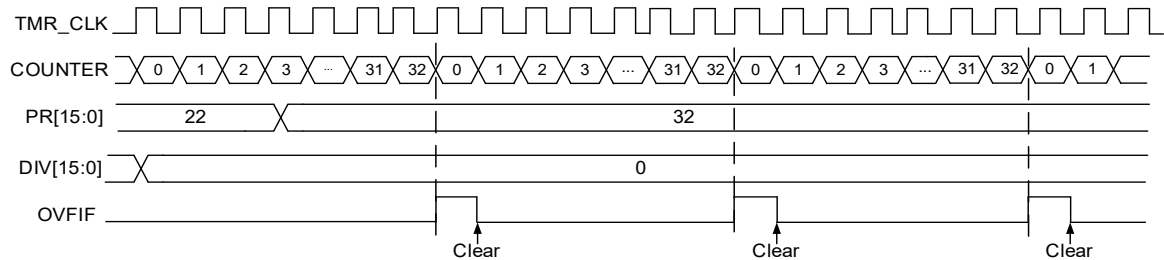


**Figure 101. Basic structure of a counter**

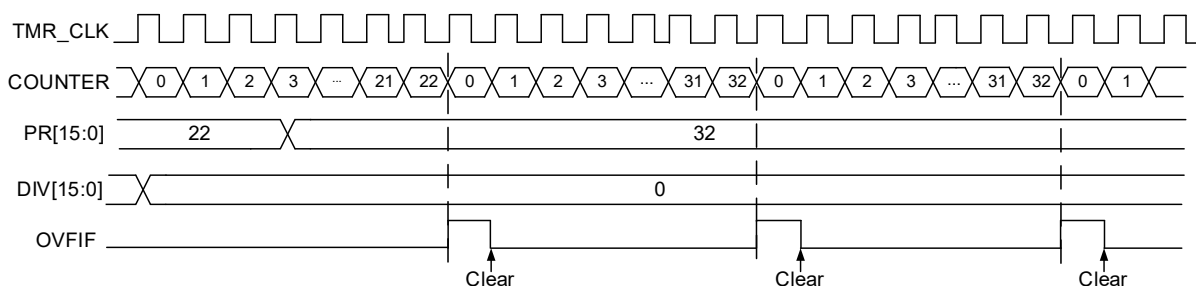
### Upcounting mode

This mode is enabled by setting TWCMSSEL[1:0] = 2'b00 and OWCDIR = 1'b0 in the TMRx\_CTRL1 register.

In upcounting mode, the counter counts from 0 to the value programmed in the TMRx\_PR register, restarts from 0, and generates a counter overflow event, with setting OVFI = 1. If the overflow event is disabled, the register is no longer reloaded with the prescaler and re-loaded value after counter overflow occurs; otherwise, the prescaler and re-loaded value will be updated at an overflow event.



**Figure 102. Overflow event when PRBEN = 0**



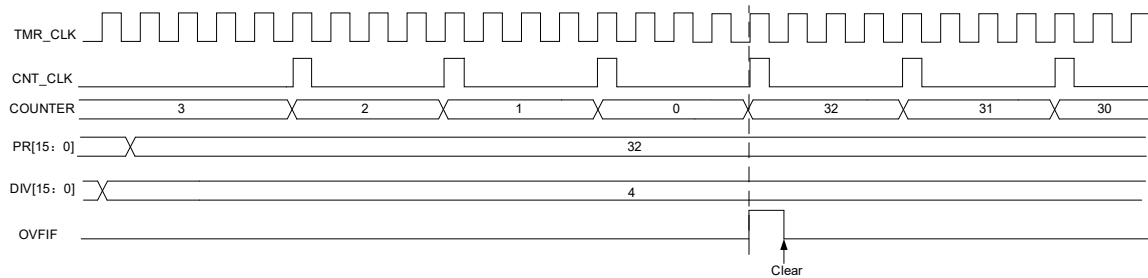
**Figure 103. Overflow event when PRBEN = 1**



### Downcounting mode

This mode is enabled by setting  $TWCMSEL[1:0] = 2'b00$  and  $OWCDIR = 1'b1$  in the  $TMRx\_CTRL1$  register.

In downcounting mode, the counter counts from the value programmed in the  $TMRx\_PR$  register down to 0, and restarts from the value programmed, and generates a counter underflow event.

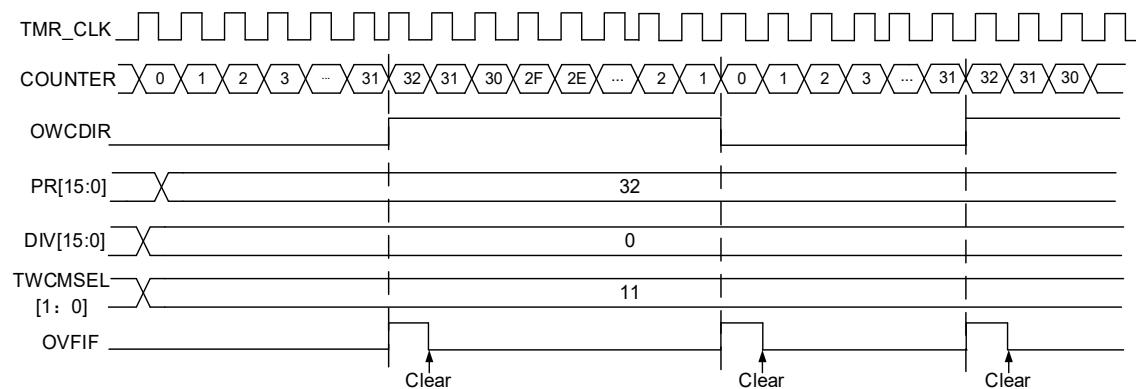


**Figure 104. Counter timing diagram with internal clock divided by 4**

### Up/down counting mode

This mode is enabled by setting  $TWCMSEL[1:0] \neq 2'b00$  in the  $TMRx\_CTRL1$  register. In up/down counting mode, the counter counts up/down alternatively. When the counter counts from the value programmed in the  $TMRx\_PR$  register down to 1, an underflow event is generated, and then restarts counting from 0. When the counter counts from 0 to the value of the  $TMRx\_PR$  register -1, an overflow event is generated, and then restarts counting from the value of the  $TMRx\_PR$  register. The  $OWCDIR$  bit indicates the current counting direction.

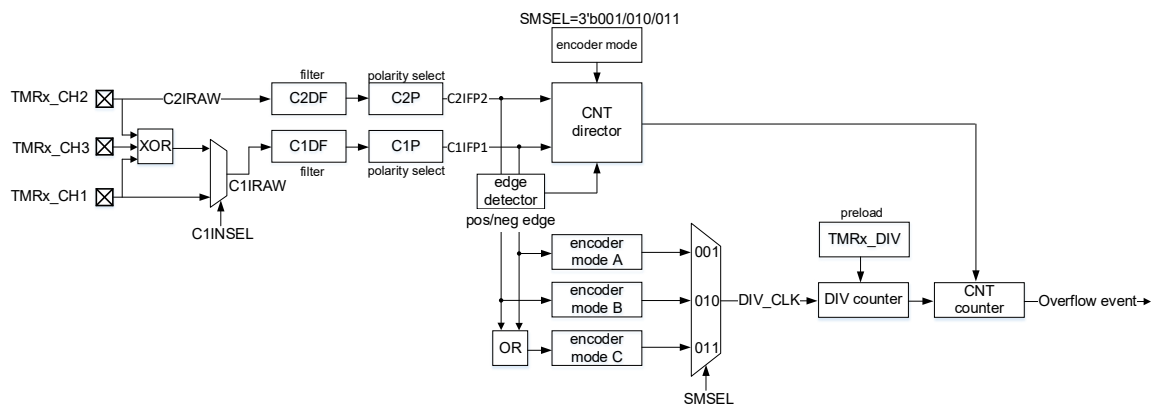
Note: The  $OWCDIR$  is read-only in up/down counting mode.



**Figure 105. Counter timing diagram with internal clock divided by 1 and  $TMRx\_PR = 0x32$**

## Encoder interface mode

In this mode, the two input (TMRx\_CH1 and TMRx\_CH2) signals are required. Depending on the level on one input, the counter counts up or down on the edge of the other input signal. The OWCDIR bit indicates the direction of the counter, as shown in the table below:



**Figure 106. Encoder mode structure**

Encoder mode A: SMSEL = 3'b001. The counter counts on the selected C1IFP1 edge (rising and falling edges), and the counting direction is dependent on the edge direction of C1IFP1 and the level of C2IFP2.

Encoder mode B: SMSEL = 3'b010. The counter counts on the selected C2IFP2 edge (rising and falling edges), and the counting direction is dependent on the edge direction of C2IFP2 and the level of C1IFP1.

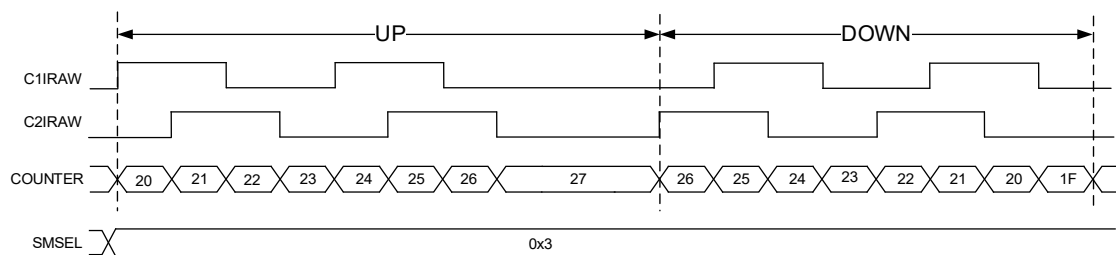
Encoder mode C: SMSEL = 3'b011. The counter counts on both C1IFP1 and C2IFP2 edges (rising and falling edges). The counting direction is dependent on the C1IFP1 edge direction and C2IFP2 level, and C2IFP2 edge direction and C1IFP1 level.

To use encoder mode, follow the procedures below:

- Set channel 1 input signal filtering through the C1DF[3:0] bit in the TMRx\_CM1 register;  
Set channel 1 input signal active level through the C1P bit in the TMRx\_CCTRL register.
- Set channel 2 input signal filtering through the C2DF[3:0] bit in the TMRx\_CM1 register;  
Set channel 2 input signal active level through the C2P bit in the TMRx\_CCTRL register.
- Set channel 1 as input mode through the C1C[1:0] bit in the TMRx\_CM1 register;  
Set channel 2 as input mode through the C2C[1:0] bit in the TMRx\_CM1 register.
- Select encoder mode A (SMSEL = 3'b001), encoder mode B (SMSEL = 3'b010), or encoder mode C (SMSEL = 3'b011) by setting the SMSEL[2:0] bit in the TMRx\_STCTRL register.
- Set counting cycles through the PR[15:0] bit in the TMRx\_PR register.
- Set counting frequency through the DIV[15:0] bit in the TMRx\_DIV register.
- Configure the corresponding IOs of TMRx\_CH1 and TMRx\_CH2 as multiplexed mode.
- Enable counter through the TMREN bit in the TMRx\_CTRL1 register.

**Table 49. Counting direction versus encoder signals**

| Active edge                        | Level on opposite signal<br>(C1IFP1 to C2IFP2, C2IFP2<br>to C1IFP1) | C1IFP1 signal |          | C2IFP2 signal |          |
|------------------------------------|---|---------------|----------|---------------|----------|
|                                    |   | Rising        | Falling  | Rising        | Falling  |
| Count on C1IFP1 only               | High  | Down          | Up       | No count      | No count |
|                                    | Low   | Up            | Down     | No count      | No count |
| Count on C2IFP2 only               | High  | No count      | No count | Up            | Down     |
|                                    | Low   | No count      | No count | Down          | Up       |
| Count on both C1IFP1<br>and C2IFP2 | High  | Down          | Up       | Up            | Down     |
|                                    | Low   | Up            | Down     | Down          | Up       |

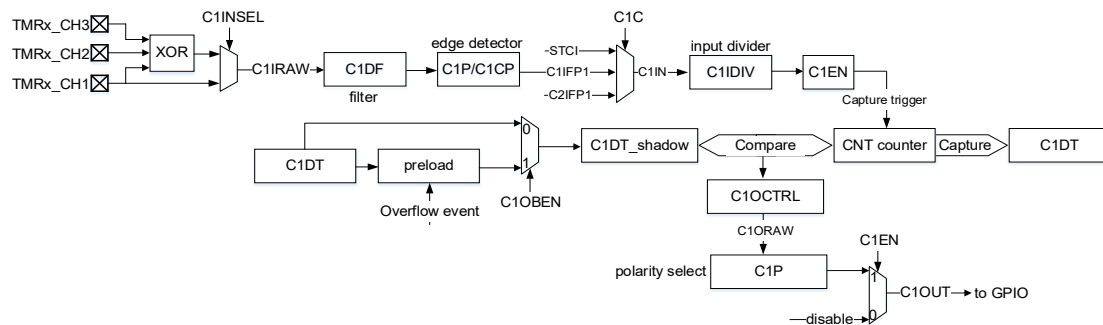


**Figure 107. Example of counter behavior in encoder interface mode (encoder mode C)**

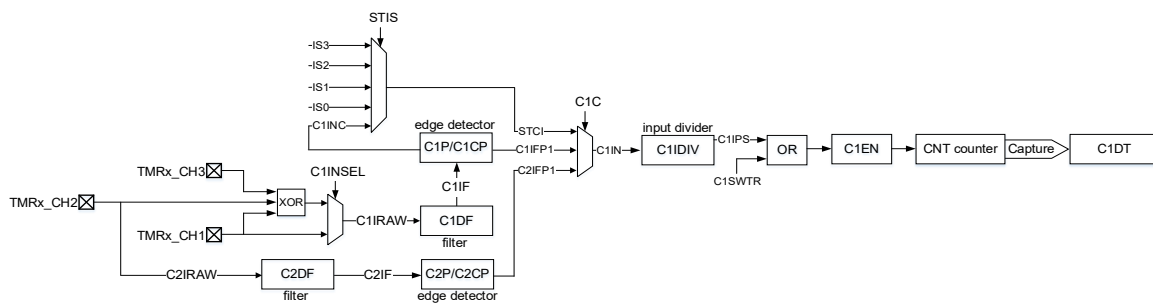
### TMR input function

Each of timers (TMR2 and TMR3) has four independent channels, with each channel being configured as input or output. As input, each channel input signal is processed as follows:

- TMRx\_CHx outputs the pre-processed CxIRAW. The C1INSE bit is used to select the source of C1IRAW from TMRx\_CH1, or XOR-ed TMRx\_CH1, TMRx\_CH2 and TMRx\_CH3.  
The sources of C2IRAW, C3IRAW and C4IRAW are TMRx\_CH2, TMRx\_CH3 and TMRx\_CH4 respectively.
- CxIRAW inputs digital filter and outputs filtered CxIF signal. The digital filter uses the CxDF bit to program sampling frequency and sampling times.
- CxIF inputs edge detector, and outputs the CxIFPx signal after edge selection. The edge selection depends on both CxP and CxCP bits. It is possible to select input rising edge, falling edge or both edges.
- CxIFPx inputs capture signal selector, and outputs the CxIN signal after capture signal selection. The capture signal selection is defined by CxC bits. It is possible to select CxIFPx, CyIFPx or STCI as CxIN source. Of those, CyIFPx (x≠y) is the CyIFPy signal that is from Y channel and processed by channel-x edge detector. The STCI comes from slave timer controller, and its source is selected by STIS bit.
- CxIN outputs the CxIPS signal that is divided by input channel divider. The divider factor can be defined as No division, /2, /4 or /8, by the CxIDIV bit.



**Figure 108. Input/output channel 1 main circuit**



**Figure 109. Channel 1 input stage**

### Input mode

In input mode, the TMRx\_CxDT register latches the current counter values after the selected trigger signal is detected, and the capture compare interrupt flag bit (CxIF) is set to 1. An interrupt or a DMA request will be generated if the CxIEN and CxDEN bits are enabled. If the selected trigger signal is detected when the CxIF is set to 1, a capture overflow event is generated, and the previous counter value will be overwritten by the current counter value, with setting CxRF to 1.

To capture the rising edge of C1IN input, follow the procedure below

- Set C1C = 01 in the TMRx\_CM1 register to select the C1IN as channel 1 input.
- Set C1IN signal filter bandwidth (CxDF[3:0]).
- Set the active edge on the C1IN channel by writing C1P = 0 (rising edge) in the TMRx\_CCTRL register.
- Program C1IN signal capture frequency divider (C1DIV[1:0]).
- Enable channel 1 input capture (C1EN = 1).
- If needed, enable the relevant interrupt or DMA request by setting the C1IEN bit in the TMRx\_IDEN register or the C1DEN bit in the TMRx\_IDEN register.

### Timer Input XOR function

The 3 timer input pins (TMRx\_CH1, TMRx\_CH2 and TMRx\_CH3) are connected to the channel 1 (selected by setting the C1INSE in the TMRx\_CTRL2 register) through an XOR gate.

The XOR gate can be used to connect Hall sensors. For example, connect the three XOR inputs to the three Hall sensors respectively so as to calculate the position and speed of the rotation by analyzing three Hall sensor signals.

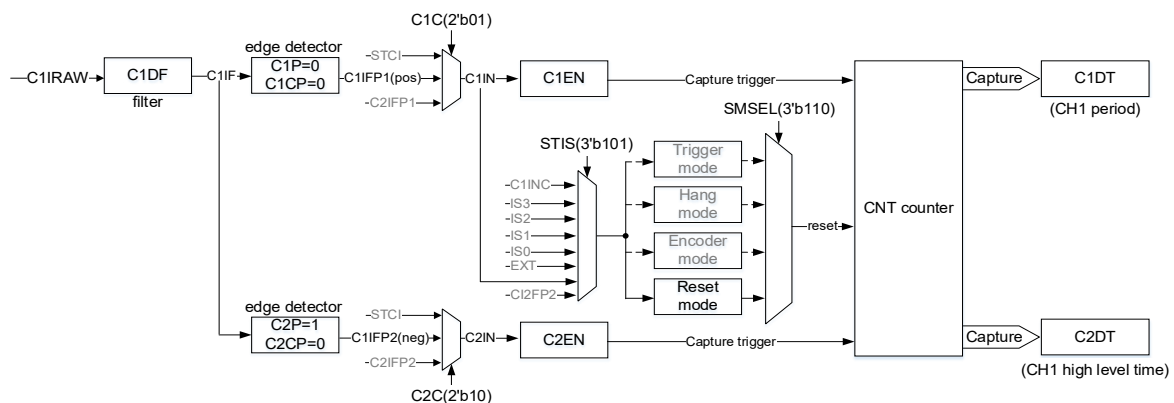
### PWM input

PWM input mode is applied to channel 1 and 2. To use this mode, both C1IN and C2IN are mapped on to the same TMRx\_CHx, and the CxIFPx of either channel 1 or channel 2 must be configured as trigger input and slave mode controller is configured in reset mode.

The PWM input mode can be used to measure the period and duty cycle of the PWM input signal. For example, the user can measure the period and duty cycle of the PWM applied on channel 1 using the following procedures:

- Set C1C = 2'b01: select C1IN for C1IFP1.
- Set C1P = 1'b0, select C1IFP1 rising edge active.
- Set C2C = 2'b10, select C2IN for C1IFP2.
- Set C2P = 1'b1, select C1IFP2 falling edge active.
- Set STIS = 3'b101, select the slave mode timer trigger signal as C1IFP1.
- Set SMSEL = 3'b100: configure the slave mode controller in reset mode.
- Set C1EN = 1'b1 and C2EN = 1'b1. Enable channel 1 and input capture.

After above configuration, the rising edge of channel 1 input signal will trigger the capture and stores the capture value into C1DT register, and it will reset the counter at the same time. The falling edge of the channel 1 input signal triggers the capture and stores the capture value into C2DT register. The period of the channel 1 input signal is calculated through C1DT, and its duty cycle through C2DT.



**Figure 110. PWM input mode configuration example**

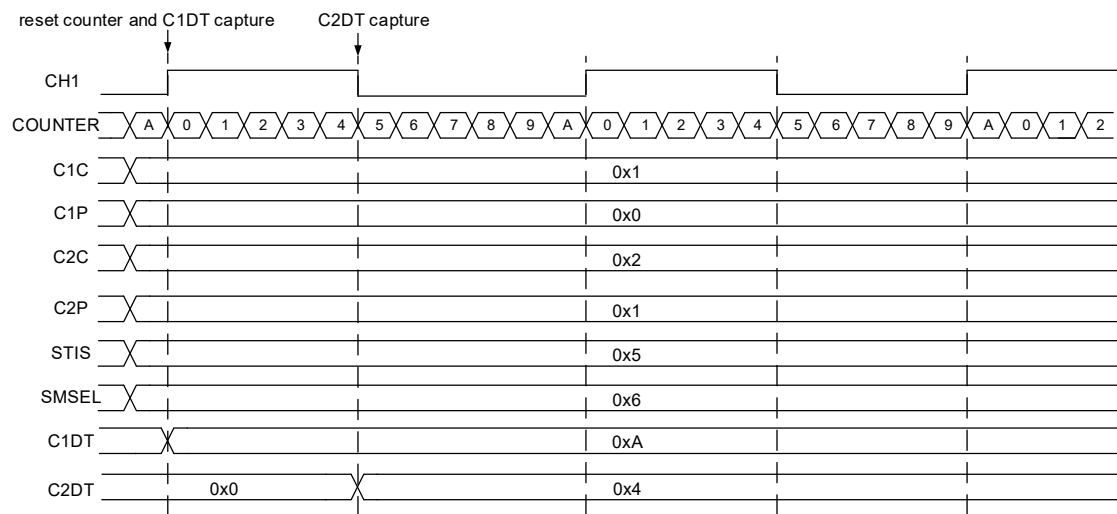


Figure 111. PWM input mode

### TMR output function

The TMR output consists of a comparator and an output controller. It is used to program the period, duty cycle and polarity of the output signal.

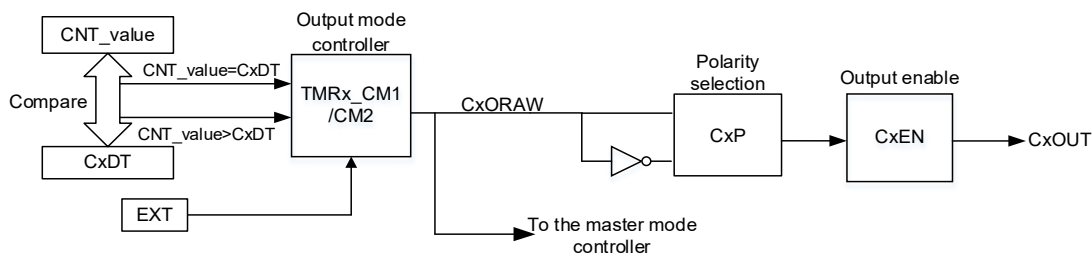


Figure 112. Capture/compare channel output stage (channel 1 to 4)

### Output mode

Write  $CxC[1:0] \neq 2'b00$  to configure the channel as output to implement multiple output modes. In this case, the counter value is compared with the value in the  $TMRx\_CxDT$  register, and the intermediate signal  $CxORAW$  is generated according to the output mode selected by  $CxOCTRL[2:0]$ , which is sent to IO after being processed by the output control circuit. The period of the output signal is configured by the  $TMRx\_PR$  register, while the duty cycle by the  $TMRx\_CxDT$  register.

Output compare modes include:

#### PWM mode A:

Enable PWM mode A by setting  $CxOCTRL = 3'b110$ . In upcounting mode,  $C1ORAW$  outputs high when  $TMRx\_C1DT > TMRx\_CVAL$ ; otherwise, it is low. In downcounting mode,  $C1ORAW$  outputs low when  $TMRx\_C1DT < TMRx\_CVAL$ ; otherwise, it is high.

To use PWM mode A, the following procedures are recommended:

- Set PWM period through TMRx\_PR register.
- Set PWM duty cycles through TMRx\_CxDT.
- Select PWM mode A by setting CxOCTRL = 3'b110 in the TMRx\_CM1/CM2 register.
- Set counting frequency through TMRx\_DIV register.
- Select counting mode by setting the TWCMSSEL[1:0] bit in the TMRx\_CTRL1 register.
- Select output polarity through the CxP and CxCP bits in the TMRx\_CCTRL register.
- Enable channel output through the CxEN and CxCEN bits in the TMRx\_CCTRL register.
- Enable TMRx output through the OEN bit in the TMRx\_BRK register.
- Configure GPIOs corresponding to TMR output channels as multiplexed mode.
- Enable TMRx to start counting through the TMREN bit in the TMRx\_CTRL1 register.

#### **PWM mode B:**

Enable PWM mode B by setting CxOCTRL = 3'b111. In upcounting mode, C1ORAW outputs low when TMRx\_C1DT > TMRx\_CVAL; otherwise, it is high. In downcounting mode, C1ORAW outputs high when TMRx\_C1DT < TMRx\_CVAL; otherwise, it is low.

#### **Forced output mode:**

Enable forced output mode by setting CxOCTRL = 3'b100/101. In this case, the CxORAW is forced to be the programmed level, regardless of the counter value. Despite this, the channel flag bit and DMA request still depend on the compare result.

#### **Output compare mode:**

Enable output compare mode by setting CxOCTRL = 3'b001/010/011. In this case, when the counter value matches the value of the CxDT register, the CxORAW is forced high (CxOCTRL = 3'b001), low (CxOCTRL = 3'b010) or toggling (CxOCTRL = 3'b011).

#### **One-pulse mode:**

This is a particular case of PWM mode. Enable one-pulse by setting OCMEN = 1. In this mode, the comparison match is performed in the current counting period. The TMREN bit is cleared as soon as the current counting is completed. Therefore, only one pulse is output. When configured in upcounting mode, the configuration must follow the rule: CVAL < CxDT ≤ PR; in downcounting mode, CVAL > CxDT is required.

#### **Fast output mode:**

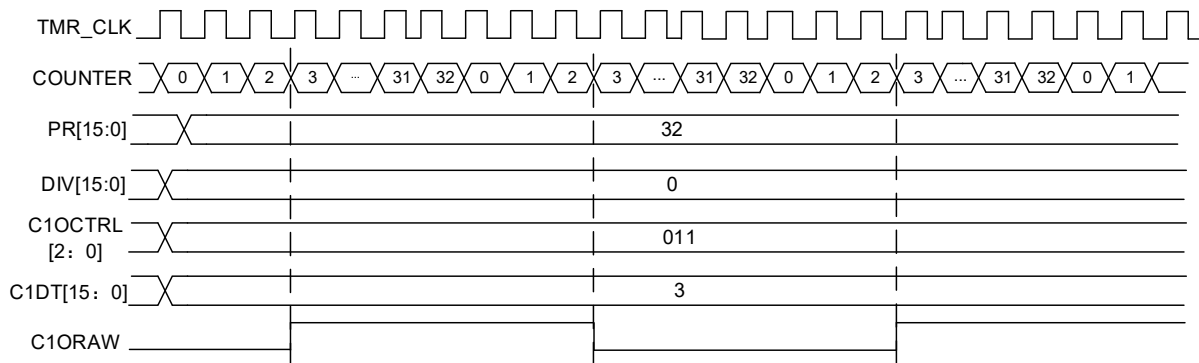
Enable this mode by setting CxOIEN = 1. If enabled, the CxORAW signal will not change when the counter value matches the CxDT, but change at the beginning of the current counting period. In other words, the comparison result is advanced, so the comparison result between the counter value and the TMRx\_CxDT register will determine the level of CxORAW in advance.

Figure 113 gives an example of output compare mode (toggle) with C1DT = 0x3. When the counter value is equal to 0x3, C1OUT toggles.

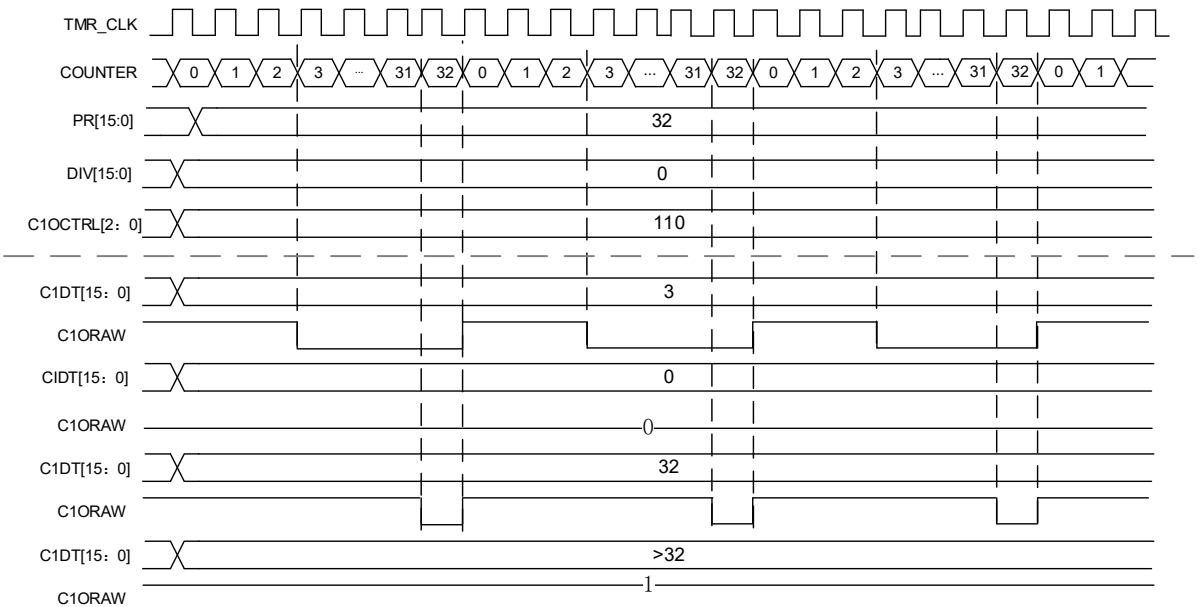
Figure 114 gives an example of the combination between upcounting mode and PWM mode A. The output signal behaves when PR = 0x32 but CxDT is configured with a different value.

Figure 115 gives an example of the combination between up/down counting mode and PWM mode A. The output signal behaves when PR = 0x32 but CxDT is configured with a different value.

Figure 116 gives an example of the combination between upcounting mode and one-pulse PWM mode B. The counter only counts only one cycle, and the output signal sends only one pulse.

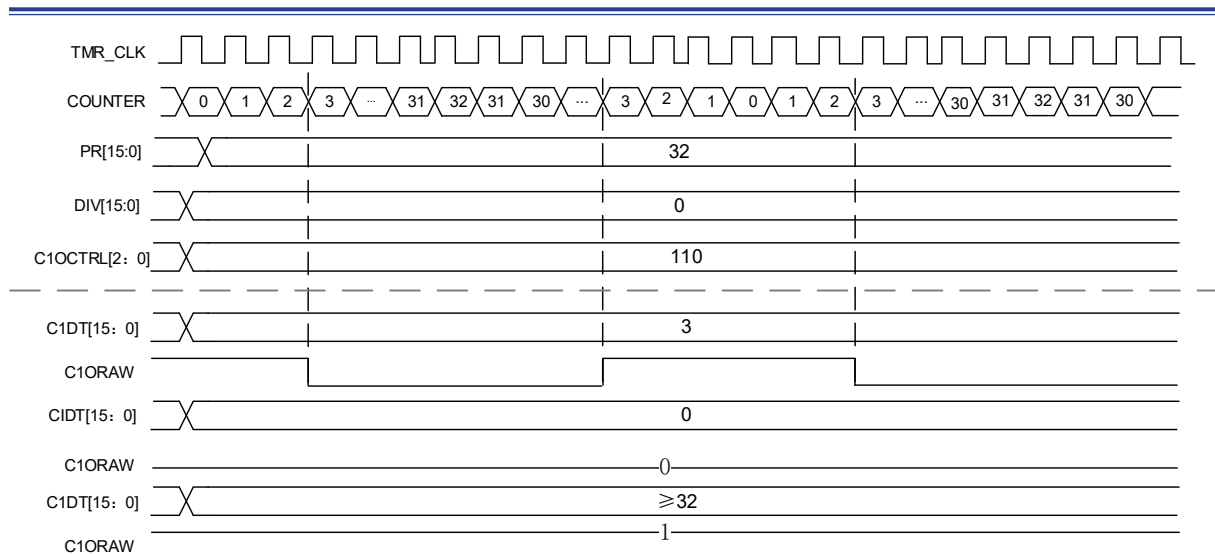


**Figure 113. C1ORAW toggles when counter value matches the C1DT value**

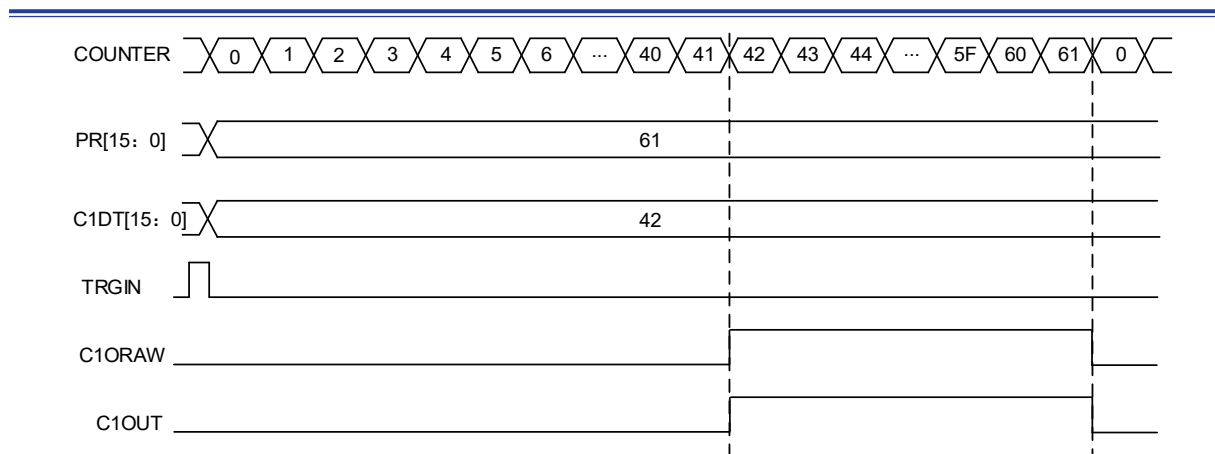


**Figure 114. Upcounting mode and PWM mode A**





**Figure 115. Up/down counting mode and PWM mode A**



**Figure 116. One-pulse mode**

#### Master mode timer event output

When TMR is used as a master timer, one of the following source of signals can be selected as TRGOUT output to a slave mode timer. This is done by setting the PTOS bit in the TMRxCTRL2 register.

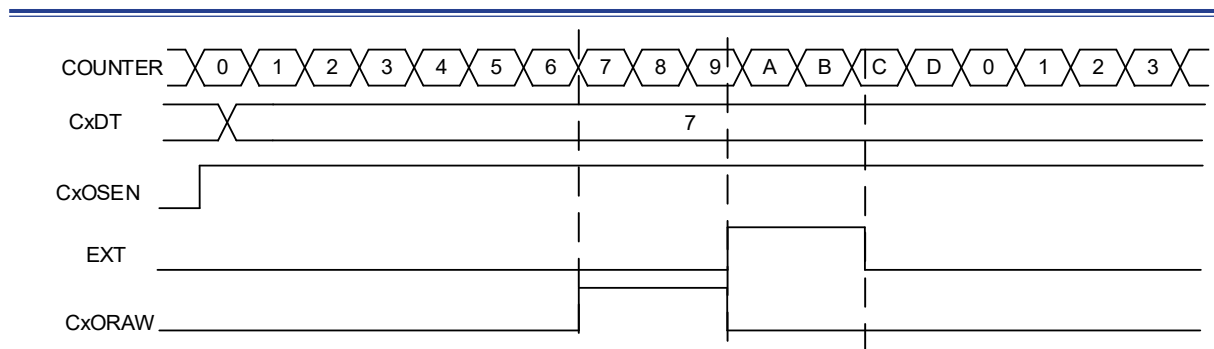
- PTOS = 3'b000, TRGOUT output software overflow event (OVFSWTR bit in TMRx\_SWEVT register) or reset event
- PTOS = 3'b001, TRGOUT output counter enable
- PTOS = 3'b010, TRGOUT output counter overflow event
- PTOS = 3'b011, TRGOUT output capture and compare event
- PTOS = 3'b100, TRGOUT output C1ORAW

- PTOS = 3'b101, TRGOUT output C2ORAW
- PTOS = 3'b110, TRGOUT output C3ORAW
- PTOS = 3'b111, TRGOUT output C4ORAW

#### CxORAW clear

When the CxOSEN bit is set to 1, the CxORAW signal for a given channel is cleared by applying a high level to the EXT input. The CxORAW signal remains unchanged until the next overflow event.

This function can only be used in output capture or PWM modes, and does not work in forced mode. Figure 117 shows the example of clearing CxORAW signal. When the EXT input is high, the CxORAW signal, which was originally high, is driven low; when the EXT is low, the CxORAW signal outputs the corresponding level according to the comparison result between the counter value and CxDT value.



**Figure 117. Clearing CxORAW (PWM mode A) by EXT input**

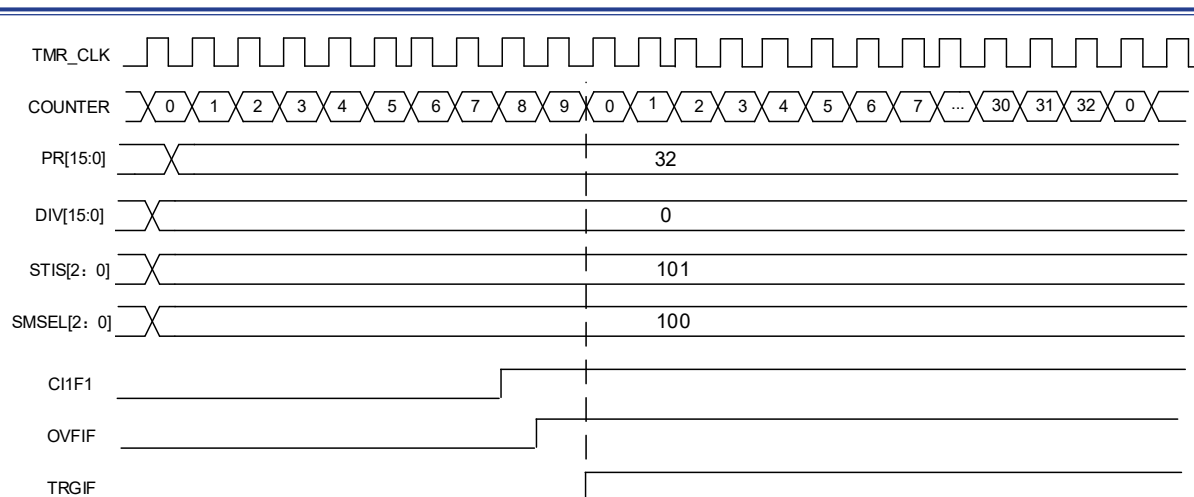
#### TMR synchronization

The timers are linked together internally for timer synchronization. Master timer is selected by setting the PTOS[2:0] bit; Slave timer is selected by setting the SMSEL[2:0] bit.

Slave mode include:

### Slave mode: Reset mode

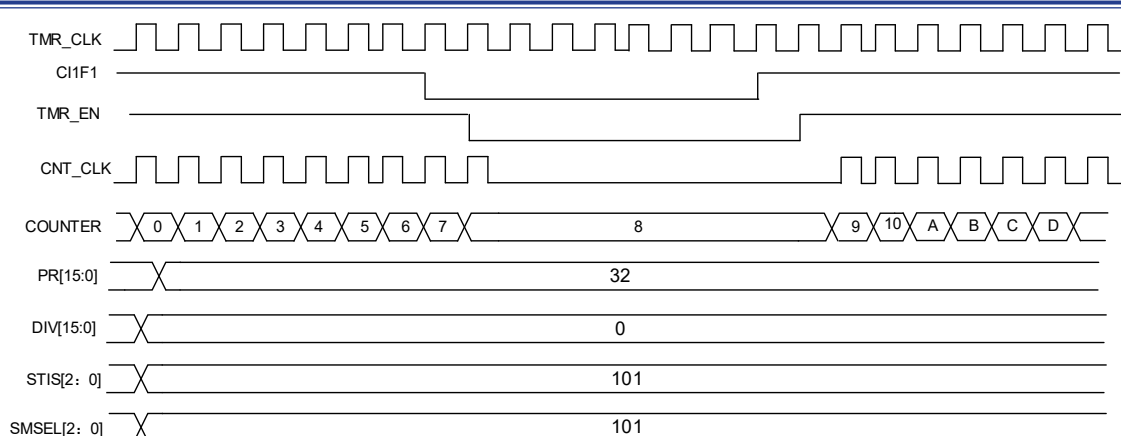
The counter and its prescaler can be reset by a selected trigger signal. An overflow event is generated when OVFS = 0.



**Figure 118. Example of reset mode**

### Slave mode: Suspend mode

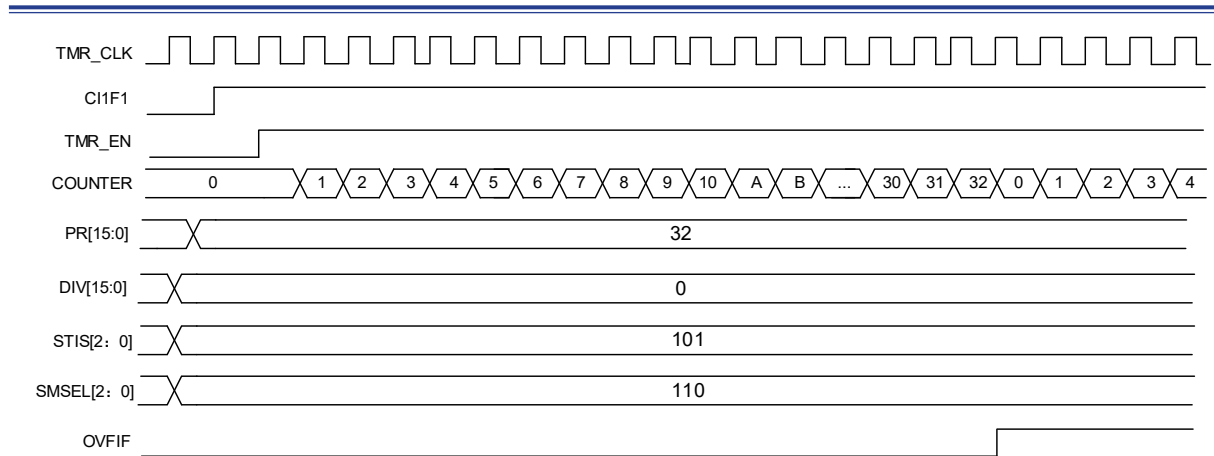
In this mode, the counter is controlled by a selected trigger input. The counter starts counting when the trigger input is high and stops as soon as the trigger input is low.



**Figure 119. Example of suspend mode**

### Slave mode: Trigger mode

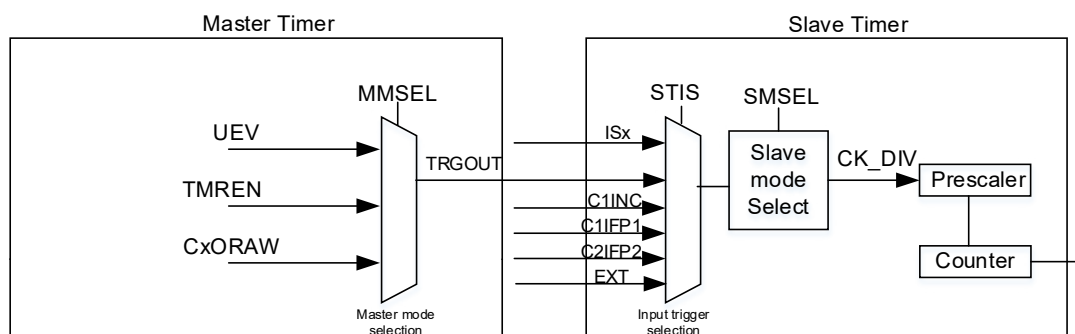
The counter can start counting on the rising edge of a selected trigger input (TMR\_EN = 1).



**Figure 120. Example of trigger mode**

### Master/slave timer interconnection

Both Master and slave timer can be configured in different master and slave modes respectively. The combination of both them can be used for various purposes. Figure 121 provides an example of interconnection between master timer and slave timer.



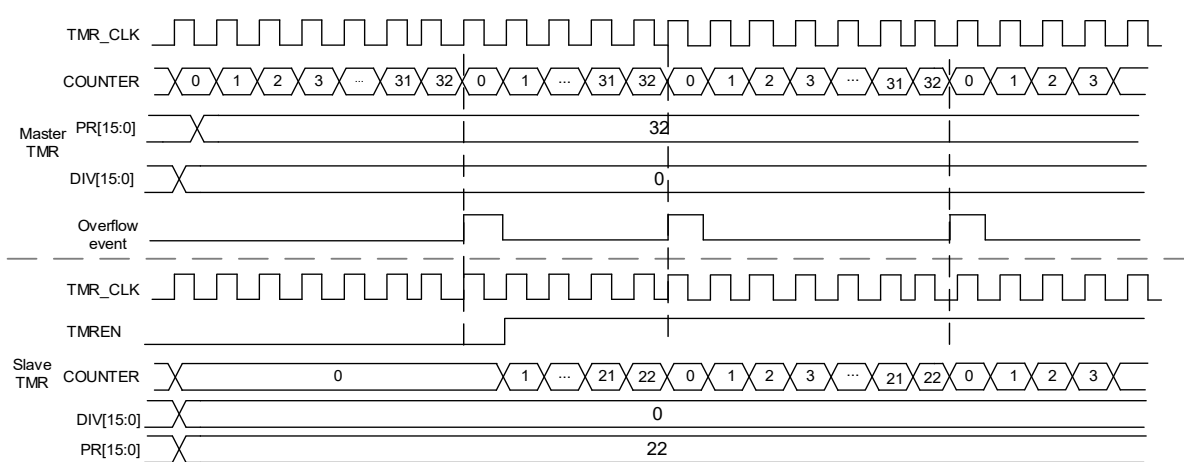
**Figure 121. Master/slave timer connection**

### Using master timer to clock the slave timer:

- Configure master timer output signal TRGOUT as an overflow event (PTOS[2:0] = 3'b010). The master timer outputs a pulse signal at each counter overflow event, which is used as the counting clock of the slave timer.
- Configure the master timer counting period (TMRx\_PR registers).
- Configure the slave timer trigger input signal TRGIN as master timer output (STIS[2:0] in the TMRx\_STCTRL register).
- Configure the slave timer to use external clock mode A (SMSEL[2:0] = 3'b111 in the TMRx\_STCTRL register).
- Set TMREN = 1 in both master timer and slave timer to enable them.

#### Using master timer to start slave timer:

- Configure master timer output signal TRGOUT as an overflow event (PTOS[2:0] = 3'b010). The master timer outputs a pulse signal at each counter overflow event, which is used as the counting clock of the slave timer.
- Configure master timer counting period (TMRx\_PR registers).
- Configure slave timer trigger input signal TRGIN as master timer input.
- Configure slave timer as trigger mode (SMSSEL = 3'b110 in the TMR2\_STCTRL register).
- Set TMREN = 1 to enable master timer.

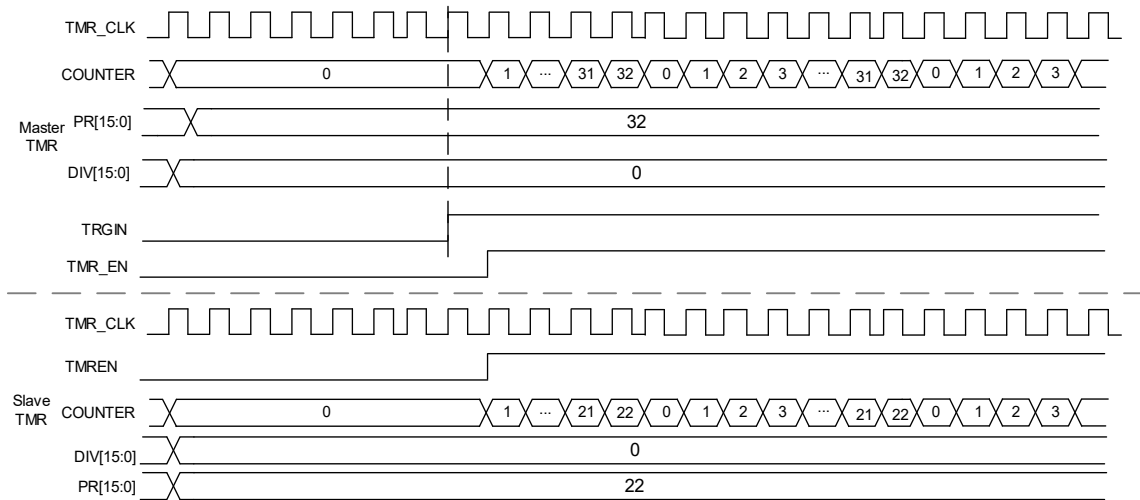


**Figure 122. Using master timer to start slave timer**

#### Starting master and slave timers synchronously by an external trigger:

In this example, configure the master timer as master/slave mode synchronously and enable its slave timer synchronization function. This mode is used for synchronization between master timer and slave timer.

- Set the STS bit of the master timer.
- Configure master timer output signal TRGOUT as an overflow event (PTOS[2:0] = 3'b010). The master timer outputs a pulse signal at each counter overflow event, which is used as the counting clock of the slave timer.
- Configure the slave timer mode of the master timer as trigger mode, and select C1IN as trigger source.
- Configure slave timer trigger input signal TRGIN as master timer output.
- Configure slave timer as trigger mode (SMSSEL = 3'b110 in the TMR2\_STCTRL register).



**Figure 123. Starting master and slave timers synchronously by an external trigger**

### Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the TMRx counter stops counting by setting the TMRx\_PAUSE in the DEBUG module.

### TMR2 and TMR3 registers

These peripheral registers must be accessed by word (32 bits).

TMR2 and TMR3 register are mapped into a 32-bit addressable space.

**Table 50. TMR2 and TMR3 register map and reset value**

| Register     | Offset | Reset value |
|--------------|--------|-------------|
| TMRx_CTRL1   | 0x00   | 0x0000      |
| TMRx_CTRL2   | 0x04   | 0x0000      |
| TMRx_STCTRL  | 0x08   | 0x0000      |
| TMRx_IDEN    | 0x0C   | 0x0000      |
| TMRx_ISTS    | 0x10   | 0x0000      |
| TMRx_SWEVT   | 0x14   | 0x0000      |
| TMRx_CM1     | 0x18   | 0x0000      |
| TMRx_CM2     | 0x1C   | 0x0000      |
| TMRx_CCTRL   | 0x20   | 0x0000      |
| TMRx_CVAL    | 0x24   | 0x0000      |
| TMRx_DIV     | 0x28   | 0x0000      |
| TMRx_PR      | 0x2C   | 0x0000      |
| TMRx_C1DT    | 0x34   | 0x0000      |
| TMRx_C2DT    | 0x38   | 0x0000      |
| TMRx_C3DT    | 0x3C   | 0x0000      |
| TMRx_C4DT    | 0x40   | 0x0000      |
| TMRx_DMACTRL | 0x48   | 0x0000      |
| TMRx_DMADT   | 0x4C   | 0x0000      |

### TMR2 and TMR3 control register 1 (TMRx\_CTRL1)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:11 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 10    | PMEN     | 0x0         | rw   | <p>Plus Mode Enable</p> <p>This bit is used to enable TMRx plus mode. In this mode, TMRx_CVAL, TMRx_PR and TMRx_CxDT are extended from 16-bit to 32-bit.</p> <p>0: Disabled<br/>1: Enabled</p> <p>Note: This function is only valid for TMR2 and TMR5. It is not applicable to other TMRs.</p> <p>In plus mode or when disabled, only 16-bit value can be written to TMRx_CVAL, TMRx_PR and TMRx_CxDT registers.</p>   |
| Bit 9:8   | CLKDIV   | 0x0         | rw   | <p>Clock division</p> <p>This field is used to define the relationship between digital filter sampling frequency (<math>f_{DTS}</math>) and timer clock frequency (<math>f_{CK\_INT}</math>).</p> <p>00: No division, <math>f_{DTS} = f_{CK\_INT}</math><br/>01: Divided by 2, <math>f_{DTS} = f_{CK\_INT}/2</math><br/>10: Divided by 4, <math>f_{DTS} = f_{CK\_INT}/4</math><br/>11: Reserved</p>  |
| Bit 7     | PRBEN    | 0x0         | rw   | <p>Period buffer enable</p> <p>0: Period buffer is disabled<br/>1: Period buffer is enabled</p>  |
| Bit 6:5   | TWCMSEL  | 0x0         | rw   | <p>Two-way counting mode selection</p> <p>00: One-way counting mode, depending on the OWCDIR bit<br/>01: Two-way counting mode 1, count up and down alternately, the CxIF bit is set only when the counter counts down<br/>10: Two-way counting mode 2, count up and down alternately, the CxIF bit is set only when the counter counts up<br/>11: Two-way counting mode 3, count up and down alternately, the CxIF bit is set when the counter counts up / down</p> |
| Bit 4     | OWCDIR   | 0x0         | rw   | <p>One-way count direction</p> <p>0: Up<br/>1: Down</p>  |
| Bit 3     | OCMEN    | 0x0         | rw   | <p>One cycle mode enable</p> <p>This bit is use to select whether to stop counting at an overflow event</p> <p>0: The counter does not stop at an overflow event<br/>1: The counter stops at an overflow event</p>   |
| Bit 2     | OVFS     | 0x0         | rw   | <p>Overflow event source</p> <p>This bit is used to select overflow event or DMA request sources.</p> <p>0: Counter overflow, setting the OVFSWTR bit or overflow event generated by slave timer controller<br/>1: Only counter overflow generates an overflow event</p>   |
| Bit 1     | OVFEN    | 0x0         | rw   | <p>Overflow event enable</p> <p>0: Enabled<br/>1: Disabled</p>   |
| Bit 0     | TMREN    | 0x0         | rw   | <p>TMR enable</p> <p>0: Disabled<br/>1: Enabled</p>  |

### TMR2 and TMR3 control register 2 (TMRx\_CTRL2)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:8 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 7    | C1INSEL  | 0x0         | rw   | C1IN selection<br>0: CH1 pin is connected to C1IRAW input<br>1: The XOR result of CH1, CH2 and CH3 pins is connected to C1IRAW input   |
| Bit 6:4  | PTOS     | 0x0         | rw   | Master TMR output selection<br>This field is used to select the TMRx signal sent to the slave timer.<br>000: Software overflow or Reset<br>001: Enable<br>010: Reset<br>011: Compare pulse<br>100: C1ORAW signal<br>101: C2ORAW signal<br>110: C3ORAW signal<br>111: C4ORAW signal |
| Bit 3    | DRS      | 0x0         | rw   | DMA request source<br>0: Capture/compare event<br>1: Overflow event  |
| Bit 2:0  | Reserved | 0x0         | resd | Kept at its default value.   |

### TMR2 and TMR3 slave timer control register (TMRx\_STCTRL)

| Bit       | Name   | Reset value | Type | Description  |
|-----------|--------|-------------|------|--|
| Bit 15    | ESP    | 0x0         | rw   | External signal polarity<br>0: High or rising edge<br>1: Low or falling edge   |
| Bit 14    | ECMBEN | 0x0         | rw   | External clock mode B enable<br>This bit is used to enable external clock mode B<br>0: Disabled<br>1: Enabled  |
| Bit 13:12 | ESDIV  | 0x0         | rw   | External signal divide<br>This field is used to select the frequency division of an external trigger<br>00: Normal<br>01: Divided by 2<br>10: Divided by 4<br>11: Divided by 8 |



| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 11:8 | ESF      | 0x0         | rw   | <p>External signal filter</p> <p>This field is used to filter an external signal. The external signal can be sampled only after it has been generated N times</p> <p>0000: No filter, sampling by <math>f_{DTS}</math></p> <p>0001: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 2</math></p> <p>0010: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 4</math></p> <p>0011: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 8</math></p> <p>0100: <math>f_{SAMPLING} = f_{DTS}/2</math>, <math>N = 6</math></p> <p>0101: <math>f_{SAMPLING} = f_{DTS}/2</math>, <math>N = 8</math></p> <p>0110: <math>f_{SAMPLING} = f_{DTS}/4</math>, <math>N = 6</math></p> <p>0111: <math>f_{SAMPLING} = f_{DTS}/4</math>, <math>N = 8</math></p> <p>1000: <math>f_{SAMPLING} = f_{DTS}/8</math>, <math>N = 6</math></p> <p>1001: <math>f_{SAMPLING} = f_{DTS}/8</math>, <math>N = 8</math></p> <p>1010: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 6</math></p> <p>1011: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 8</math></p> <p>1100: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 8</math></p> <p>1101: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 5</math></p> <p>1110: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 6</math></p> <p>1111: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 8</math></p> |
| Bit 7    | STS      | 0x0         | rw   | <p>Subordinate TMR synchronization</p> <p>If enabled, master and slave timer can be synchronized.</p> <p>0: Disabled</p> <p>1: Enabled</p>  |
| Bit 6:4  | STIS     | 0x0         | rw   | <p>Subordinate TMR input selection</p> <p>This field is used to select the subordinate TMR input.</p> <p>000: Internal selection 0 (IS0)</p> <p>001: Internal selection 1 (IS1)</p> <p>010: Internal selection 2 (IS2)</p> <p>011: Internal selection 3 (IS3)</p> <p>100: C1IRAW input detector (C1INC)</p> <p>101: Filtered input 1 (C1IF1)</p> <p>110: Filtered input 2 (C1IF2)</p> <p>111: External input (EXT)</p> <p>Please refer to Table 48 for more information on ISx for each timer.</p>  |
| Bit 3    | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 2:0  | SMSEL    | 0x0         | rw   | <p>Subordinate TMR mode selection</p> <p>000: Slave mode is disabled</p> <p>001: Encoder mode A</p> <p>010: Encoder mode B</p> <p>011: Encoder mode C</p> <p>100: Reset mode – Rising edge of the TRGIN input reinitializes the counter</p> <p>101: Suspend mode – The counter starts counting when the TRGIN is high</p> <p>110: Trigger mode – A trigger event is generated at the rising edge of the TRGIN input</p> <p>111: External clock mode A – Rising edge of the TRGIN input clocks the counter</p> <p>Note: Please refer to count mode section for the details on encoder mode A/B/C.</p>  |

### TMR2 and TMR3 DMA/interrupt enable register (TMRx\_IDEN)

| Bit    | Name     | Reset value | Type | Description  |
|--------|----------|-------------|------|--|
| Bit 15 | Reserved | 0x0         | resd | Kept at its default value                                      |
| Bit 14 | TDEN     | 0x0         | rw   | Trigger DMA request enable<br>0: Disabled<br>1: Enabled        |
| Bit 13 | Reserved | 0x0         | resd | Kept at its default value                                      |
| Bit 12 | C4DEN    | 0x0         | rw   | Channel 4 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 11 | C3DEN    | 0x0         | rw   | Channel 3 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 10 | C2DEN    | 0x0         | rw   | Channel 2 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 9  | C1DEN    | 0x0         | rw   | Channel 1 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 8  | OVFDEN   | 0x0         | rw   | Overflow event DMA request enable<br>0: Disabled<br>1: Enabled |
| Bit 7  | Reserved | 0x0         | resd | Kept at its default value                                      |
| Bit 6  | TIEN     | 0x0         | rw   | Trigger interrupt enable<br>0: Disabled<br>1: Enabled          |
| Bit 5  | Reserved | 0x0         | resd | Kept at its default value                                      |
| Bit 4  | C4IEN    | 0x0         | rw   | Channel 4 interrupt enable<br>0: Disabled<br>1: Enabled        |
| Bit 3  | C3IEN    | 0x0         | rw   | Channel 3 interrupt enable<br>0: Disabled<br>1: Enabled        |
| Bit 2  | C2IEN    | 0x0         | rw   | Channel 2 interrupt enable<br>0: Disabled<br>1: Enabled        |
| Bit 1  | C1IEN    | 0x0         | rw   | Channel 1 interrupt enable<br>0: Disabled<br>1: Enabled        |
| Bit 0  | OVFIEN   | 0x0         | rw   | Overflow interrupt enable<br>0: Disabled<br>1: Enabled         |

### TMR2 and TMR3 interrupt status register (TMRx\_ISTS)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15:13 | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 12    | C4RF     | 0x0         | rw0c | Channel 4 recapture flag<br>Please refer to C1RF description.   |
| Bit 11    | C3RF     | 0x0         | rw0c | Channel 3 recapture flag<br>Please refer to C1RF description.   |
| Bit 10    | C2RF     | 0x0         | rw0c | Channel 2 recapture flag<br>Please refer to C1RF description.   |
| Bit 9     | C1RF     | 0x0         | rw0c | Channel 1 recapture flag<br>This bit indicates whether a recapture is detected when C1IF = 1. This bit is set by hardware, and cleared by writing "0".<br>0: No capture is detected<br>1: Capture is detected.  |
| Bit 8:7   | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 6     | TRGIF    | 0x0         | rw0c | Trigger interrupt flag<br>This bit is set by hardware on a trigger event. It is cleared by writing "0".<br>0: No trigger event occurs<br>1: Trigger event is generated.<br>Trigger event: an active edge is detected on TRGIN input, or any edge in suspend mode.   |
| Bit 5     | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 4     | C4IF     | 0x0         | rw0c | Channel 4 interrupt flag<br>Please refer to C1IF description.   |
| Bit 3     | C3IF     | 0x0         | rw0c | Channel 3 interrupt flag<br>Please refer to C1IF description.   |
| Bit 2     | C2IF     | 0x0         | rw0c | Channel 2 interrupt flag<br>Please refer to C1IF description.   |
| Bit 1     | C1IF     | 0x0         | rw0c | Channel 1 interrupt flag<br>If the channel 1 is configured as input mode:<br>This bit is set by hardware on a capture event. It is cleared by software or read access to the TMRx_C1DT.<br>0: No capture event occurs<br>1: Capture event is generated<br>If the channel 1 is configured as output mode:<br>This bit is set by hardware on a compare event. It is cleared by software.<br>0: No compare event occurs<br>1: Compare event is generated |
| Bit 0     | OVFIF    | 0x0         | rw0c | Overflow interrupt flag<br>This bit is set by hardware on an overflow event. It is cleared by software.<br>0: No overflow event occurs<br>1: Overflow event is generated. If OVFEN = 0 and OVFS = 0 in the TMRx_CTRL1 register:<br>– An overflow event is generated when OVFG = 1 in the TMRx_SWEVE register;<br>– An overflow event is generated when the counter CVAL is reinitialized by a trigger event.  |

### TMR2 and TMR3 software event register (TMRx\_SWEVT)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:7 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 6    | TRGSWTR  | 0x0         | rw   | Trigger event triggered by software<br>This bit is set by software to generate a trigger event.<br>0: No effect<br>1: Generate a trigger event.       |
| Bit 5    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 4    | C4SWTR   | 0x0         | wo   | Channel 4 event triggered by software<br>Please refer to C1M description.   |
| Bit 3    | C3SWTR   | 0x0         | wo   | Channel 3 event triggered by software<br>Please refer to C1M description.   |
| Bit 2    | C2SWTR   | 0x0         | wo   | Channel 2 event triggered by software<br>Please refer to C1M description.   |
| Bit 1    | C1SWTR   | 0x0         | wo   | Channel 1 event triggered by software<br>This bit is set by software to generate a channel 1 event.<br>0: No effect<br>1: Generate a channel 1 event. |
| Bit 0    | OVFSWTR  | 0x0         | wo   | Overflow event triggered by software<br>This bit is set by software to generate an overflow event.<br>0: No effect<br>1: Generate an overflow event.  |

### TMR2 and TMR3 channel mode register1 (TMRx\_CM1)

#### Output compare mode:

| Bit       | Name    | Reset value | Type | Description  |
|-----------|---------|-------------|------|--|
| Bit 15    | C2OSEN  | 0x0         | rw   | Channel 2 output switch enable   |
| Bit 14:12 | C2OCTRL | 0x0         | rw   | Channel 2 output control   |
| Bit 11    | C2OBEN  | 0x0         | rw   | Channel 2 output buffer enable   |
| Bit 10    | C2OIEN  | 0x0         | rw   | Channel 2 output enable immediately  |
| Bit 9:8   | C2C     | 0x0         | rw   | Channel 2 configuration<br>This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN = '0':<br>00: Output<br>01: Input, C2IN is mapped on C2IFP2<br>10: Input, C2IN is mapped on C1IFP2<br>11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS register. |
| Bit 7     | C1OSEN  | 0x0         | rw   | Channel 1 output switch enable<br>0: C1ORAW is not affected by EXT<br>1: Once high level is detect on EXT input, clear C1ORAW.   |

| Bit     | Name    | Reset value | Type | Description  |
|---------|---------|-------------|------|--|
| Bit 6:4 | C1OCTRL | 0x0         | rw   | <p>Channel 1 output control</p> <p>This field defines the behavior of the original signal C1ORAW.</p> <p>000: Disconnected. C1ORAW is disconnected from C1OUT;</p> <p>001: C1ORAW is high when TMRx_CVAL = TMRx_C1DT</p> <p>010: C1ORAW is low when TMRx_CVAL = TMRx_C1DT</p> <p>011: Switch C1ORAW level when TMRx_CVAL = TMRx_C1DT</p> <p>100: C1ORAW is forced low</p> <p>101: C1ORAW is forced high.</p> <p>110: PWM mode A</p> <ul style="list-style-type: none"> <li>– OWCDIR = 0, C1ORAW is high once TMRx_C1DT &gt; TMRx_CVAL, else low;</li> <li>– OWCDIR = 1, C1ORAW is low once TMRx_C1DT &lt; TMRx_CVAL, else high;</li> </ul> <p>111: PWM mode B</p> <ul style="list-style-type: none"> <li>– OWCDIR = 0, C1ORAW is low once TMRx_C1DT &gt; TMRx_CVAL, else high;</li> <li>– OWCDIR = 1, C1ORAW is high once TMRx_C1DT &lt; TMRx_CVAL, else low.</li> </ul> <p>Note: In the configurations other than 000', the C1OUT is connected to C1ORAW. The C1OUT output level is not only subject to the changes of C1ORAW, but also the output polarity set by CCTRL.</p> |
| Bit 3   | C1OBEN  | 0x0         | rw   | <p>Channel 1 output buffer enable</p> <p>0: Buffer function of TMRx_C1DT is disabled. The new value written to the TMRx_C1DT takes effect immediately.</p> <p>1: Buffer function of TMRx_C1DT is enabled. The value to be written to the TMRx_C1DT is stored in the buffer register, and can be sent to the TMRx_C1DT register only on an overflow event.</p>  |
| Bit 2   | C1OIEN  | 0x0         | rw   | <p>Channel 1 output enable immediately</p> <p>In PWM mode A or B, this bit is used to accelerate the channel 1 output's response to the trigger event.</p> <p>0: Need to compare the CVAL with C1DT before generating an output</p> <p>1: No need to compare the CVAL and C1DT. An output is generated immediately when a trigger event occurs.</p>  |
| Bit 1:0 | C1C     | 0x0         | rw   | <p>Channel 1 configuration</p> <p>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':</p> <p>00: Output</p> <p>01: Input, C1IN is mapped on C1IFP1</p> <p>10: Input, C1IN is mapped on C2IFP1</p> <p>11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.</p>   |

**Input capture mode:**

| Bit       | Name   | Reset value | Type | Description  |
|-----------|--------|-------------|------|--|
| Bit 15:12 | C2DF   | 0x0         | rw   | Channel 2 digital filter   |
| Bit 11:10 | C2IDIV | 0x0         | rw   | Channel 2 input divider  |
| Bit 9: 8  | C2C    | 0x0         | rw   | Channel 2 configuration<br>This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN = '0':<br>00: Output<br>01: Input, C2IN is mapped on C2IFP2<br>10: Input, C2IN is mapped on C1IFP2<br>11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.  |
| Bit 7: 4  | C1DF   | 0x0         | rw   | Channel 1 digital filter<br>This field defines the digital filter of the channel 1. N stands for the number of filtering, indicating that the input edge can pass the filter only after N sampling events.<br>0000: No filter, sampling is done at $f_{DTS}$<br>0001: $f_{SAMPLING} = f_{CK\_INT}$ , N = 2<br>0010: $f_{SAMPLING} = f_{CK\_INT}$ , N = 4<br>0011: $f_{SAMPLING} = f_{CK\_INT}$ , N = 8<br>0100: $f_{SAMPLING} = f_{DTS}/2$ , N = 6<br>0101: $f_{SAMPLING} = f_{DTS}/2$ , N = 8<br>0110: $f_{SAMPLING} = f_{DTS}/4$ , N = 6<br>0111: $f_{SAMPLING} = f_{DTS}/4$ , N = 8<br>1000: $f_{SAMPLING} = f_{DTS}/8$ , N = 6<br>1001: $f_{SAMPLING} = f_{DTS}/8$ , N = 8<br>1010: $f_{SAMPLING} = f_{DTS}/16$ , N = 6<br>1011: $f_{SAMPLING} = f_{DTS}/16$ , N = 8<br>1100: $f_{SAMPLING} = f_{DTS}/16$ , N = 8<br>1101: $f_{SAMPLING} = f_{DTS}/32$ , N = 5<br>1110: $f_{SAMPLING} = f_{DTS}/32$ , N = 6<br>1111: $f_{SAMPLING} = f_{DTS}/32$ , N = 8 |
| Bit 3: 2  | C1IDIV | 0x0         | rw   | Channel 1 input divider<br>This field defines Channel 1 input divider.<br>00: No divider. An input capture is generated at each active edge.<br>01: An input compare is generated every 2 active edges<br>10: An input compare is generated every 4 active edges<br>11: An input compare is generated every 8 active edges<br>Note: the divider is reset once C1EN = '0'   |
| Bit 1: 0  | C1C    | 0x0         | rw   | Channel 1 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':<br>00: Output<br>01: Input, C1IN is mapped on C1IFP1<br>10: Input, C1IN is mapped on C2IFP1<br>11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.  |

## TMR2 and TMR3 channel mode register2 (TMRx\_CM2)

### Output compare mode:

| Bit       | Name    | Reset value | Type | Description   |
|-----------|---------|-------------|------|---|
| Bit 15    | C4OSEN  | 0x0         | rw   | Channel 4 output switch enable  |
| Bit 14:12 | C4OCTRL | 0x0         | rw   | Channel 4 output control  |
| Bit 11    | C4OBEN  | 0x0         | rw   | Channel 4 output buffer enable  |
| Bit 10    | C4OIEN  | 0x0         | rw   | Channel 4 output enable immediately   |
| Bit 9: 8  | C4C     | 0x0         | rw   | Channel 4 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C4EN = '0':<br>00: Output<br>01: Input, C4IN is mapped on C4IFP4<br>10: Input, C4IN is mapped on C3IFP4<br>11: Input, C4IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS. |
| Bit 7     | C3OSEN  | 0x0         | rw   | Channel 3 output switch enable  |
| Bit 6:4   | C3OCTRL | 0x0         | rw   | Channel 3 output control  |
| Bit 3     | C3OBEN  | 0x0         | rw   | Channel 3 output buffer enable  |
| Bit 2     | C3OIEN  | 0x0         | rw   | Channel 3 output enable immediately   |
| Bit 1:0   | C3C     | 0x0         | rw   | Channel 3 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C3EN = '0':<br>00: Output<br>01: Input, C3IN is mapped on C3IFP3<br>10: Input, C3IN is mapped on C4IFP3<br>11: Input, C3IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS. |

### Input capture mode:

| Bit       | Name   | Reset value | Type | Description   |
|-----------|--------|-------------|------|---|
| Bit 15:12 | C4DF   | 0x0         | rw   | Channel 4 digital filter  |
| Bit 11:10 | C4IDIV | 0x0         | rw   | Channel 4 input divider   |
| Bit 9:8   | C4C    | 0x0         | rw   | Channel 4 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C4EN = '0':<br>00: Output<br>01: Input, C4IN is mapped on C4IFP4<br>10: Input, C4IN is mapped on C3IFP4<br>11: Input, C4IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS. |
| Bit 7:4   | C3DF   | 0x0         | rw   | Channel 3 digital filter  |
| Bit 3:2   | C3IDIV | 0x0         | rw   | Channel 3 input divider   |
| Bit 1:0   | C3C    | 0x0         | rw   | Channel 3 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C3EN = '0':<br>00: Output<br>01: Input, C3IN is mapped on C3IFP3<br>10: Input, C3IN is mapped on C4IFP3<br>11: Input, C3IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS. |

**TMR2 and TMR3 channel control register (TMRx\_CTRL)**

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15:14 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 13    | C4P      | 0x0         | rw   | Channel 4 polarity<br>Please refer to C1P description.  |
| Bit 12    | C4EN     | 0x0         | rw   | Channel 4 enable<br>Please refer to C1EN description.   |
| Bit 11:10 | Reserved | 0x0         | resd | Default value   |
| Bit 9     | C3P      | 0x0         | rw   | Channel 3 polarity<br>Please refer to C1P description.  |
| Bit 8     | C3EN     | 0x0         | rw   | Channel 3 enable<br>Please refer to C1EN description.   |
| Bit 7:6   | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 5     | C2P      | 0x0         | rw   | Channel 2 polarity<br>Please refer to C1P description.  |
| Bit 4     | C2EN     | 0x0         | rw   | Channel 2 enable<br>Please refer to C1EN description.   |
| Bit 3:2   | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 1     | C1P      | 0x0         | rw   | Channel 1 polarity<br>When the channel 1 is configured as output mode:<br>0: C1OUT is active high<br>1: C1OUT is active low<br>When the channel 1 is configured as input mode:<br>0: C1IN active edge is on its rising edge. When used as external trigger, C1IN is not inverted.<br>1: C1IN active edge is on its falling edge. When used as external trigger, C1IN is inverted. |
| Bit0      | C1EN     | 0x0         | rw   | Channel 1 enable<br>0: Input or output is disabled<br>1: Input or output is enabled   |

**Table 51. Standard CxOUT channel output control bit**

| CxEN bit | CxOUT output state                     |
|----------|--|
| 0        | Output disabled (CxOUT = 0, Cx_EN = 0) |
| 1        | CxOUT = CxORAW + polarity, Cx_EN = 1   |

Note: The state of the external I/O pins connected to the standard CxOUT channel depends on the CxOUT channel state and the GPIO and IOMUX registers.

**TMR2 and TMR3 counter value (TMRx\_CVAL)**

| Bit       | Name | Reset value | Type | Description   |
|-----------|------|-------------|------|---|
| Bit 31:16 | CVAL | 0x0         | rw   | Counter value<br>When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the CVAL is expanded to 32 bits. |
| Bit 15:0  | CVAL | 0x0         | rw   | Counter value   |



#### TMR2 and TMR3 division value (TMRx\_DIV)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 15:0 | DIV  | 0x0         | rw   | Divider value<br>The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK} / (DIV[15:0] + 1)$ .<br>DIV contains the value written at an overflow event. |

#### TMR2 and TMR3 period register (TMRx\_PR)

| Bit       | Name | Reset value | Type | Description  |
|-----------|------|-------------|------|--|
| Bit 31:16 | PR   | 0x0         | rw   | Period value<br>When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the PR is expanded to 32 bits. |
| Bit 15:0  | PR   | 0x0         | rw   | Period value<br>This defines the period value of the TMRx counter. The timer stops working when the period value is 0.       |

#### TMR2 and TMR3 channel 1 data register (TMRx\_C1DT)

| Bit       | Name | Reset value | Type | Description   |
|-----------|------|-------------|------|---|
| Bit 31:16 | C1DT | 0x0         | rw   | Channel 1 data register<br>When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the C1DT is expanded to 32 bits.   |
| Bit 15:0  | C1DT | 0x0         | rw   | Channel 1 data register<br>When the channel 1 is configured as input mode:<br>The C1DT is the CVAL value stored by the last channel 1 input event (C1IN).<br>When the channel 1 is configured as output mode:<br>C1DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C1OBEN bit, and the corresponding output is generated on C1OUT as configured. |

#### TMR2 and TMR3 channel 2 data register (TMRx\_C2DT)

| Bit       | Name | Reset value | Type | Description   |
|-----------|------|-------------|------|---|
| Bit 31:16 | C2DT | 0x0         | rw   | Channel 2 data register<br>When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the C2DT is expanded to 32 bits.   |
| Bit 15:0  | C2DT | 0x0         | rw   | Channel 2 data register<br>When the channel 2 is configured as input mode:<br>The C2DT is the CVAL value stored by the last channel 2 input event (C1IN).<br>When the channel 2 is configured as output mode:<br>C2DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C2OBEN bit, and the corresponding output is generated on C2OUT as configured. |

### TMR2 and TMR3 channel 3 data register (TMRx\_C3DT)

| Bit       | Name | Reset value | Type | Description   |
|-----------|------|-------------|------|---|
| Bit 31:16 | C3DT | 0x0         | rw   | Channel 3 data register<br>When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the C3DT is expanded to 32 bits.   |
| Bit 15:0  | C3DT | 0x0         | rw   | Channel 3 data register<br>When the channel 3 is configured as input mode:<br>The C3DT is the CVAL value stored by the last channel 3 input event (C1IN).<br>When the channel 3 is configured as output mode:<br>C3DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C3OBEN bit, and the corresponding output is generated on C3OUT as configured. |

### TMR2 and TMR3 channel 4 data register (TMRx\_C4DT)

| Bit       | Name | Reset value | Type | Description   |
|-----------|------|-------------|------|---|
| Bit 31:16 | C4DT | 0x0         | rw   | Channel 4 data register<br>When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the C4DT is expanded to 32 bits.   |
| Bit 15:0  | C4DT | 0x0         | rw   | Channel 4 data register<br>When the channel 4 is configured as input mode:<br>The C4DT is the CVAL value stored by the last channel 4 input event (C1IN).<br>When the channel 4 is configured as output mode:<br>C4DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C4OBEN bit, and the corresponding output is generated on C4OUT as configured. |

### TMR2 and TMR3 DMA control register (TMRx\_DMACTRL)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:13 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 12:8  | DTB      | 0x0         | rw   | DMA transfer bytes<br>This field defines the number of DMA transfers:<br>00000: 1 byte      00001: 2 bytes<br>00010: 3 bytes    00011: 4 bytes<br>.....<br>10000: 17 bytes   10001: 18 bytes |
| Bit 7:5   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 4:0   | ADDR     | 0x0         | rw   | DMA transfer address offset<br>ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register.<br>00000: TMRx_CTRL1<br>00001: TMRx_CTRL2<br>00010: TMRx_STCTRL<br>.....   |

### TMR2 and TMR3 DMA data register (TMRx\_DMATD)

| Bit      | Name  | Reset value | Type | Description  |
|----------|-------|-------------|------|--|
| Bit 15:0 | DMADT | 0x0         | rw   | DMA data register<br>A read or write operation to the DMADT register accesses the TMR registers at the following address:<br>TMRx peripheral address + ADDR × 4 to TMRx peripheral address + ADDR × 4 + DTB × 4. |

## General-purpose timer (TMR9 to TMR14)

### TMR13 and TMR14 introduction

The general-purpose timer (TMR13 and TMR14) consists of a 16-bit counter supporting upcounting mode. These timers can be synchronized.

### TMR13 and TMR14 main features

The main functions of general-purpose TMR13 and TMR14 include:

- Source of counter clock: internal clock
- 16-bit upcounter
- 1x independent channels for input capture, output compare, PWM generation
- Synchronization control between master and slave timers
- Interrupt is generated at overflow and channel events

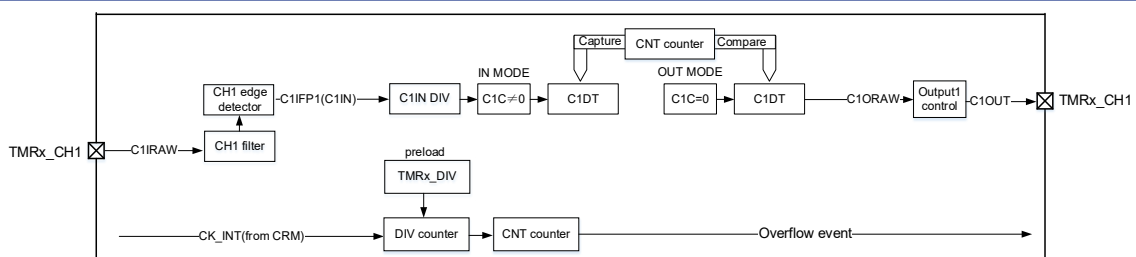


Figure 124. Block diagram of general-purpose TMR13/14

### TMR13 and TMR14 function overview

#### Counting clock

TMR13/14 counting clock can be provided by internal clock source (CK\_INT).

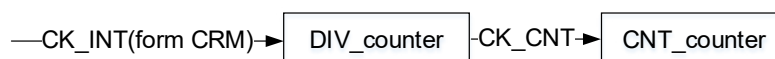


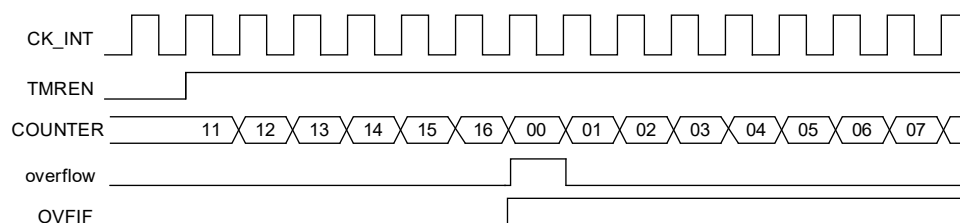
Figure 125. Counting clock

### Internal clock (CK\_INT)

By default, the CK\_INT divided by the prescaler is used to drive the counter to start counting. When TMR's APB clock prescaler factor is 1, the CK\_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

Follow the configuration steps below:

- Set counting frequency through TMRx\_DIV register
- Set counting cycles through TMRx\_PR register
- Enable a counter by setting the TMREN bit in the TMRx\_CTRL1 register



**Figure 126. Control circuit with CK\_INT, TMRx\_DIV = 0x0 and TMRx\_PR = 0x16**

### Counting mode

The general-purpose timer (TMR13 and TMR14) consists of a 16-bit counter supporting upcounting mode.

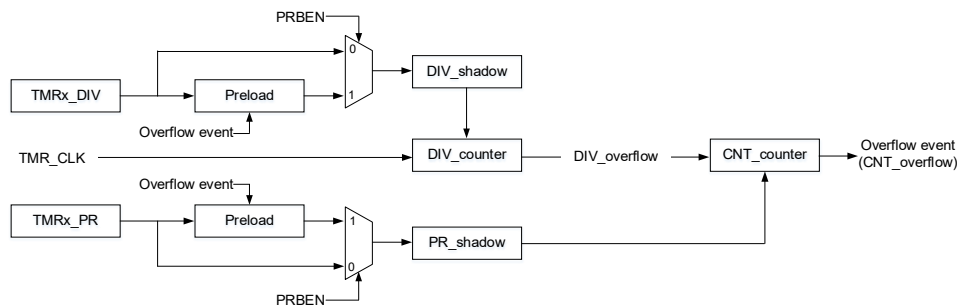
The TMRx\_PR register is used to define counting period of counter. The value in the TMRx\_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN = 1), the value in the TMRx\_PR register is transferred to the shadow register only at an overflow event.

TMRx\_DIV register is used to define the counter frequency of the counter. The counter counts once every DIV[15:0] + 1 clock cycle. Similar to TMRx\_PR register, after enabling periodic buffer, the value of the TMRx\_DIV register are transferred into the shadow register at each overflow event.

Reading the TMRx\_CNT register returns the current counter value. Writing the TMRx\_CNT register will update the current counter value.

An overflow event is enabled by default. It can be disabled by setting OVFEN = 1 in the TMRx\_CTRL1 register. The OVFS bit in the TMRx\_CTRL1 register is used to select the source of an overflow event, which is, by default, counter overflow or underflow, setting OVFSWTR, reset signal generated by slave mode timer controller in reset mode. Once the OVFS is set, an overflow event is generated only when overflow or underflow occurs.

Setting the TMREN bit (TMREN = 1) enables the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR\_EN is set 1 clock cycle after the TMREN is set.

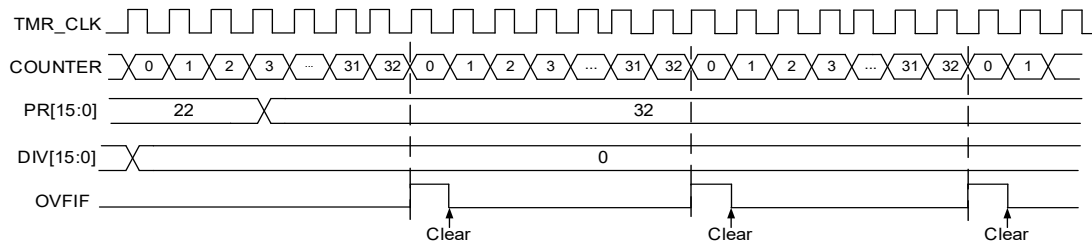


**Figure 127. Basic structure of a counter**

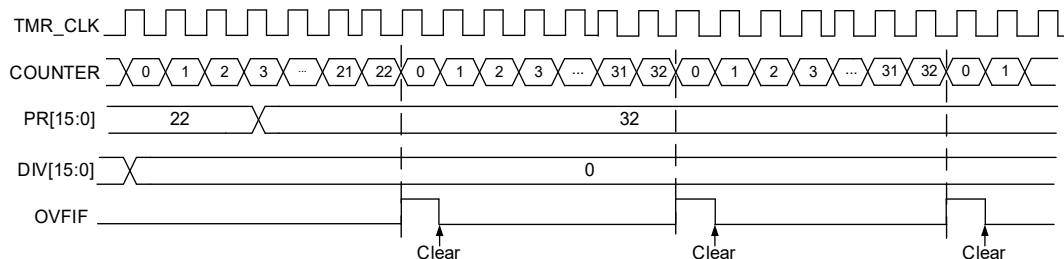
### Upcounting mode

This mode is enabled by setting  $TWCMSEL[1:0] = 2'b00$  and  $OWCDIR = 1'b0$  in the  $TMRx\_CTRL1$  register.

In upcounting mode, the counter counts from 0 to the value programmed in the  $TMRx\_PR$  register, restarts from 0, and generates a counter overflow event, with setting  $OVFIF$  bit to 1. If the overflow event is disabled, the counter is no longer reloaded with the prescaler and re-loaded value on counter overflow; otherwise, the prescaler and re-loaded value will be updated on an overflow event.



**Figure 128. Overflow event when  $PRBEN = 0$**



**Figure 129. Overflow event when  $PRBEN = 1$**

## TMR input function

Each timer of TMR13 and TMR14 has an independent channel that can be configured as input or output.

As input, each channel input signal is processed as follows:

- TMRx\_CHx outputs the pre-processed CxIRAW. The C1INSE bit is used to select TMRx\_CHx as the source of C1IRAW.
- CxIRAW inputs digital filter and outputs filtered CxIF signal. The digital filter uses the CxDF bit to program sampling frequency and sampling times.
- CxIF inputs edge detector, and outputs the CxIFPx signal after edge selection. The edge selection depends on both CxP and CxCP bits. It is possible to select input rising edge, falling edge or both edges.
- CxIFPx inputs capture signal selector, and outputs the CxIN signal after capture signal selection. The capture signal selection is defined by CxC bits. It is possible to select CxIFPx as CxIN source.
- CxIN outputs the CxIPS signal that is divided by input channel divider. The divider factor can be defined as No division, /2, /4 or /8, by the CxIDIV bit.

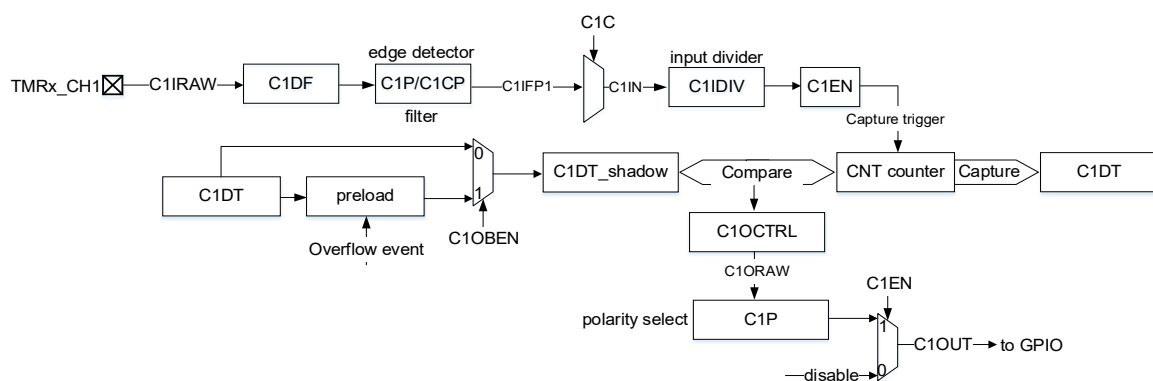


Figure 130. Input/output channel 1 main circuit

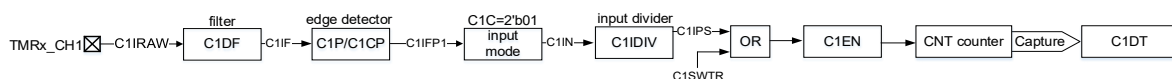


Figure 131. Channel 1 input stage

### Input mode

In input mode, the TMRx\_CxDT register latches the current counter values after the selected trigger signal is detected, and the capture compare interrupt flag bit (CxIF) is set to 1. An interrupt or DMA request will be generated if the CxIEN bit or CxDEN bit is enabled. If the selected trigger signal is detected when CxIF = 1, a capture overflow event is generated. The previous counter value will be overwritten by the current counter value, and the CxRF is set to 1.

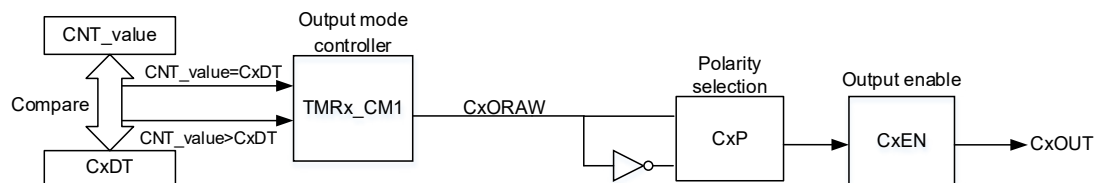
To capture the rising edge of C1IN input, follow the procedure below:

- Set C1C = 01 in the TMRx\_CM1 register to select the C1IN as channel 1 input.

- Set C1IN signal filter bandwidth (CxDF[3:0]).
- Set the active edge on the C1IN channel by writing C1P = 0 (rising edge) in the TMRx\_CCTR register.
- Program C1IN signal capture frequency divider (C1DIV[1:0]).
- Enable channel 1 input capture (C1EN = 1).
- If needed, enable the relevant interrupt by setting the C1IEN bit in the TMRx\_IDEN register.

### TMR output function

The TMR output consists of a comparator and an output controller. It is used to program the period, duty cycle and polarity of the output signal.



**Figure 132. Capture/compare channel output stage**

### Output mode

Write CxC[1:0] ≠ 2'b00 to configure the channel as output to implement multiple output modes. In this case, the counter value is compared with the value in the TMRx\_CxDT register, and the intermediate signal CxORAW is generated according to the output mode selected by CxOCTRL[2:0], which is sent to IO after being processed by the output control circuit. The period of the output signal is configured by the TMRx\_PR register, while the duty cycle by the TMRx\_CxDT register.

Output compare modes include:

#### PWM mode A:

Enable PWM mode A by setting CxOCTRL = 3'b110. In upcounting mode, C1ORAW outputs high when TMRx\_C1DT > TMRx\_CVAL; otherwise, it is low. In downcounting mode, C1ORAW outputs low when TMRx\_C1DT < TMRx\_CVAL; otherwise, it is high.

To use PWM mode A, the following procedures are recommended:

- Set PWM period through TMRx\_PR register.
- Set PWM duty cycles through TMRx\_CxD.
- Select PWM mode A by setting CxOCTRL = 3'b110 in the TMRx\_CM1/CM2 register.
- Set counting frequency through TMRx\_DIV register.
- Select counting mode by setting the TWCMSEL[1:0] bit in the TMRx\_CTRL1 register.
- Select output polarity through the CxP and CxCP bits in the TMRx\_CCTRL register.
- Enable channel output through the CxEN and CxCEN bits in the TMRx\_CCTRL register.
- Enable TMRx output through the OEN bit in the TMRx\_BRK register.
- Configure GPIOs corresponding to TMR output channels as multiplexed mode.
- Enable TMRx to start counting through the TMREN bit in the TMRx\_CTRL1 register.

### PWM mode B:

Enable PWM mode B by setting  $CxOCTRL = 3'b111$ . In upcounting mode, C1ORAW outputs low when  $TMRx\_C1DT > TMRx\_CVAL$ ; otherwise, it is high. In downcounting mode, C1ORAW outputs high when  $TMRx\_C1DT < TMRx\_CVAL$ ; otherwise, it is low.

### Forced output mode:

Enable forced output mode by setting  $CxOCTRL = 3'b100/101$ . In this case, the CxORAW is forced to be the programmed level, regardless of the counter value. Despite this, the channel flag bit and DMA request still depend on the compare result.

### Output compare mode:

Enable output compare mode by setting  $CxOCTRL = 3'b001/010/011$ . In this case, when the counter value matches the value of the CxDT register, the CxORAW is forced high ( $CxOCTRL = 3'b001$ ), low ( $CxOCTRL = 3'b010$ ) or toggling ( $CxOCTRL = 3'b011$ ).

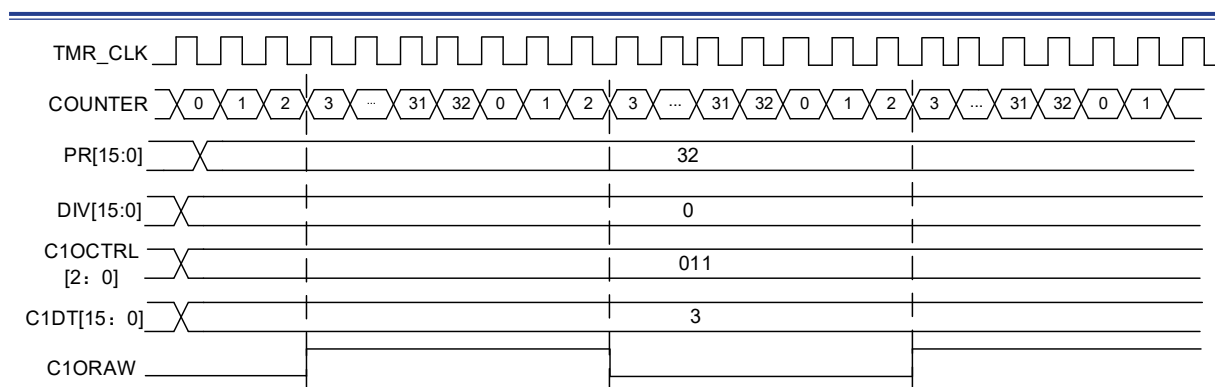
### One-pulse mode:

This is a particular case of PWM mode. Enable one-pulse by setting  $OCMEN = 1$ . In this mode, the comparison match is performed in the current counting period. The  $TMREN$  bit is cleared as soon as the current counting is completed. Therefore, only one pulse is output. When in upcounting mode, the configuration must follow the rule:  $CVAL < CxDT \leq PR$ ; in downcounting mode,  $CVAL > CxDT$  is required.

Figure 133 gives an example of output compare mode (toggle) with  $C1DT = 0x3$ . When the counter value is equal to  $0x3$ , C1OUT toggles.

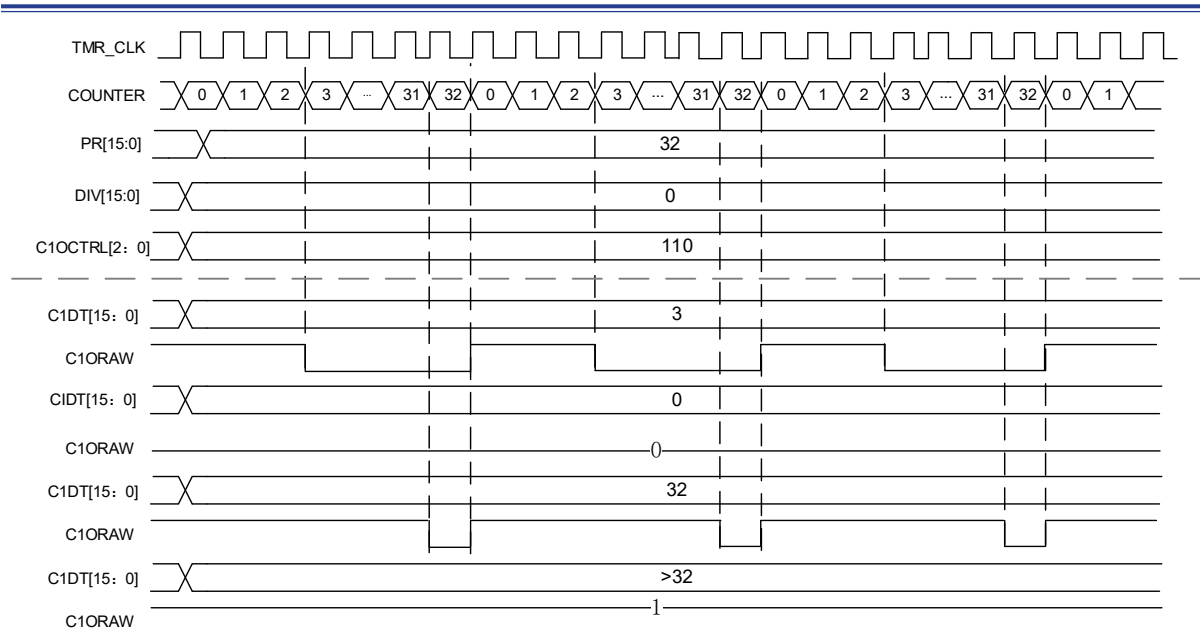
Figure 134 gives an example of the combination between upcounting mode and PWM mode A. The output signal behaves when  $PR = 0x32$  but  $CxDT$  is configured with a different value.

Figure 135 gives an example of the combination between upcounting mode and one-pulse PWM mode B. The counter only counts only one cycle, and the output signal sends only one pulse.

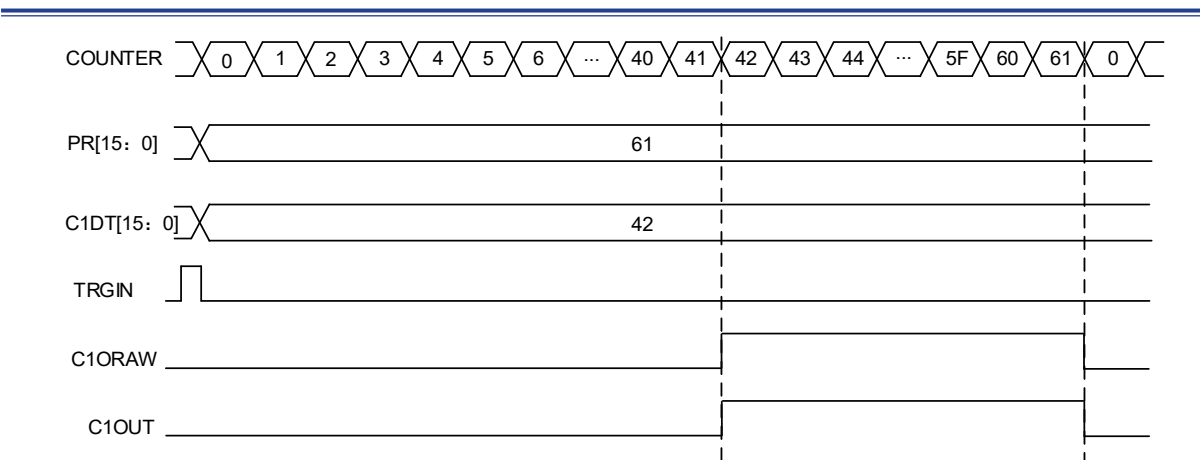


**Figure 133. C1ORAW toggles when counter value matches the C1DT value**





**Figure 134. Upcounting mode and PWM mode A**



**Figure 135. One-pulse mode**

### Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the TMRx counter stops counting by setting the TMRx\_PAUSE in the DEBUG module.

## TMR13 and TMR14 registers

These peripheral registers must be accessed by word (32 bits).

TMR13 and TMR14 registers are mapped into a 16-bit addressable space.

**Table 52. TMR13 and TMR14 register map and reset value**

| Register   | Offset | Reset value |
|------------|--------|-------------|
| TMRx_CTRL1 | 0x00   | 0x0000      |
| TMRx_IDEN  | 0x0C   | 0x0000      |
| TMRx_ISTS  | 0x10   | 0x0000      |
| TMRx_SWEVT | 0x14   | 0x0000      |
| TMRx_CM1   | 0x18   | 0x0000      |
| TMRx_CCTRL | 0x20   | 0x0000      |
| TMRx_CVAL  | 0x24   | 0x0000      |
| TMRx_DIV   | 0x28   | 0x0000      |
| TMRx_PR    | 0x2C   | 0x0000      |
| TMRx_C1DT  | 0x34   | 0x0000      |
| TMR14_RMP  | 0x50   | 0x0000      |

### TMR13 and TMR14 control register 1 (TMRx\_CTRL1)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15:10 | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 9:8   | CLKDIV   | 0x0         | rw   | Clock divider<br>This field is used to define the relationship between digital filter sampling frequency ( $f_{DTS}$ ) and timer clock frequency ( $f_{CK\_INT}$ ).<br>00: No division, $f_{DTS} = f_{CK\_INT}$<br>01: Divided by 2, $f_{DTS} = f_{CK\_INT}/2$<br>10: Divided by 4, $f_{DTS} = f_{CK\_INT}/4$<br>11: Reserved |
| Bit 7     | PRBEN    | 0x0         | rw   | Period buffer enable<br>0: Period buffer is disabled<br>1: Period buffer is enabled   |
| Bit 6:4   | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 3     | OCMEN    | 0x0         | rw   | One cycle mode enable<br>This bit is use to select whether to stop counting at an overflow event<br>0: The counter does not stop at an overflow event<br>1: The counter stops at an overflow event  |
| Bit 2     | OVFS     | 0x0         | rw   | Overflow event source<br>This bit is used to select overflow event or DMA request sources.<br>0: Counter overflow, setting the OVFSWTR bit or overflow event generated by slave timer controller<br>1: Only counter overflow generates an overflow event  |
| Bit 1     | OVFEN    | 0x0         | rw   | Overflow event enable<br>0: Enabled<br>1: Disabled  |
| Bit 0     | TMREN    | 0x0         | rw   | TMR enable<br>0: Enabled<br>1: Disabled   |

#### TMR13 and TMR14 DMA/interrupt enable register (TMRx\_IDEN)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:2 | Reserved | 0x0         | resd | Kept at its default value.                              |
| Bit 1    | C1IEN    | 0x0         | rw   | Channel 1 interrupt enable<br>0: Disabled<br>1: Enabled |
| Bit 0    | OVIEN    | 0x0         | rw   | Overflow interrupt enable<br>0: Disabled<br>1: Enabled  |

#### TMR13 and TMR14 interrupt status register (TMRx\_ISTS)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:10 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 9     | C1RF     | 0x0         | rw0c | Channel 1 recapture flag<br>This bit indicates whether a recapture is detected when C1IF = 1. This bit is set by hardware, and cleared by writing "0".<br>0: No capture is detected<br>1: Capture is detected.   |
| Bit 8:2   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 1     | C1IF     | 0x0         | rw0c | Channel 1 interrupt flag<br>If the channel 1 is configured as input mode:<br>This bit is set by hardware on a capture event. It is cleared by software or read access to the TMRx_C1DT<br>0: No capture event occurs<br>1: Capture event is generated<br>If the channel 1 is configured as output mode:<br>This bit is set by hardware on a compare event. It is cleared by software.<br>0: No compare event occurs<br>1: Compare event is generated |
| Bit 0     | OVIIF    | 0x0         | rw0c | Overflow interrupt flag<br>This bit is set by hardware on an overflow event. It is cleared by software.<br>0: No overflow event occurs<br>1: Overflow event is generated.  |

#### TMR13 and TMR14 software event register (TMRx\_SWEVT)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:2 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 1    | C1SWTR   | 0x0         | wo   | Channel 1 event triggered by software<br>This bit is set by software to generate a channel 1 event.<br>0: No effect<br>1: Generate a channel 1 event. |
| Bit 0    | OVSFSTR  | 0x0         | wo   | Overflow event triggered by software<br>This bit is set by software to generate an overflow event.<br>0: No effect<br>1: Generate an overflow event.  |

### TMR13 and TMR14 channel mode register 1 (TMRx\_CM1)

The channel can be used in input (capture mode) or output (compare mode). The direction of a channel is defined by the corresponding CxC bits. All the other bits of this register have different functions in input and output modes. The CxOx describes its function in output mode when the channel is in output mode, while the CxIx describes its function in output mode when the channel is in input mode. Attention must be given to the fact that the same bit can have different functions in input mode and output mode.

#### Output compare mode:

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:7 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 6:4  | C1OCTRL  | 0x0         | rw   | Channel 1 output control<br>This field defines the behavior of the original signal C1ORAW.<br>000: Disconnected. C1ORAW is disconnected from C1OUT;<br>001: C1ORAW is high when TMRx_CVAL = TMRx_C1DT<br>010: C1ORAW is low when TMRx_CVAL = TMRx_C1DT<br>011: Switch C1ORAW level when TMRx_CVAL = TMRx_C1DT<br>100: C1ORAW is forced low<br>101: C1ORAW is forced high.<br>110: PWM mode A<br>– OWCDIR = 0, C1ORAW is high once TMRx_C1DT > TMRx_CVAL, else low;<br>– OWCDIR = 1, C1ORAW is low once TMRx_C1DT < TMRx_CVAL, else high;<br>111: PWM mode B<br>– OWCDIR = 0, C1ORAW is low once TMRx_C1DT > TMRx_CVAL, else high;<br>– OWCDIR = 1, C1ORAW is high once TMRx_C1DT < TMRx_CVAL, else low.<br>Note: In the configurations other than 000', the C1OUT is connected to C1ORAW. The C1OUT output level is not only subject to the changes of C1ORAW, but also the output polarity set by CCTRL. |
| Bit 3    | C1OBEN   | 0x0         | rw   | Channel 1 output buffer enable<br>0: Buffer function of TMRx_C1DT is disabled. The new value written to the TMRx_C1DT takes effect immediately.<br>1: Buffer function of TMRx_C1DT is enabled. The value to be written to the TMRx_C1DT is stored in the buffer register, and can be sent to the TMRx_C1DT register only on an overflow event.  |
| Bit 2    | C1OIEN   | 0x0         | rw   | Channel 1 output enable immediately<br>In PWM mode A or B, this bit is used to accelerate the channel 1 output's response to the trigger event.<br>0: Need to compare the CVAL with C1DT before generating an output<br>1: No need to compare the CVAL and C1DT. An output is generated immediately when a trigger event occurs.  |
| Bit 1:0  | C1C      | 0x0         | rw   | Channel 1 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':<br>00: Output<br>01: Input, C1IN is mapped on C1IFP1<br>10: Reserved<br>11: Reserved  |

**Input capture mode:**

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:8 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 7:4  | C1DF     | 0x0         | rw   | <p>Channel 1 digital filter</p> <p>This field defines the digital filter of the channel 1. N stands for the number of filtering, indicating that the input edge can pass the filter only after N sampling events.</p> <p>0000: No filter, sampling is done at <math>f_{DTS}</math></p> <p>0001: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 2</math></p> <p>0010: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 4</math></p> <p>0011: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 8</math></p> <p>0100: <math>f_{SAMPLING} = f_{DTS}/2</math>, <math>N = 6</math></p> <p>0101: <math>f_{SAMPLING} = f_{DTS}/2</math>, <math>N = 8</math></p> <p>0110: <math>f_{SAMPLING} = f_{DTS}/4</math>, <math>N = 6</math></p> <p>0111: <math>f_{SAMPLING} = f_{DTS}/4</math>, <math>N = 8</math></p> <p>1000: <math>f_{SAMPLING} = f_{DTS}/8</math>, <math>N = 6</math></p> <p>1001: <math>f_{SAMPLING} = f_{DTS}/8</math>, <math>N = 8</math></p> <p>1010: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 5</math></p> <p>1011: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 6</math></p> <p>1100: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 8</math></p> <p>1101: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 5</math></p> <p>1110: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 6</math></p> <p>1111: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 8</math></p> |
| Bit 3:2  | C1DIV    | 0x0         | rw   | <p>Channel 1 input divider</p> <p>This field defines Channel 1 input divider.</p> <p>00: No divider. An input capture is generated at each active edge.</p> <p>01: An input compare is generated every 2 active edges</p> <p>10: An input compare is generated every 4 active edges</p> <p>11: An input compare is generated every 8 active edges</p> <p>Note: the divider is reset once C1EN = '0'</p>   |
| Bit 1:0  | C1C      | 0x0         | rw   | <p>Channel 1 configuration</p> <p>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':</p> <p>00: Output</p> <p>01: Input, C1IN is mapped on C1IFP1</p> <p>10: Reserved</p> <p>11: Reserved</p>   |

**TMR13 and TMR14 channel control register (TMRx\_CTRL)**

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:2 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 1    | C1P      | 0x0         | rw   | <p>Channel 1 polarity</p> <p>When the channel 1 is configured as output mode:</p> <p>0: C1OUT is active high</p> <p>1: C1OUT is active low</p> <p>When the channel 1 is configured as input mode:</p> <p>0: C1IN active edge is on its rising edge. When used as external trigger, C1IN is not inverted.</p> <p>1: C1IN active edge is on its falling edge. When used as external trigger, C1IN is inverted.</p> |

| Bit  | Name | Reset value | Type | Description   |
|------|------|-------------|------|---|
| Bit0 | C1EN | 0x0         | rw   | Channel 1 enable<br>0: Input or output is disabled<br>1: Input or output is enabled |

**Table 53. Standard CxOUT channel output control bit**

| CxEN bit | CxOUT output state          |
|----------|-----------------------------|
| 0        | Output disabled (CxOUT = 0) |
| 1        | CxOUT = CxORAW + polarity   |

Note: The state of the external I/O pins connected to the standard CxOUT channel depends on the CxOUT channel state and the GPIO and IOMUX registers.

#### TMR13 and TMR14 counter value (TMRx\_CVAL)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---------------|
| Bit 15:0 | CVAL | 0x0         | rw   | Counter value |

#### TMR13 and TMR14 division value (TMRx\_DIV)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 15:0 | DIV  | 0x0         | rw   | Divider value<br>The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK} / (DIV[15:0] + 1)$ .<br>DIV contains the value written at an overflow event. |

#### TMR13 and TMR14 period register (TMRx\_PR)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | PR   | 0x0         | rw   | Period value<br>This defines the period value of the TMRx counter. The timer stops working when the period value is 0. |

#### TMR13 and TMR14 channel 1 data register (TMRx\_C1DT)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | C1DT | 0x0         | rw   | Channel 1 data register<br>When the channel 1 is configured as input mode:<br>The C1DT is the CVAL value stored by the last channel 1 input event (C1IN)<br>When the channel 1 is configured as output mode:<br>C1DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C1OBEN bit, and the corresponding output is generated on C1OUT as configured. |

#### TMR14 channel input remap register (TMR14\_RMP)

| Bit      | Name           | Reset value | Type | Description   |
|----------|----------------|-------------|------|---|
| Bit 15:2 | Reserved       | 0x00        | resd | Kept at its default value   |
| Bit 1:0  | TMR14_CH1_IRMP | 0x0         | rw   | TMR14 channel 1 input remap<br>00: TMR14 channel 1 input is connected to GPIO<br>01: ERTC_CLK<br>10: HEXT/32<br>11: CLK_OUT |

## General-purpose timer (TMR15)

### TMR15 introduction

The general-purpose timer (TMR15) consists of a 16-bit upcounter, two capture/compare registers, and two independent channels to achieve embedded dead-time, input capture and programmable PWM output.

### TMR15 main features

- Source of count clock is selectable : internal clock, external clock and internal trigger
- 16-bit upcounter, and 8-bit repetition counter
- Two independent channels for input capture, output compare, PWM generation, one-pulse mode output and embedded dead-time
- One independent channel for complementary output
- TMR brake function
- Synchronization control between master and slave timers
- Interrupt/DMA is generated at overflow event, trigger event, brake signal input and channel event
- Support TMR burst DMA transfer

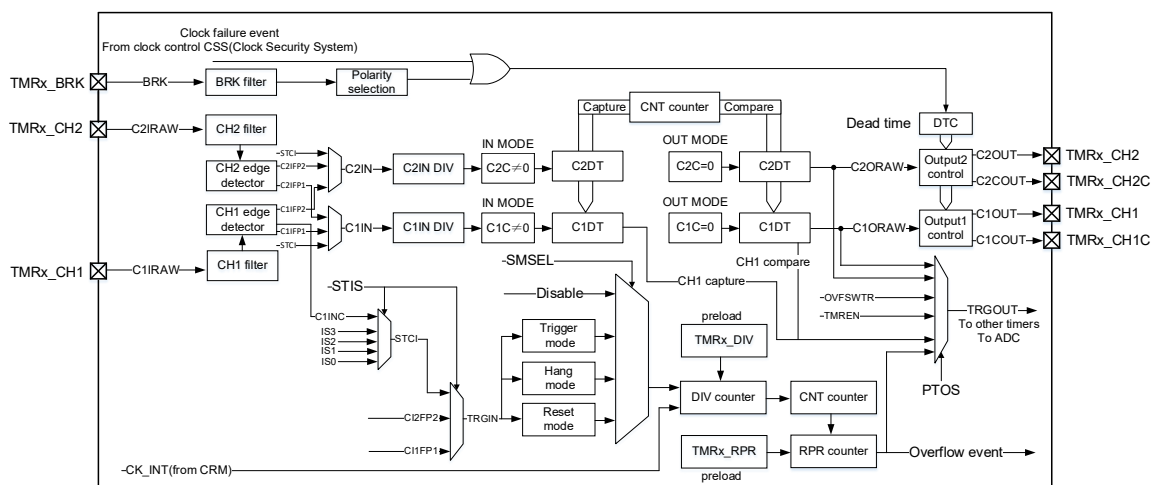
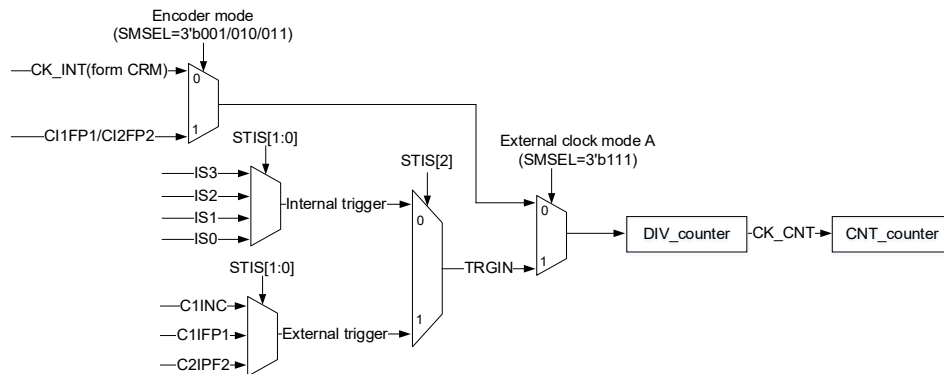


Figure 136. TMR15 block diagram

### TMR15 function overview

#### Counting clock

The count clock of TMR15 can be provided by the internal clock (CK\_INT), external clock (external clock mode A) and internal trigger input (ISx).



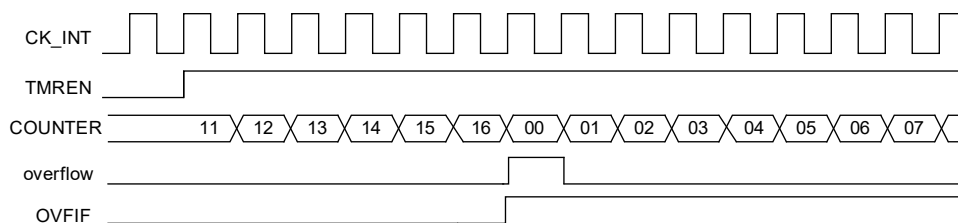
**Figure 137. Basic structure of a counter**

### Internal clock (CK\_INT)

By default, the CK\_INT, which is divided by the prescaler, is used to drive the counter to start counting. When TMR's APB clock prescaler factor is 1, the CK\_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

Follow the configuration steps below:

- Set counting frequency through TMRx\_DIV register
- Set counting cycles through TMRx\_PR register
- Enable a counter by setting the TMREN bit in the TMRx\_CTRL1 register



**Figure 138. Control circuit with CK\_INT divided by 1, TMRx\_DIV = 0x0 and PR = 0x16**

### External clock (TRGIN/EXT)

The counter clock can be provided by external clock source TRGIN.

SMSEL = 3'b111: External clock mode A is selected. By setting the STIS[2:0] bit, select an external clock source TRGIN signal to drive the counter to start counting.

The external clock sources include: C1INC (STIS = 3'b100, channel 1 rising edge and falling edge), C1IFP1 (STIS = 3'b101, a signal after channel 1 filter and polarity selection), C2IFP2 (STIS = 3'b110, a signal after channel 2 filter and polarity selection)

To use external clock mode A, follow the steps below:

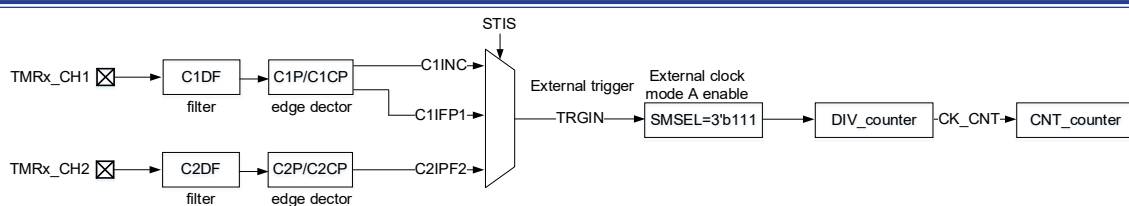
- Set external source TRGIN parameters

If the TMRx\_CH1 is used as a source of TRGIN, it is necessary to configure channel 1 input filter (C1DF[3:0] in TMRx\_CM1 register) and channel 1 input polarity (C1P/C1CP in TMRx\_CCTRL register);



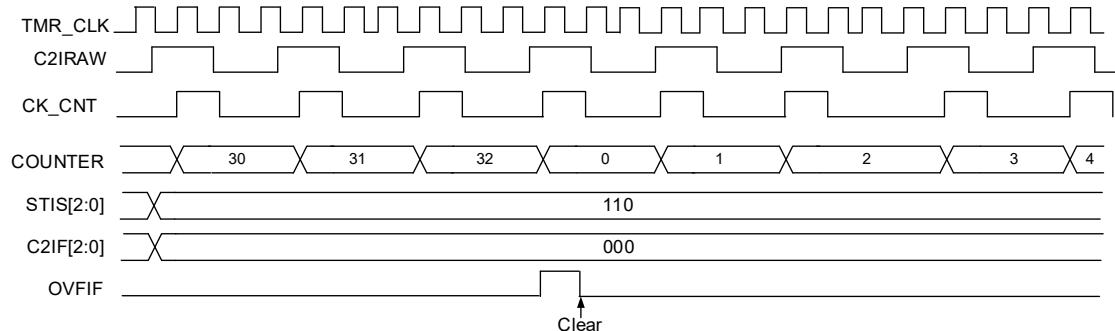
If the TMRx\_CH2 is used as source of TRGIN, it is necessary to configure channel 1 input filter (C2DF[3:0] in TMRx\_CM1 register) and channel 2 input polarity (C2P/C2CP in TMRx\_CCTR register).

- Set TRGIN signal source through the STIS[1:0] bit in TMRx\_STCTRL register.
- Enable external clock mode A by setting SMSEL = 3'b111 in TMRx\_STCTR register.
- Set counting frequency through the DIV[15:0] in TMRx\_DIV register.
- Set counting period through the PR[15:0] in TMRx\_PR register.
- Enable counter through the TMREN bit in TMRx\_CTRL1 register.



**Figure 139. Block diagram of external clock mode A**

Note: The delay between the signal on the input side and the actual clock of the counter is due to the synchronization circuit.



**Figure 140. Counting in external clock mode A, PR = 0x32, DIV = 0x0**

### Internal trigger input (ISx)

Timer synchronization allows interconnection between several timers. The TMR\_CLK of one timer can be provided by the TRGOUT signal output by another timer. Set the STIS[2:0] bit to select internal trigger signal to enable counting.

The TMR15 consists of a 16-bit prescaler, which is used to generate the CK\_CNT that enables the counter to count. The frequency division relationship between the CK\_CNT and TMR\_CLK can be adjusted by setting the value of the TMR15\_DIV register. The prescaler value can be modified at any time, but it takes effect only when the next overflow event occurs.

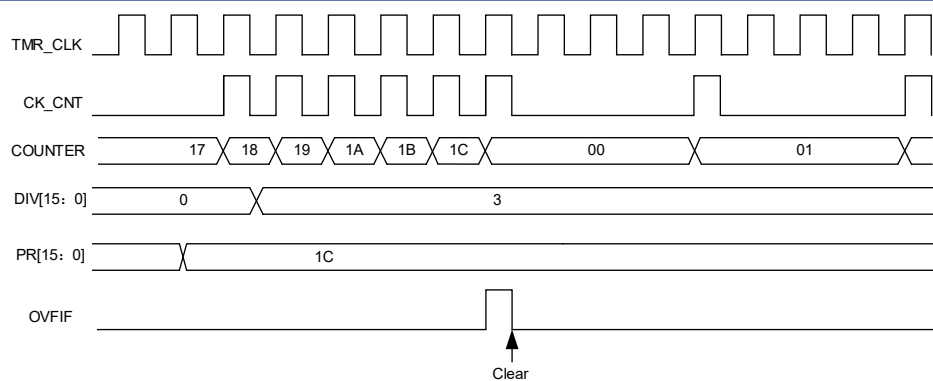
Below is the configuration procedure for internal trigger input:

- Set counting cycles through TMRx\_PR register.
- Set counting frequency through TMRx\_DIV register.

- Select internal trigger by setting STIS[2:0] = 3'b000~3'b011 in TMRx\_STCTRL register.
- Select external clock mode A by setting SMSEL[2:0] = 3'b111 in TMRx\_STCTRL register.
- Enable TMRx to start counting through the TMREN in TMRx\_CTRL1 register.

**Table 54. TMRx internal trigger connection**

| Slave controller | IS0 (STIS = 000) | IS1 (STIS = 001) | IS2 (STIS = 010) | IS3 (STIS = 011) |
|------------------|------------------|------------------|------------------|------------------|
| TMR1             | TMR15            | TMR2             | TMR3             | —                |
| TMR2             | TMR1             | TMR15            | TMR3             | USB_OTG_SOF      |
| TMR3             | TMR1             | TMR2             | TMR15            | —                |
| TMR15            | TMR2             | TMR3             | TMR16            | TMR17_OC         |



**Figure 141. Counter timing with prescaler value changing from 1 to 4**

### Counting mode

The TMR15 supports multiple counting modes to meet various application scenarios. It consists of a 16-bit upcounter.

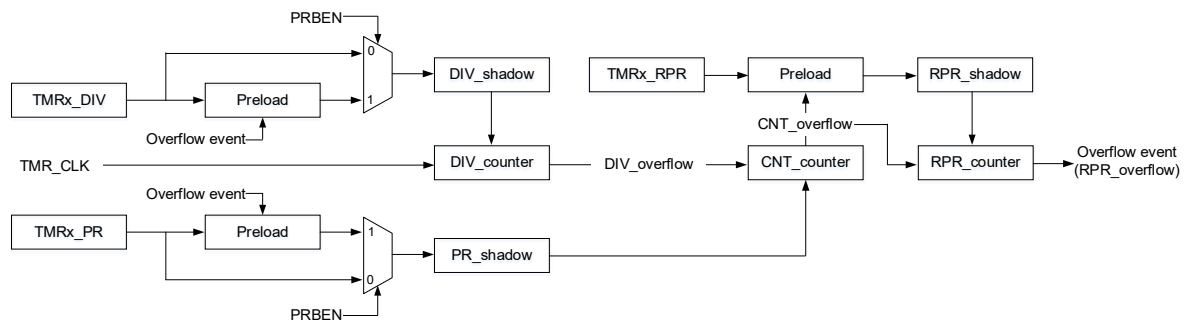
The TMRx\_PR register is used to define counting period of counter. The value in the TMRx\_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN = 1), the value in the TMRx\_PR register is transferred to the shadow register only at an overflow event.

TMRx\_DIV register is used to define the counter frequency of the counter. The counter counts once every DIV[15:0] + 1 clock cycle. Similar to TMRx\_PR register, after enabling periodic buffer, the value of the TMRx\_DIV register are transferred into the shadow register at each overflow event.

Reading the TMRx\_CNT register returns the current counter value. Writing the TMRx\_CNT register will update the current counter value.

An overflow event is enabled by default. It can be disabled by setting OVFEN = 1 in the TMRx\_CTRL1 register. The OVFS bit in the TMRx\_CTRL1 register is used to select the source of an overflow event, which is, by default, counter overflow or underflow, setting OVFSWTR, reset signal generated by slave mode timer controller in reset mode. Once the OVFS is set, an overflow event is generated only when overflow or underflow occurs.

Setting the TMREN bit (TMREN = 1) enables the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR\_EN is set 1 clock cycle after the TMREN is set.

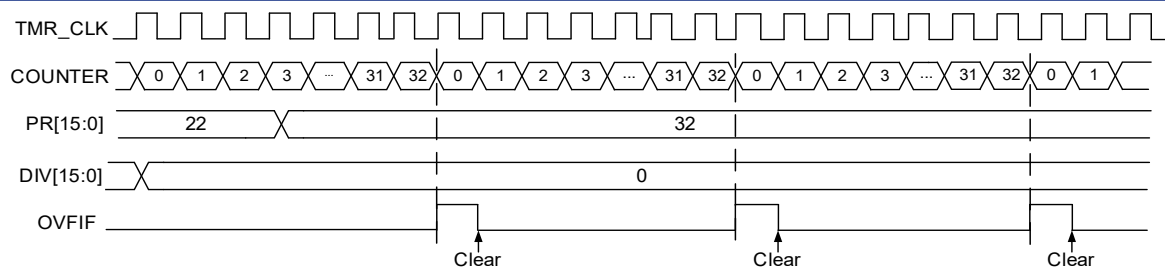


**Figure 142. Basic structure of a counter**

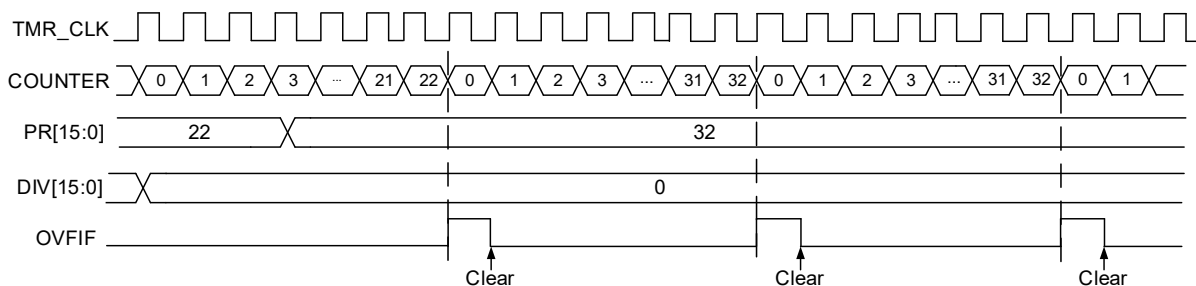
## Upcounting mode

This mode is enabled by setting `TWCMSEL[1:0] = 2'b00` and `OWCDIR = 1'b0` in the `TMRx_CTRL1` register.

In upcounting mode, the counter counts from 0 to the value programmed in the TMR15\_PR register, restarts from 0, and generates a counter overflow event, with setting OVIF bit to 1. If the overflow event is disabled, the counter is no longer reloaded with the prescaler and re-loaded value on counter overflow; otherwise, the prescaler and re-loaded value will be updated on an overflow event.



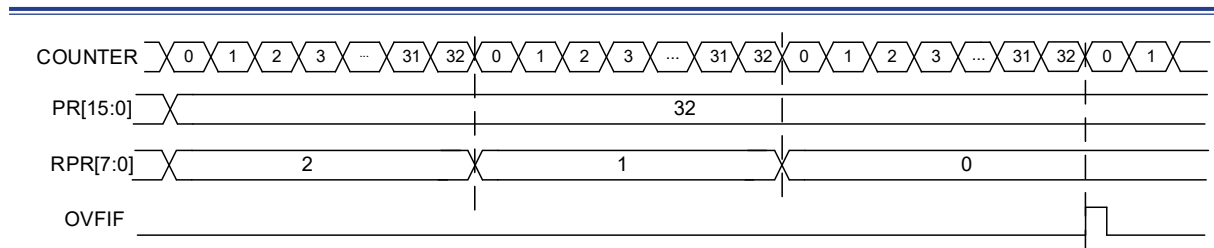
**Figure 143. Overflow event when PRBEN = 0**



**Figure 144. Overflow event when PRBEN = 1**

### Repetition counter mode:

The TMRx\_RPR register is used to set repetition counting mode. This mode is enabled when the repetition counter value is not equal to 0. In this mode, an overflow event is generated when a counter overflow occurs ( $RPR[7:0] + 1$ ). The repetition counter is decremented at each counter overflow. An overflow event is generated when the repetition counter reaches 0. The frequency of the overflow event can be adjusted by setting the repetition counter value.



**Figure 145. OVFIIF when RPR = 2**

### TMR input function

TMR15 has two independent channels. Each channel can be configured as input or output.

As input, each channel input signal is processed as follows:

- TMRx\_CHx outputs the pre-processed CxIRAW. The C1INSE bit is used to select TMRx\_CHx as the source of C1IRAW.
- CxIRAW inputs digital filter and outputs filtered CxIF signal. The digital filter uses the CxDF bit to program sampling frequency and sampling times.
- CxIF inputs edge detector, and outputs the CxIFPx signal after edge selection. The edge selection depends on both CxP and CxCP bits. It is possible to select input rising edge, falling edge or both edges.
- CxIFPx inputs capture signal selector, and outputs the CxIN signal after capture signal selection. The capture signal selection is defined by CxC bits. It is possible to select CxIFPx, CyIFPx or STCI as CxIN source. Of those, CyIFPx ( $x \neq y$ ) is the CyIFPy signal that is from Y channel and processed by channel-x edge detector (for example, C1IFP2 is the C1IFP1 signal that is from channel 1 and processed by channel 2 edge detector). The STCI comes from slave timer controller, and its source is selected by STIS bit.
- CxIN outputs the CxIPS signal that is divided by input channel divider. The divider factor can be defined as No division, /2, /4 or /8, by the CxIDIV bit.

```

graph LR
    STIS --> MUX_C1C
    STCI --> MUX_C1C
    C1INC --> MUX_C1C
    TMRx_CH1 --> C1IRAW
    C1IRAW --> C1DF
    C1DF -- C1IF --> C1P_C1CP
    C1P_C1CP -- C1IFP1 --> MUX_C1C
    TMRx_CH2 --> C2IRAW
    C2IRAW --> C2DF
    C2DF -- C2IF --> C2P_C2CP
    C2P_C2CP -- C2IFP1 --> MUX_C1C
    MUX_C1C -- C1IN --> C1DIV
    C1DIV -- C1IPS --> OR
    C1SWTR --> OR
    OR --> C1EN
    C1EN --> CNT_counter
    CNT_counter -- Capture --> C1DT

```

- Set C1C = 01 in the TMRx\_CM1 register to select the C1IN as channel 1 input.
- Set C1IN signal filter bandwidth (CxDF[3:0]).
- Set the active edge on the C1IN channel by writing C1P = 0 (rising edge) in the TMR15\_CCTRL register.
- Program C1IN signal capture frequency divider (C1DIV[1:0]).
- Enable channel 1 input capture (C1EN = 1).
- If needed, enable the relevant interrupt or DMA request by setting the C1IEN bit in the TMR15\_IDEN register or the C1DEN bit in the TMR15\_IDEN register.

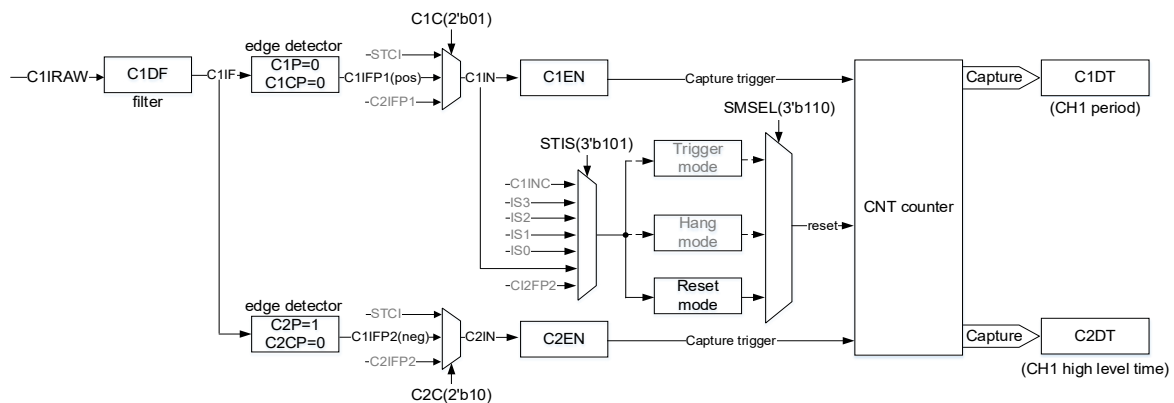
## PWM input

PWM input mode is applied to channel 1 and 2. To use this mode, both C1IN and C2IN are mapped on the same TMRx\_CHx, and the CxIFPx of either channel 1 or channel 2 must be configured as trigger input and slave mode controller is configured in reset mode.

The PWM input mode can be used to measure the period and duty cycle of the PWM input signal. For example, the user can measure the period and duty cycle of the PWM applied on channel 1 using the following procedures:

- Set C1C = 2'b01: select C1IN for C1IFP1.
- Set C1P = 1'b0, select C1IFP1 rising edge active.
- Set C2C = 2'b10, select C2IN for C1IFP2.
- Set C2P = 1'b1, select C1IFP2 falling edge active.
- Set STIS = 3'b101, select the slave mode timer trigger signal as C1IFP1.
- Set SMSEL = 3'b100: configure the slave mode controller in reset mode.
- Set C1EN = 1'b1 and C2EN = 1'b1. Enable channel 1 and input capture.

After above configuration, the rising edge of channel 1 input signal will trigger the capture and stores the capture value into C1DT register, and it will reset the counter at the same time. The falling edge of the channel 1 input signal triggers the capture and stores the capture value into C2DT register. The period of the channel 1 input signal is calculated through C1DT, and its duty cycle through C2DT.



**Figure 148. PWM input mode configuration example**

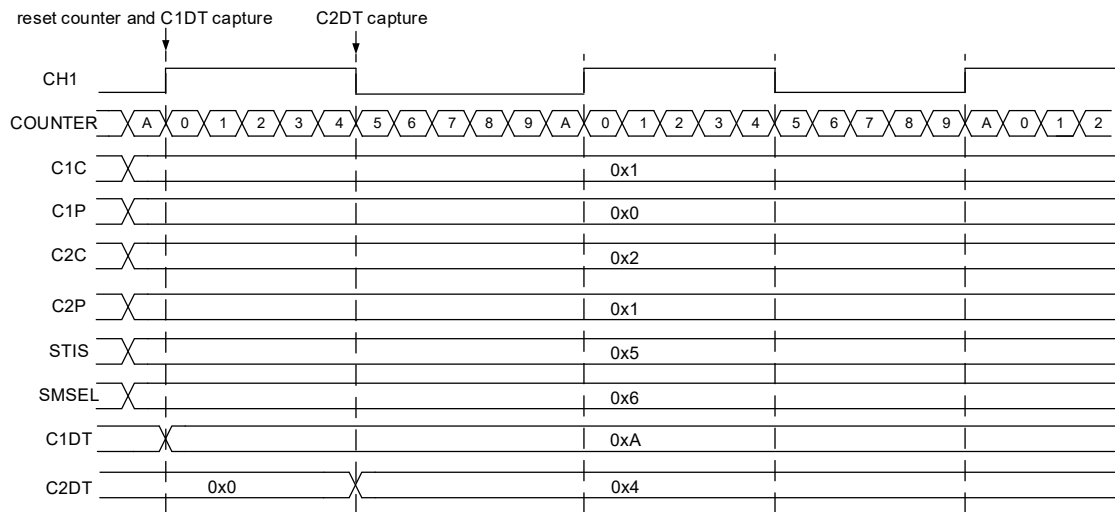


Figure 149. PWM input mode

### TMR output function

The TMR output consists of a comparator and an output controller. It is used to program the period, duty cycle and polarity of the output signal. The advanced-control timer output function varies from one channel to one channel.

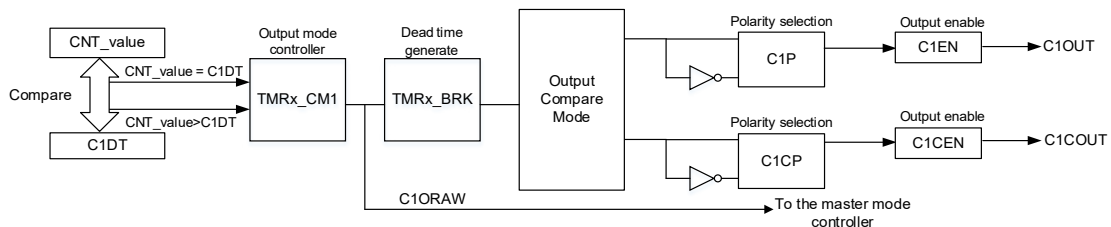


Figure 150. Channel 1 output stage

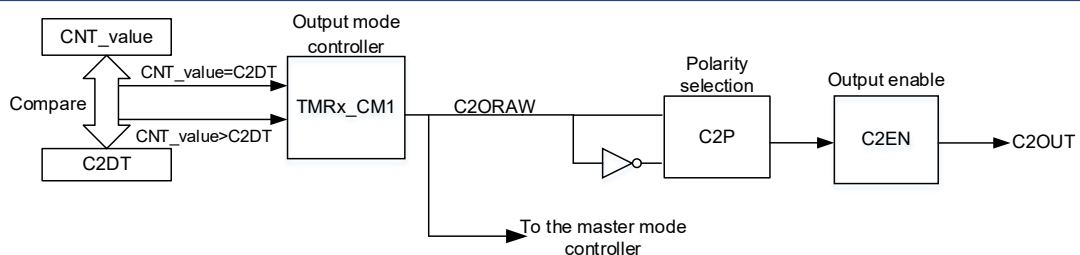


Figure 151. Channel 2 output stage

## Output mode

Write  $CxC[1:0] \neq 2'b00$  to configure the channel as output to implement multiple output modes. In this case, the counter value is compared with the value in the  $TMRx\_CxDT$  register, and the intermediate signal  $CxORAW$  is generated according to the output mode selected by  $CxOCTRL[2:0]$ , which is sent to IO after being processed by the output control circuit. The period of the output signal is configured by the  $TMR15\_PR$  register, while the duty cycle by the  $TMRx\_CxDT$  register.

### PWM mode A:

Enable PWM mode A by setting  $CxOCTRL = 3'b110$ . In upcounting mode,  $C1ORAW$  outputs high when  $TMRx\_C1DT > TMRx\_CVAL$ ; otherwise, it is low. In downcounting mode,  $C1ORAW$  outputs low when  $TMRx\_C1DT < TMRx\_CVAL$ ; otherwise, it is high.

To use PWM mode A, the following procedures are recommended:

- Set PWM periods through  $TMRx\_PR$  register.
- Set PWM duty cycles through  $TMRx\_CxDT$ .
- Select PWM mode A by setting  $CxOCTRL = 3'b110$  in the  $TMRx\_CM1/CM2$  register.
- Set counting frequency through  $TMRx\_DIV$  register.
- Select counting mode by setting the  $TWCMSEL[1:0]$  bit in the  $TMRx\_CTRL1$  register.
- Select output polarity through the  $CxP$  and  $CxCP$  bits in the  $TMRx\_CCTRL$  register.
- Enable channel output through the  $CxEN$  and  $CxCEN$  bits in the  $TMRx\_CCTRL$  register.
- Enable  $TMRx$  output through the  $OEN$  bit in the  $TMRx\_BRK$  register.
- Configure GPIOs corresponding to TMR output channels as multiplexed mode.
- Enable  $TMRx$  to start counting through the  $TMREN$  bit in the  $TMRx\_CTRL1$  register.

### PWM mode B:

Enable PWM mode B by setting  $CxOCTRL = 3'b111$ . In upcounting mode,  $C1ORAW$  outputs low when  $TMRx\_C1DT > TMRx\_CVAL$ ; otherwise, it is high. In downcounting mode,  $C1ORAW$  outputs high when  $TMRx\_C1DT < TMRx\_CVAL$ ; otherwise, it is low.

### Forced output mode:

Enable forced output mode by setting  $CxOCTRL = 3'b100/101$ . In this case, the  $CxORAW$  is forced to be the programmed level, regardless of the counter value. Despite this, the channel flag bit and DMA request still depend on the compare result.

### Output compare mode:

Enable output compare mode by setting  $CxOCTRL = 3'b001/010/011$ . In this case, when the counter value matches the value of the  $CxDT$  register, the  $CxORAW$  is forced high ( $CxOCTRL = 3'b001$ ), low ( $CxOCTRL = 3'b010$ ) or toggling ( $CxOCTRL = 3'b011$ ).

### One-pulse mode:

This is a particular case of PWM mode. Enable one-pulse by setting  $OCMEN = 1$ . In this mode, the comparison match is performed in the current counting period. The  $TMREN$  bit is cleared as soon as the current counting is completed. Therefore, only one pulse is output. When in upcounting mode, the configuration must follow the rule:  $CVAL < CxDT \leq PR$ ; in downcounting mode,  $CVAL > CxDT$  is required.



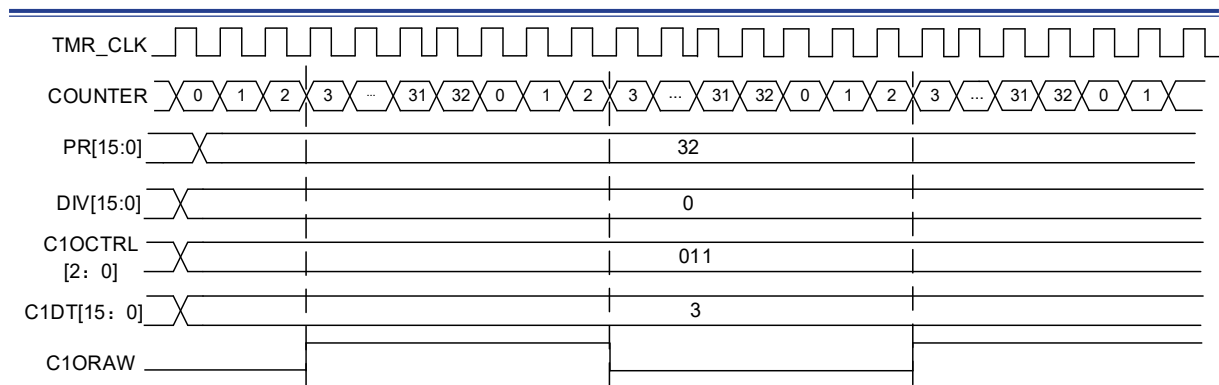
### Fast output mode:

Enable this mode by setting CxOIEN = 1. If enabled, the CxORAW signal will not change when the counter value matches the CxDT, but change at the beginning of the current counting period. In other words, the comparison result is advanced, so the comparison result between the counter value and the TMRx\_CxDT register will determine the level of CxORAW in advance.

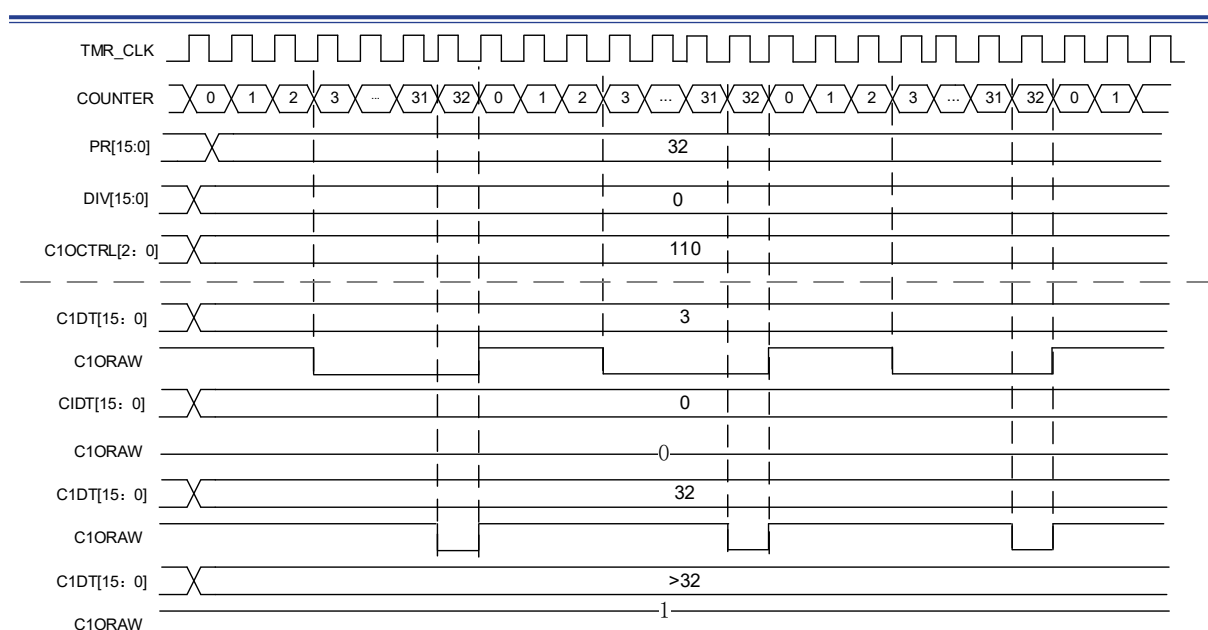
Figure 152 gives an example of output compare mode (toggle) with C1DT = 0x3. When the counter value is equal to 0x3, C1OUT toggles.

Figure 153 gives an example of the combination between upcounting mode and PWM mode A. The output signal behaves when PR = 0x32 but CxDT is configured with a different value.

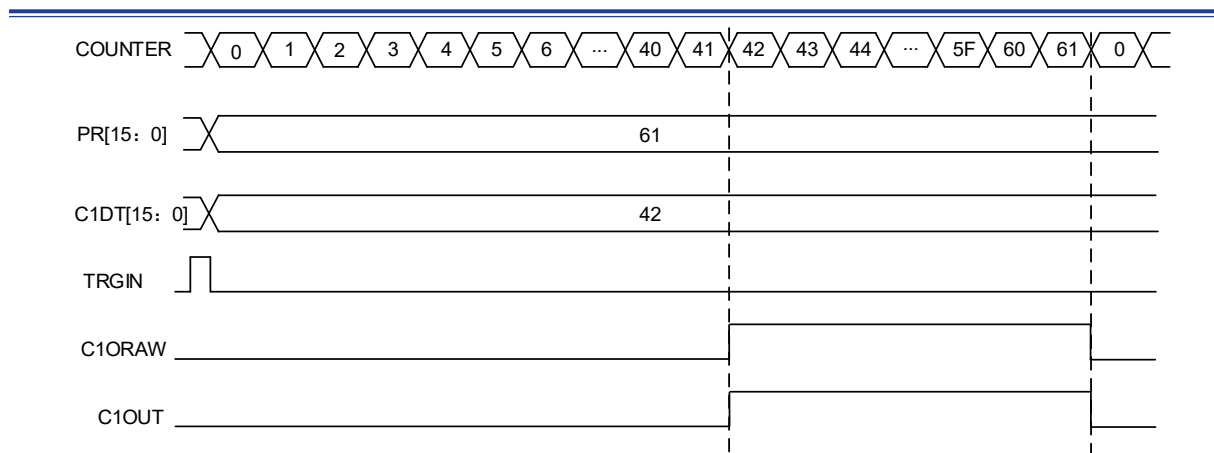
Figure 154 gives an example of the combination between upcounting mode and one-pulse PWM mode B. The counter only counts only one cycle, and the output signal sends only one pulse.



**Figure 152. C1ORAW toggles when counter value matches the C1DT value**



**Figure 153. Upcounting mode and PWM mode A**



**Figure 154. One-pulse mode**

### Master mode timer event output

When TMR is used as a master timer, one of the following source of signals can be selected as TRGOUT output to a slave mode timer. This is done by setting the PTOS bit in the TMRxCTRL2 register.

- PTOS = 3'b000, TRGOUT output software overflow event (OVFSWTR bit in TMRx\_SWEVT register) or reset event
- PTOS = 3'b001, TRGOUT output counter enable
- PTOS = 3'b010, TRGOUT output counter overflow event
- PTOS = 3'b011, TRGOUT output capture and compare event
- PTOS = 3'b100, TRGOUT output C1ORAW
- PTOS = 3'b101, TRGOUT output C2ORAW

### Dead-time insertion

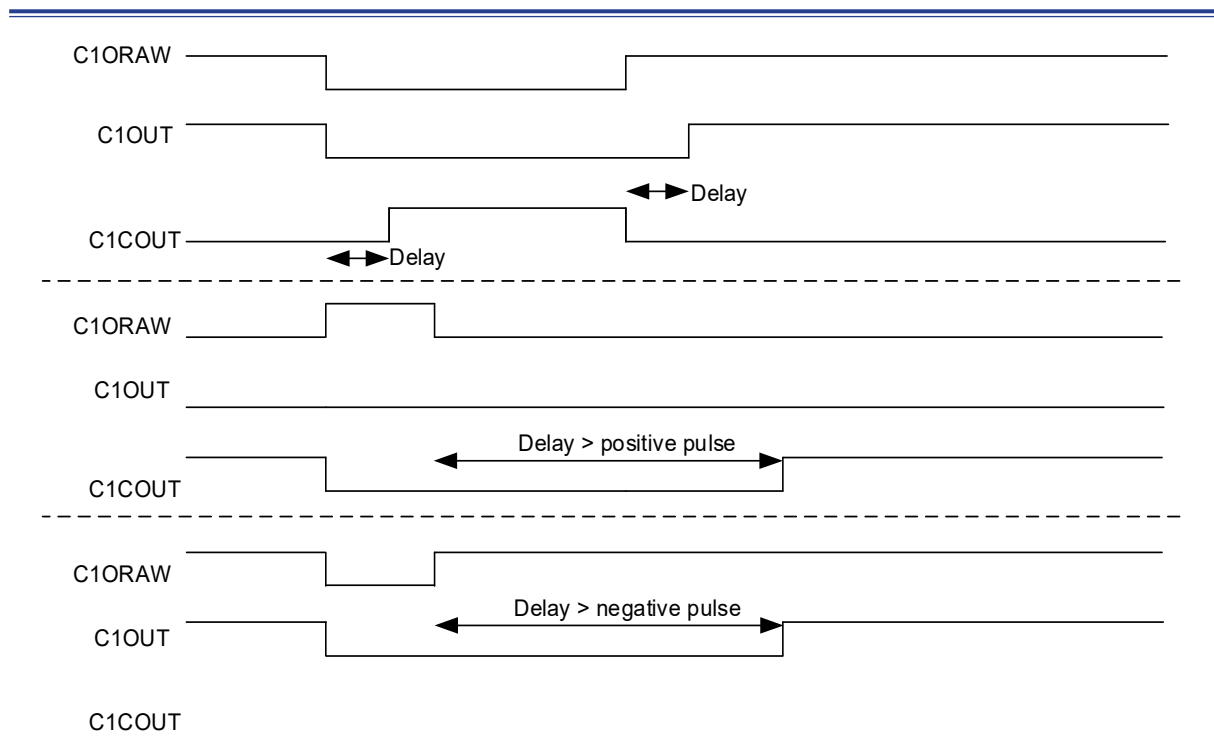
The TMR15 contains a set of reverse channel output. This function is enabled by the CxCEN bit and its polarity is selected by CxCP. Refer to Table 62 for more information about the output state of CxOUT and CxCOUT.

The dead-time is activated when switching to IDLEF state (OEN falling down to 0).

Setting both CxEN and CxCEN bits, and using DTC[7:0] bit to insert dead-time of different durations. After the dead-time insertion, the rising edge of the CxOUT is delayed compared to the rising edge of the reference signal; the rising edge of the CxCOU is delayed compared to the falling edge of the reference signal.

If the delay is greater than the width of the active output, C1OUT and C1COUT will not generate corresponding pulses. Therefore, the dead-time should be less than the width of the active output.

Figure 155 gives an example of dead-time insertion when CxP = 0, CxCP = 0, OEN = 1, CxEN = 1 and CxCEN = 1.



**Figure 155. Complementary output with dead-time insertion**

### TMR brake function

When the brake function is enabled (BRKEN = 1), the CxOUT and CxCOUT are jointly controlled by OEN, FCSODIS, FCSOEN, CxIOS and CxCIOS. But, CxOUT and CxCOUT cannot be set both to active level at the same time. Please refer to Table 56 for more details.

The brake source can be the brake input pin or a clock failure event. The polarity is controlled by the BRKV bit.

When a brake event occurs, there are the following actions:

- The OEN bit is cleared asynchronously, and the channel output state is selected by setting the FCSODIS bit. This function works even if the MCU oscillator is off.
- Once OEN = 0, the channel output level is defined by the CxIOS bit. If FCSODIS = 0, the timer output is disabled; otherwise, the output enable remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state, that is, 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 active, then the dead-time generator is activated. The CxIOS and CxCIOS bits are used to program the level after dead-time. Even in this case, the CxIOS and CxCIOS cannot be driven to their active level at the same time. It should be noted that because of synchronization on OEN, the dead-time duration is usually longer than usual (around 2 clk\_tmr clock cycles)
  - If FCSODIS = 0, the timer releases the enable output; otherwise, it keeps the enable output; the enable output becomes high as soon as one of the CxEN and CxCEN bits becomes high.

- If the brake interrupt or DMA request is enabled, the brake statue flag is set, and a brake interrupt or DMA request can be generated.
- If AOEN = 1, the OEN bit is automatically set again at the next overflow event.

Note: When the brake input is active, the OEN cannot be set, nor the status flag, BRKIF can be cleared.

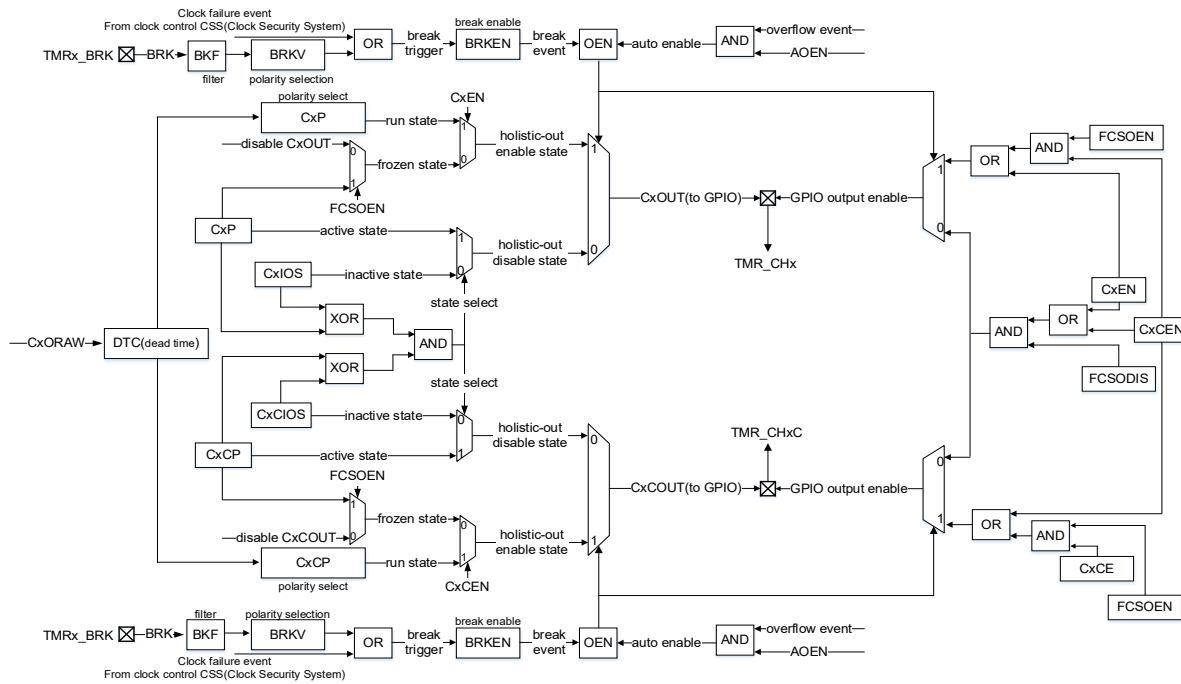


Figure 156. Example of TMR output control

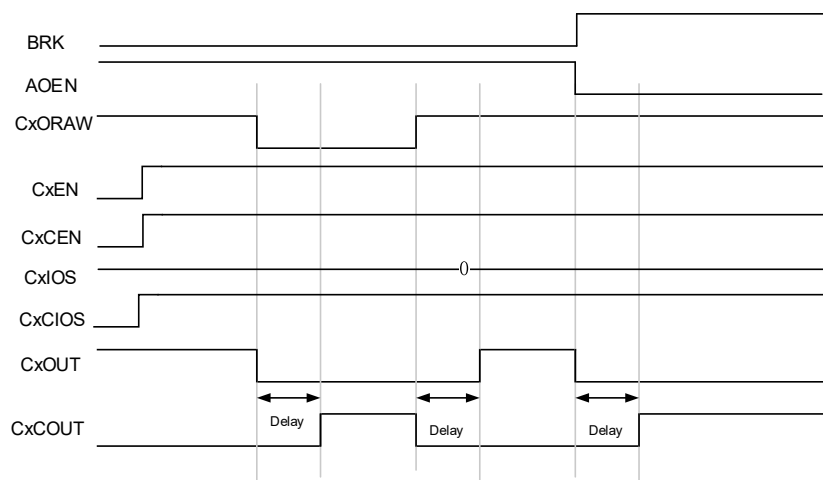


Figure 157. Example of TMR brake function

## TMR synchronization

The timers are linked together internally for timer synchronization. Master timer is selected by setting the PTOS[2:0] bit; Slave timer is selected by setting the SMSEL[2:0] bit.

Slave modes include:

### Slave mode: Reset mode

The counter and its prescaler can be reset by a selected trigger signal. An overflow event can be generated when OVFS = 0.

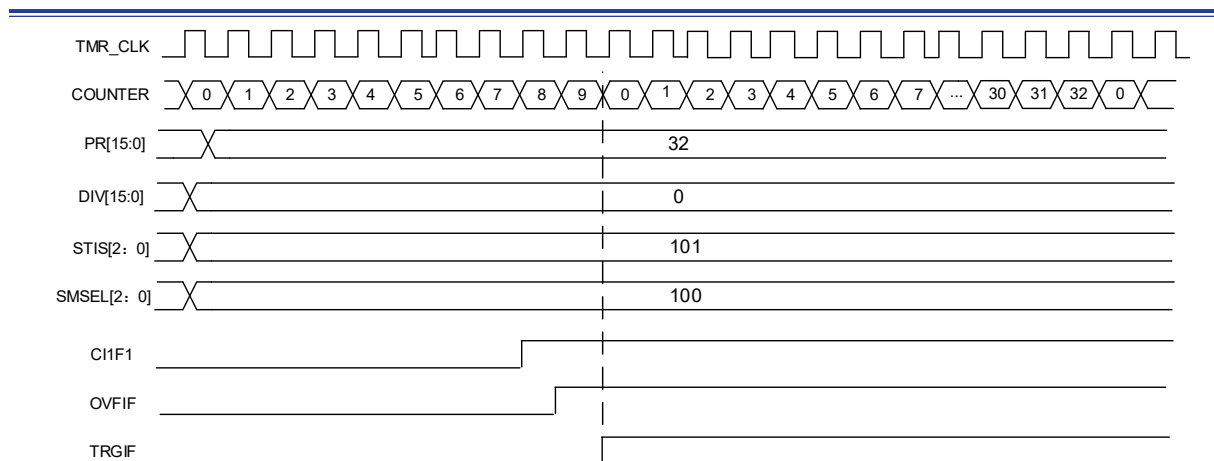


Figure 158. Example of reset mode

### Slave mode: Suspend mode

In this mode, the counter is controlled by a selected trigger input. The counter starts counting when the trigger input is high and stops as soon as the trigger input is low.

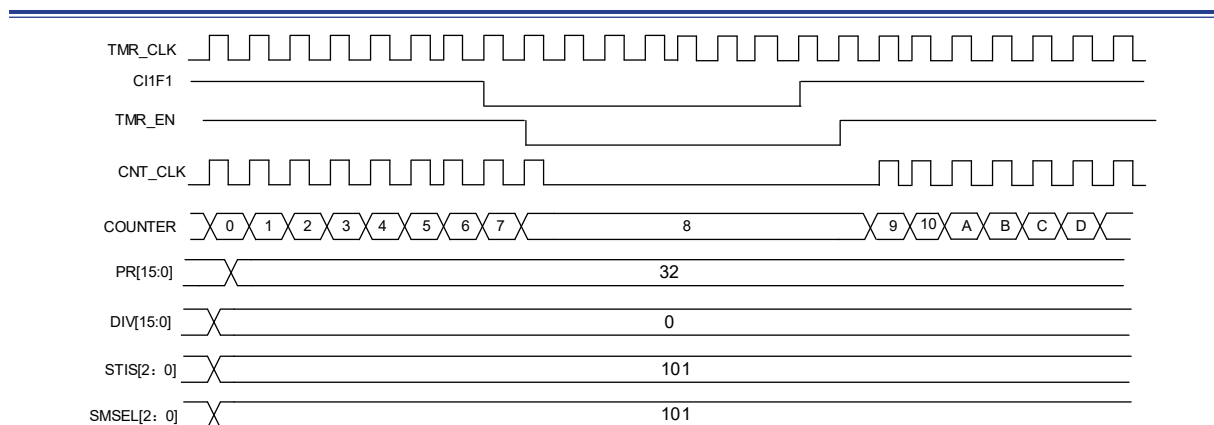
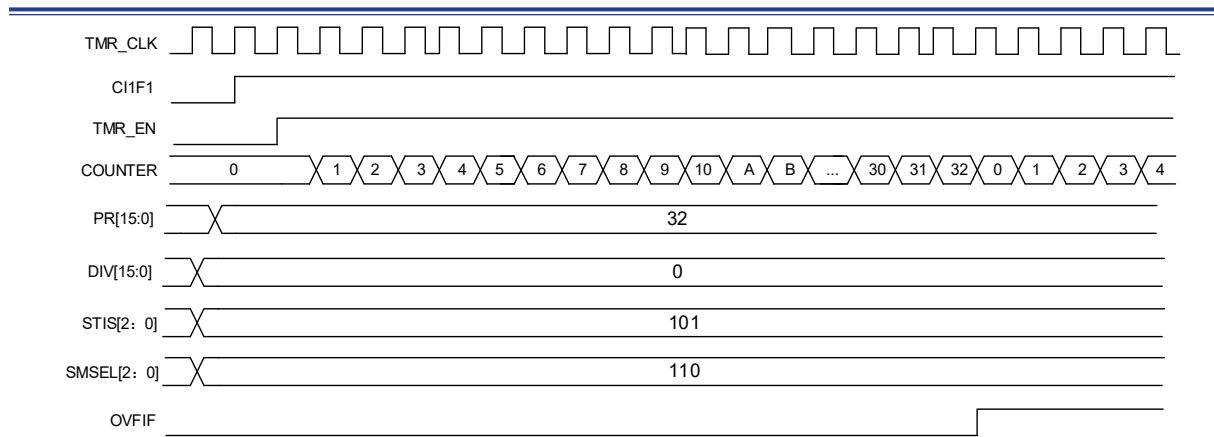


Figure 159. Example of suspend mode

### Slave mode: Trigger mode

The counter can start counting on the rising edge of a selected trigger input (TMR\_EN = 1).



**Figure 160. Example of trigger mode**

See chapter 14.2.3.5 for more information about timer synchronization.

### Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the TMRx counter stops counting by setting the TMRx\_PAUSE in the DEBUG module.

## TMR15 registers

These peripheral registers must be accessed by word (32 bits).

TMR15 registers are mapped into a 16-bit addressable space.

**Table 55. TMR15 register map and reset value**

| Register      | Offset | Reset value |
|---------------|--------|-------------|
| TMR15_CTRL1   | 0x00   | 0x0000      |
| TMR15_CTRL2   | 0x04   | 0x0000      |
| TMR15_STCTRL  | 0x08   | 0x0000      |
| TMR15_IDEN    | 0x0C   | 0x0000      |
| TMR15_ISTS    | 0x10   | 0x0000      |
| TMR15_SWEVT   | 0x14   | 0x0000      |
| TMR15_CM1     | 0x18   | 0x0000      |
| TMR15_CCTRL   | 0x20   | 0x0000      |
| TMR15_CVAL    | 0x24   | 0x0000      |
| TMR15_DIV     | 0x28   | 0x0000      |
| TMR15_PR      | 0x2C   | 0x0000      |
| TMR15_RPR     | 0x30   | 0x0000      |
| TMR15_C1DT    | 0x34   | 0x0000      |
| TMR15_C2DT    | 0x38   | 0x0000      |
| TMR15_BRK     | 0x44   | 0x0000      |
| TMR15_DMACTRL | 0x48   | 0x0000      |
| TMR15_DMADT   | 0x4C   | 0x0000      |

### TMR15 control register 1 (TMR15\_CTRL1)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15:10 | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 9:8   | CLKDIV   | 0x0         | rw   | Clock divider<br>This field is used to define the relationship between digital filter sampling frequency ( $f_{DTS}$ ) and timer clock frequency ( $f_{CK\_INT}$ ). It is also used to set the ratio relationship between dead time base ( $T_{DTS}$ ) and timer clock period ( $T_{CK\_INT}$ )<br>00: No division, $f_{DTS} = f_{CK\_INT}$<br>01: Divided by 2, $f_{DTS} = f_{CK\_INT}/2$<br>10: Divided by 4, $f_{DTS} = f_{CK\_INT}/4$<br>11: Reserved |
| Bit 7     | PRBEN    | 0x0         | rw   | Period buffer enable<br>0: Period buffer is disabled<br>1: Period buffer is enabled   |
| Bit 6:4   | Reserved | 0x0         | resd | Default value   |
| Bit 3     | OCMEN    | 0x0         | rw   | One cycle mode enable<br>This bit is use to select whether to stop counting at an update event<br>0: The counter does not stop at an update event<br>1: The counter stops at an update event  |
| Bit 2     | OVFS     | 0x0         | rw   | Overflow event source<br>This bit is used to select overflow event or DMA request sources.<br>0: Counter overflow, setting the OVFSWTR bit or overflow event generated by slave timer controller<br>1: Only counter overflow generates an overflow event  |
| Bit 1     | OVFEN    | 0x0         | rw   | Overflow event enable<br>0: Enabled<br>1: Disabled  |
| Bit 0     | TMREN    | 0x0         | rw   | TMR enable<br>0: Enabled<br>1: Disabled   |

### TMR15 control register 2 (TMR15\_CTRL2)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:11 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 10    | C2IOS    | 0x0         | rw   | Channel 2 idle output state   |
| Bit 9     | C1CIOS   | 0x0         | rw   | Channel 1 complementary idle output state<br>OEN = 0 after dead-time:<br>0: C1OUTL = 0<br>1: C1OUTL = 1 |
| Bit 8     | C1IOS    | 0x0         | rw   | Channel 1 idle output state<br>OEN = 0 after dead-time:<br>0: C1OUT = 0<br>1: C1OUT = 1                 |
| Bit 7     | Reserved | 0x0         | resd | Kept at its default value.  |

| Bit     | Name     | Reset value | Type | Description   |
|---------|----------|-------------|------|---|
| Bit 6:4 | PTOS     | 0x0         | rw   | Master TMR output selection<br>This field is used to select the TMRx signal sent to the slave timer.<br>000: Software overflow or Reset<br>001: Enable<br>010: Overflow<br>011: Compare pulse<br>100: C1ORAW signal<br>101: C2ORAW signal   |
| Bit 3   | DRS      | 0x0         | rw   | DMA request source<br>0: Capture/compare event<br>1: Overflow event   |
| Bit 2   | CCFS     | 0x0         | rw   | Channel control bit flash selection<br>This bit only acts on channels that have complementary output.<br>If the channel control bits are buffered:<br>0: Control bits are updated by setting the HALL bit<br>1: Control bits are updated by setting the HALL bit or a rising edge on TRGIN. |
| Bit 1   | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 0   | CBCTRL   | 0x0         | rw   | Channel buffer control<br>This bit acts on channels that have complementary output.<br>0: CxEN, CxCEN and CxOCTRL bits are not buffered.<br>1: CxEN, CxCEN and CxOCTRL bits are buffered.   |

#### TMR15 slave timer control register (TMR15\_STCTRL)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:8 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 7    | STS      | 0x0         | rw   | Subordinate TMR synchronization<br>If enabled, master and slave timer can be synchronized.<br>0: Disabled<br>1: Enabled   |
| Bit 6:4  | STIS     | 0x0         | rw   | Subordinate TMR input selection<br>This field is used to select the subordinate TMR input.<br>000: Internal selection 0 (IS0)<br>001: Internal selection 1 (IS1)<br>010: Internal selection 2 (IS2)<br>011: Internal selection 3 (IS3)<br>100: C1IRAW input detector (C1INC)<br>101: Filtered input 1 (C1IF1)<br>110: Filtered input 2 (C1IF2)<br>111: External input (EXT)<br>Please refer to Table 54 for more information on ISx for each timer. |
| Bit 3    | Reserved | 0x0         | resd | Kept at its default value.  |



| Bit     | Name  | Reset value | Type | Description   |
|---------|-------|-------------|------|---|
| Bit 2:0 | SMSEL | 0x0         | rw   | Subordinate TMR mode selection<br>000: Slave mode is disabled<br>001: Encoder mode A<br>010: Encoder mode B<br>011: Encoder mode C<br>100: Reset mode – Rising edge of the TRGIN input reinitializes the counter<br>101: Suspend mode – The counter starts counting when the TRGIN is high<br>110: Trigger mode – A trigger event is generated at the rising edge of the TRGIN input<br>111: External clock mode A – Rising edge of the TRGIN input clocks the counter<br>Note: Please refer to count mode section for the details on encoder mode A/B/C. |

#### TMR15 DMA/interrupt enable register (TMR15\_IDEN)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15    | Reserved | 0x0         | resd | Kept at its default value.                                     |
| Bit 14    | TDEN     | 0x0         | rw   | Trigger DMA request enable<br>0: Disabled<br>1: Enabled        |
| Bit 13    | HALLDE   | 0x0         | rw   | HALL DMA request enable<br>0: Disabled<br>1: Enabled           |
| Bit 12:11 | Reserved | 0x0         | resd | Kept at its default value.                                     |
| Bit 10    | C2DEN    | 0x0         | rw   | Channel 2 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 9     | C1DEN    | 0x0         | rw   | Channel 1 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 8     | OVFDEN   | 0x0         | rw   | Overflow event DMA request enable<br>0: Disabled<br>1: Enabled |
| Bit 7     | BRKIE    | 0x0         | rw   | Brake interrupt enable<br>0: Disabled<br>1: Enabled            |
| Bit 6     | TIEN     | 0x0         | rw   | Trigger interrupt enable<br>0: Disabled<br>1: Enabled          |
| Bit 5     | HALLIEN  | 0x0         | rw   | HALL interrupt enable<br>0: Disabled<br>1: Enabled             |
| Bit 4:3   | Reserved | 0x0         | resd | Kept at its default value.                                     |
| Bit 2     | C2IEN    | 0x0         | rw   | Channel 2 interrupt enable<br>0: Disabled<br>1: Enabled        |
| Bit 1     | C1IEN    | 0x0         | rw   | Channel 1 interrupt enable<br>0: Disabled<br>1: Enabled        |

| Bit   | Name   | Reset value | Type | Description  |
|-------|--------|-------------|------|--|
| Bit 0 | OVFIEN | 0x0         | rw   | Overflow interrupt enable<br>0: Disabled<br>1: Enabled |

#### TMR15 interrupt status register (TMR15\_ISTS)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:11 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 10    | C2RF     | 0x0         | rw0c | Channel 2 recapture flag<br>Please refer to C1RF description.  |
| Bit 9     | C1RF     | 0x0         | rw0c | Channel 1 recapture flag<br>This bit indicates whether a recapture is detected when C1IF=1.<br>This bit is set by hardware, and cleared by writing "0".<br>0: No capture is detected<br>1: Capture is detected.  |
| Bit 8     | Reserved | 0x0         | resd | Default value  |
| Bit 7     | BRKIF    | 0x0         | rw0c | Brake interrupt flag<br>This bit indicates whether the brake input is active or not. It is set by hardware and cleared by writing "0"<br>0: Inactive level<br>1: Active level  |
| Bit 6     | TRGIF    | 0x0         | rw0c | Trigger interrupt flag<br>This bit is set by hardware on a trigger event. It is cleared by writing "0".<br>0: No trigger event occurs<br>1: Trigger event is generated.<br>Trigger event: an active edge is detected on TRGIN input, or any edge in suspend mode.  |
| Bit 5     | HALLIF   | 0x0         | rw0c | HALL interrupt flag<br>This bit is set by hardware on HALL event. It is cleared by writing "0".<br>0: No Hall event occurs.<br>1: Hall event is detected.<br>HALL even: CxEN, CxCEN and CxOCTRL are updated.   |
| Bit 4:3   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 2     | C2IF     | 0x0         | rw0c | Channel 2 interrupt flag<br>Please refer to C1IF description.  |
| Bit 1     | C1IF     | 0x0         | rw0c | Channel 1 interrupt flag<br>If the channel 1 is configured as input mode:<br>This bit is set by hardware on a capture event. It is cleared by software or read access to the TMRx_C1DT<br>0: No capture event occurs<br>1: Capture event is generated<br>If the channel 1 is configured as output mode:<br>This bit is set by hardware on a compare event. It is cleared by software.<br>0: No compare event occurs<br>1: Compare event is generated |

| Bit   | Name  | Reset value | Type | Description   |
|-------|-------|-------------|------|---|
| Bit 0 | OVFIF | 0x0         | rw0c | <p>Overflow interrupt flag</p> <p>This bit is set by hardware on an overflow event. It is cleared by software.</p> <p>0: No overflow event occurs</p> <p>1: Overflow event is generated. If OVFE = 0 and OVFS = 0 in the TMRx_CTRL1 register:</p> <ul style="list-style-type: none"> <li>– An overflow event is generated when OVFG = 1 in the TMRx_SWEVE register;</li> <li>– An overflow event is generated when the counter CVAL is reinitialized by a trigger event.</li> </ul> |

#### TMR15 software event register (TMR15\_SWEVT)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:8 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 7    | BRKSWTR  | 0x0         | wo   | <p>Brake event triggered by software</p> <p>This bit is set by software to generate a brake event.</p> <p>0: No effect</p> <p>1: Generate a brake event.</p>  |
| Bit 6    | TRGSWTR  | 0x0         | rw   | <p>Trigger event triggered by software</p> <p>This bit is set by software to generate a trigger event.</p> <p>0: No effect</p> <p>1: Generate a trigger event.</p>  |
| Bit 5    | HALLSWTR | 0x0         | wo   | <p>HALL event triggered by software</p> <p>This bit is set by software to generate a HALL event.</p> <p>0: No effect</p> <p>1: Generate a HALL event.</p> <p>Note: This bit acts only on channels that have complementary output.</p> |
| Bit 4:3  | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 2    | C2SWTR   | 0x0         | wo   | <p>Channel 2 event triggered by software</p> <p>Please refer to C1M description</p>   |
| Bit 1    | C1SWTR   | 0x0         | wo   | <p>Channel 1 event triggered by software</p> <p>This bit is set by software to generate a channel 1 event.</p> <p>0: No effect</p> <p>1: Generate a channel 1 event.</p>  |
| Bit 0    | OVFSWTR  | 0x0         | wo   | <p>Overflow event triggered by software</p> <p>This bit is set by software to generate an overflow event.</p> <p>0: No effect</p> <p>1: Generate an overflow event.</p>   |

#### TMR15 channel mode register1 (TMR15\_CM1)

The channel can be used in input (capture mode) or output (compare mode). The direction of a channel is defined by the corresponding CxC bits. All the other bits of this register have different functions in input and output modes. The CxOx describes its function in output mode when the channel is in output mode, while the CxIx describes its function in output mode when the channel is in input mode. Attention must be given to the fact that the same bit can have different functions in input mode and output mode.

**Output compare mode:**

| Bit       | Name    | Reset value | Type | Description   |
|-----------|---------|-------------|------|---|
| Bit 15    | C2OSEN  | 0x0         | rw   | Channel 2 output switch enable  |
| Bit 14:12 | C2OCTRL | 0x0         | rw   | Channel 2 output control  |
| Bit 11    | C2OBEN  | 0x0         | rw   | Channel 2 output buffer enable  |
| Bit 10    | C2OIEN  | 0x0         | rw   | Channel 2 output enable immediately   |
| Bit 9:8   | C2C     | 0x0         | rw   | Channel 2 configuration<br>This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN = '0':<br>00: Output<br>01: Input, C2IN is mapped on C2IFP2<br>10: Input, C2IN is mapped on C1IFP2<br>11: Input, C2IN is mapped on STI. This mode works only when the internal trigger input is selected by STIS register.   |
| Bit 7     | C1OSEN  | 0x0         | rw   | Channel 1 output switch enable<br>0: C1ORAW is not affected by EXT input.<br>1: Once a high level is detect on EXT input, clear C1ORAW.   |
| Bit 6:4   | C1OCTRL | 0x0         | rw   | Channel 1 output control<br>This field defines the behavior of the original signal C1ORAW.<br>000: Disconnected. C1ORAW is disconnected from C1OUT;<br>001: C1ORAW is high when TMRx_CVAL = TMRx_C1DT<br>010: C1ORAW is low when TMRx_CVAL = TMRx_C1DT<br>011: Switch C1ORAW level when TMRx_CVAL = TMRx_C1DT<br>100: C1ORAW is forced low<br>101: C1ORAW is forced high.<br>110: PWM mode A<br>– OWCDIR = 0, C1ORAW is high once TMRx_C1DT > TMRx_CVAL, else low;<br>– OWCDIR = 1, C1ORAW is low once TMRx_C1DT < TMRx_CVAL, else high;<br>111: PWM mode B<br>– OWCDIR = 0, C1ORAW is low once TMRx_C1DT > TMRx_CVAL, else high;<br>– OWCDIR = 1, C1ORAW is high once TMRx_C1DT < TMRx_CVAL, else low.<br>Note: In the configurations other than 000', the C1OUT is connected to C1ORAW. The C1OUT output level is not only subject to the changes of C1ORAW, but also the output polarity set by CCTRL. |
| Bit 3     | C1OBEN  | 0x0         | rw   | Channel 1 output buffer enable<br>0: Buffer function of TMRx_C1DT is disabled. The new value written to the TMRx_C1DT takes effect immediately.<br>1: Buffer function of TMRx_C1DT is enabled. The value to be written to the TMRx_C1DT is stored in the buffer register, and can be sent to the TMRx_C1DT register only on an overflow event.  |

| Bit     | Name  | Reset value | Type | Description   |
|---------|-------|-------------|------|---|
| Bit 2   | C1OEN | 0x0         | rw   | Channel 1 output enable immediately<br>In PWM mode A or B, this bit is used to accelerate the channel 1 output's response to the trigger event.<br>0: Need to compare the CVAL with C1DT before generating an output<br>1: No need to compare the CVAL and C1DT. An output is generated immediately when a trigger event occurs.                                      |
| Bit 1:0 | C1C   | 0x0         | rw   | Channel 1 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':<br>00: Output<br>01: Input, C1IN is mapped on C1IFP1<br>10: Input, C1IN is mapped on C2IFP1<br>11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS. |

#### Input capture mode:

| Bit       | Name   | Reset value | Type | Description  |
|-----------|--------|-------------|------|--|
| Bit 15:12 | C2DF   | 0x0         | rw   | Channel 2 digital filter   |
| Bit 11:10 | C2IDIV | 0x0         | rw   | Channel 2 input divider  |
| Bit 9:8   | C2C    | 0x0         | rw   | Channel 2 configuration<br>This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN = '0':<br>00: Output<br>01: Input, C2IN is mapped on C2IFP2<br>10: Input, C2IN is mapped on C1IFP2<br>11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.  |
| Bit 7:4   | C1DF   | 0x0         | rw   | Channel 1 digital filter<br>This field defines the digital filter of the channel 1. N stands for the number of filtering, indicating that the input edge can pass the filter only after N sampling events.<br>0000: No filter, sampling is done at $f_{DTS}$<br>0001: $f_{SAMPLING} = f_{CK\_INT}$ , N = 2<br>0010: $f_{SAMPLING} = f_{CK\_INT}$ , N = 4<br>0011: $f_{SAMPLING} = f_{CK\_INT}$ , N = 8<br>0100: $f_{SAMPLING} = f_{DTS}/2$ , N = 6<br>0101: $f_{SAMPLING} = f_{DTS}/2$ , N = 8<br>0110: $f_{SAMPLING} = f_{DTS}/4$ , N = 6<br>0111: $f_{SAMPLING} = f_{DTS}/4$ , N = 8<br>1000: $f_{SAMPLING} = f_{DTS}/8$ , N = 6<br>1001: $f_{SAMPLING} = f_{DTS}/8$ , N = 8<br>1010: $f_{SAMPLING} = f_{DTS}/16$ , N = 5<br>1011: $f_{SAMPLING} = f_{DTS}/16$ , N = 6<br>1100: $f_{SAMPLING} = f_{DTS}/16$ , N = 8<br>1101: $f_{SAMPLING} = f_{DTS}/32$ , N = 5<br>1110: $f_{SAMPLING} = f_{DTS}/32$ , N = 6<br>1111: $f_{SAMPLING} = f_{DTS}/32$ , N = 8 |

| Bit     | Name  | Reset value | Type | Description  |
|---------|-------|-------------|------|--|
| Bit 3:2 | C1DIV | 0x0         | rw   | Channel 1 input divider<br>This field defines Channel 1 input divider.<br>00: No divider. An input capture is generated at each active edge.<br>01: An input compare is generated every 2 active edges<br>10: An input compare is generated every 4 active edges<br>11: An input compare is generated every 8 active edges<br>Note: the divider is reset once C1EN = '0' |
| Bit 1:0 | C1C   | 0x0         | rw   | Channel 1 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':<br>00: Output<br>01: Input, C1IN is mapped on C1IFP1<br>10: Input, C1IN is mapped on C2IFP1<br>11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.    |

#### TMR15 Channel control register (TMR15\_CTRL)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:6 | Reserved | 0x0         | resd | Kept its default value.   |
| Bit 5    | C2P      | 0x0         | rw   | Channel 2 polarity<br>Please refer to C1P description.  |
| Bit 4    | C2EN     | 0x0         | rw   | Channel 2 enable<br>Please refer to C1EN description.   |
| Bit 3    | C1CP     | 0x0         | rw   | Channel 1 complementary polarity<br>0: C1COUT is active high.<br>1: C1COUT is active low.   |
| Bit 2    | C1CEN    | 0x0         | rw   | Channel 1 complementary enable<br>0: Output is disabled.<br>1: Output is enabled.   |
| Bit 1    | C1P      | 0x0         | rw   | Channel 1 polarity<br>When the channel 1 is configured as output mode:<br>0: C1OUT is active high<br>1: C1OUT is active low<br>When the channel 1 is configured as input mode:<br>0: C1IN active edge is on its rising edge. When used as external trigger, C1IN is not inverted.<br>1: C1IN active edge is on its falling edge. When used as external trigger, C1IN is inverted. |
| Bit0     | C1EN     | 0x0         | rw   | Channel 1 enable<br>0: Input or output is disabled<br>1: Input or output is enabled   |

**Table 56. Complementary output channel CxOUT and CxCOUT control bits with brake function**

| Control bit |             |           |          |           | Output state <sup>(1)</sup>  |  |
|-------------|-------------|-----------|----------|-----------|--|--|
| OEN bit     | FCSODIS bit | FCSEN bit | CxEN bit | CxCEN bit | CxOUT output state   | CxCOUT output state  |
| 1           | X           | 0         | 0        | 0         | Output disabled<br>(no driven by the timer)<br>CxOUT = 0, Cx_EN = 0  | Output disabled<br>(no driven by the timer)<br>CxCOUT = 0, CxCEN = 0             |
|             |             | 0         | 0        | 1         | Output disabled<br>(no driven by the timer)<br>CxOUT = 0, Cx_EN = 0  | CxORAW + polarity,<br>CxCOUT = CxORAW xor CxCP,<br>CxCEN = 1                     |
|             |             | 0         | 1        | 0         | CxORAW + polarity<br>CxOUT = CxORAW xor CxP,<br>Cx_EN = 1  | Output disabled<br>(no driven by the timer)<br>CxCOUT = 0, CxCEN = 0             |
|             |             | 0         | 1        | 1         | CxORAW + polarity + dead-time,<br>Cx_EN = 1  | CxORAW inverted + polarity +<br>dead-time,<br>CxCEN = 1                          |
|             |             | 1         | 0        | 0         | Output disabled<br>(no driven by the timer)<br>CxOUT = CxP, Cx_EN = 0  | Output disabled<br>(no driven by the timer)<br>CxCOUT = CxCP, CxCEN = 0          |
|             |             | 1         | 0        | 1         | Off-state<br>(Output enabled with inactive level)<br>CxOUT = CxP, Cx_EN = 1  | CxORAW + polarity,<br>CxCOUT = CxORAW xor CxCP,<br>CxCEN = 1                     |
|             |             | 1         | 1        | 0         | CxORAW + polarity,<br>CxOUT = CxORAW xor CxP,<br>Cx_EN = 1   | Off-state<br>(Output enabled with inactive<br>level)<br>CxCOUT = CxCP, CxCEN = 1 |
|             |             | 1         | 1        | 1         | CxORAW + polarity + dead-time,<br>Cx_EN = 1  | CxORAW inverted + polarity +<br>dead-time,<br>CxCEN = 1                          |
| 0           | 0           | X         | 0        | 0         | Output disabled  |  |
|             | 0           |           | 0        | 1         | (corresponding IO is not driven by the timer, IO floating)   |  |
|             | 0           |           | 1        | 0         | Asynchronously: CxOUT = CxP, Cx_EN = 0, CxCOUT = CxCP,<br>CxCEN = 0;   |  |
|             | 0           |           | 1        | 1         | If the clock is present: after a dead-time, CxOUT = CxIOS, CxCOUT<br>= CxCIOS, assuming that CxIOS and CxCIOS do not correspond to<br>CxOUT and CxCOUT active level. |  |
|             | 1           |           | 0        | 0         | CxEN = CxCEN = 0: output disabled (corresponding IO is not driven<br>by the timer, IO floating)  |  |
|             | 1           |           | 0        | 1         | In other cases: off-state (corresponding channel output invalid level)   |  |
|             | 1           |           | 1        | 0         | Asynchronously: CxOUT = CxP, Cx_EN = 1, CxCOUT = CxCP,<br>CxCEN = 1;   |  |
|             | 1           |           | 1        | 1         | If the clock is present: after a dead-time, CxOUT = CxIOS, CxCOUT<br>= CxCIOS, assuming that CxIOS and CxCIOS do not correspond to<br>CxOUT and CxCOUT active level. |  |

Note: 1. If the two outputs of a channel are not used (CxEN = CxCEN = 0), CxIOS, CxCIOS, CxP and CxCP must be cleared.

2. The state of the external I/O pins connected to the complementary CxOUT and CxCOUT channels depends on the CxOUT and CxCOUT channel state and the GPIO and the IOMUX registers.

#### TMR15 counter value (TMR15\_CVAL)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---------------|
| Bit 15:0 | CVAL | 0x0         | rw   | Counter value |

#### TMR15 division value (TMR15\_DIV)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | DIV  | 0x0         | rw   | Divider value<br>The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK} / (DIV[15:0] + 1)$ .<br>The value of this register is transferred to the actual prescaler register when an overflow event occurs. |

#### TMR15 period register (TMR15\_PR)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | PR   | 0x0         | rw   | Period value<br>This defines the period value of the TMRx counter. The timer stops working when the period value is 0. |

#### TMR15 repetition period register (TMR15\_RPR)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:8 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 7:0  | RPR      | 0x0         | rw   | Repetition of period value<br>This field is used to reduce the generation rate of overflow events. An overflow event is generated when the repetition counter reaches 0. |

#### TMR15 channel 1 data register (TMR15\_C1DT)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | C1DT | 0x0         | rw   | Channel 1 data register<br>When the channel 1 is configured as input mode:<br>The C1DT is the CVAL value stored by the last channel 1 input event (C1IN)<br>When the channel 1 is configured as output mode:<br>C1DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C1OBEN bit, and the corresponding output is generated on C1OUT as configured. |

#### TMR15 channel 2 data register (TMR15\_C2DT)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | C2DT | 0x0         | rw   | Channel 2 data register<br>When the channel 2 is configured as input mode:<br>The C2DT is the CVAL value stored by the last channel 2 input event (C1IN)<br>When the channel 2 is configured as output mode:<br>C2DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C2OBEN bit, and the corresponding output is generated on C2OUT as configured. |



### TMR15 brake register (TMR15\_BRK)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:17 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 19:16 | BKF      | 0x0         | rw   | Brake input filter<br>This field is used to set the filter for brake input. The filter number N indicates that the input edge can pass through filter only after N sampling events.<br>0000: No filter, sampling is done at $f_{DTS}$<br>0001: $f_{SAMPLING} = f_{CK\_INT}$ , $N = 2$<br>0010: $f_{SAMPLING} = f_{CK\_INT}$ , $N = 4$<br>0011: $f_{SAMPLING} = f_{CK\_INT}$ , $N = 8$<br>0100: $f_{SAMPLING} = f_{DTS}/2$ , $N = 6$<br>0101: $f_{SAMPLING} = f_{DTS}/2$ , $N = 8$<br>0110: $f_{SAMPLING} = f_{DTS}/4$ , $N = 6$<br>0111: $f_{SAMPLING} = f_{DTS}/4$ , $N = 8$<br>1000: $f_{SAMPLING} = f_{DTS}/8$ , $N = 6$<br>1001: $f_{SAMPLING} = f_{DTS}/8$ , $N = 8$<br>1010: $f_{SAMPLING} = f_{DTS}/16$ , $N = 5$<br>1011: $f_{SAMPLING} = f_{DTS}/16$ , $N = 6$<br>1100: $f_{SAMPLING} = f_{DTS}/16$ , $N = 8$<br>1101: $f_{SAMPLING} = f_{DTS}/32$ , $N = 5$<br>1110: $f_{SAMPLING} = f_{DTS}/32$ , $N = 6$<br>1111: $f_{SAMPLING} = f_{DTS}/32$ , $N = 8$ |
| Bit 15    | OEN      | 0x0         | rw   | Output enable<br>This bit acts on the channels as output. It is used to enable CxOUT and CxCOUT outputs.<br>0: Disabled<br>1: Enabled   |
| Bit 14    | AOEN     | 0x0         | rw   | Automatic output enable<br>OEN is set automatically at an overflow event.<br>0: Disabled<br>1: Enabled  |
| Bit 13    | BRKV     | 0x0         | rw   | Brake input validity<br>This bit is used to select the active level of a brake input.<br>0: Brake input is active low.<br>1: Brake input is active high.  |
| Bit 12    | BRKEN    | 0x0         | rw   | Brake enable<br>This bit is used to enable brake input.<br>0: Brake input is disabled.<br>1: Brake input is enabled.  |
| Bit 11    | FCSOEN   | 0x0         | rw   | Frozen channel status when holistic output enable<br>This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN = 1.<br>0: CxOUT/CxCOUT outputs are disabled.<br>1: CxOUT/CxCOUT outputs are enabled. Output inactive level.  |
| Bit 10    | FCSODIS  | 0x0         | rw   | Frozen channel status when holistic output disable<br>This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN = 0.<br>0: CxOUT/CxCOUT outputs are disabled.<br>1: CxOUT/CxCOUT outputs are enabled. Output idle level.   |

| Bit     | Name | Reset value | Type | Description   |
|---------|------|-------------|------|---|
| Bit 9:8 | WPC  | 0x0         | rw   | <p>Write protection configuration</p> <p>This field is used to enable write protection.</p> <p>00: Write protection is OFF.</p> <p>01: Write protection level 3, and the following bits are write protected:</p> <p>TMRx_BRK: DTC, BRKEN, BRKV and AOEN</p> <p>TMRx_CTRL2: CxIOS and CxCIOS</p> <p>10: Write protection level 2. The following bits and all bits in level 3 are write protected:</p> <p>TMRx_CCTRL: CxP and CxCP</p> <p>TMRx_BRK: FCSODIS and FCSOEN</p> <p>11: Write protection level 1. The following bits and all bits in level 2 are write protected:</p> <p>TMRx_CMx: C2OCTRL and C2OBEN</p> <p>Note: Once WPC &gt; 0, its content remains frozen until the next system reset.</p> |
| Bit 7:0 | DTC  | 0x00        | rw   | <p>Dead-time configuration</p> <p>This field defines the duration of the dead-time insertion. The 3-bit MSB of DTC[7:0] is used for function selection:</p> <p>0xx: <math>DT = DTC[7:0] \times TDTS</math></p> <p>10x: <math>DT = (64 + DTC[5:0]) \times TDTS \times 2</math></p> <p>110: <math>DT = (32 + DTC[4:0]) \times TDTS \times 8</math></p> <p>111: <math>DT = (32 + DTC[4:0]) \times TDTS \times 16</math></p>  |

Note: Based on lock configuration, AOEN, BRKV, BRKEN, FCSODIS, FCSOEN and DTC[7:0] can all be write protected. Thus it is necessary to configure write protection when writing to the TMRx\_BRK register for the first time.

#### TMR15 DMA control register (TMR15\_DMACTRL)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15:13 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 12:8  | DTB      | 0x0         | rw   | <p>DMA transfer bytes</p> <p>This field defines the number of DMA transfers:</p> <p>00000: 1 byte      00001: 2 bytes</p> <p>00010: 3 bytes     00011: 4 bytes</p> <p>.....</p> <p>10000: 17 bytes    10001: 18 bytes</p> |
| Bit 7:5   | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 4:0   | ADDR     | 0x0         | rw   | <p>DMA transfer address offset</p> <p>ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register:</p> <p>00000: TMRx_CTRL1</p> <p>00001: TMRx_CTRL2</p> <p>00010: TMRx_STCTRL</p> <p>.....</p>     |

#### TMR15 DMA data register (TMR15\_DMADT)

| Bit      | Name  | Reset value | Type | Description  |
|----------|-------|-------------|------|--|
| Bit 15:0 | DMADT | 0x0         | rw   | <p>DMA data register</p> <p>A write/read operation to the DMADT register accesses any TMR register located at the following address:</p> <p>TMRx peripheral address + ADDR × 4 to TMRx peripheral address + ADDR × 4 + DTB × 4</p> |

## General-purpose timers (TMR16 and TMR17)

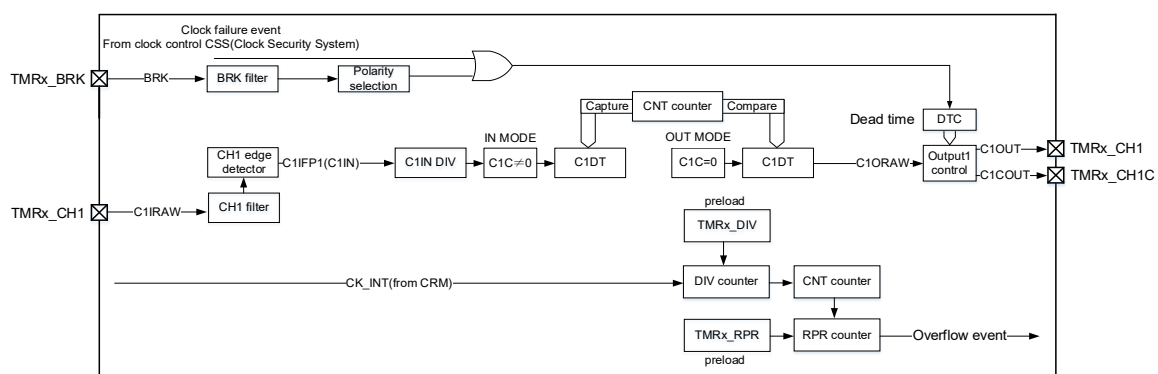
## TMR16 and TMR17 introduction

Each of the general-purpose timers (TMR16 and TMR17) consists of a 16-bit upcounter, a capture/compare register, and an independent channel to achieve embedded dead-time, input capture and programmable PWM output.

## TMR16 and TMR17 main features

The main functions of general-purpose TMR16 and TMR17 include:

- 16-bit upcounter and 8-bit repetition counter
- 1x independent channel for input capture, output compare, PWM generation, one-pulse mode output and embedded dead-time
- 1x independent channel for complementary output
- TMR brake function
- Synchronization control between master and slave timers
- Interrupt/DMA is generated at overflow event, trigger event, brake signal input and channel event
- Support TMR burst DMA transfer



**Figure 161. Block diagram of TMR16 and TMR17 timer**

## TMR16 and TMR17 function overview

## Counting clock

The count clock of TMR16 and TMR17 can only be provided by the internal clock (CK\_INT).



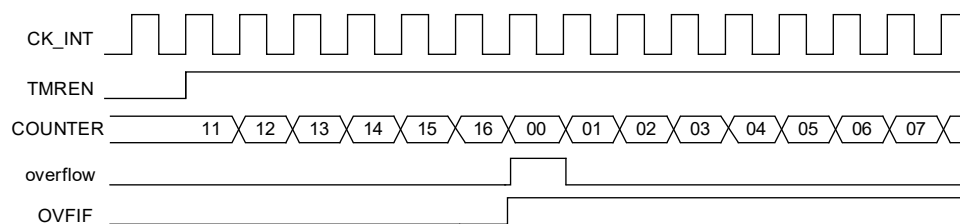
### Figure 162. Counting clock

### Internal clock (CK\_INT)

By default, the CK\_INT, which is divided by the prescaler, is used to drive the counter to start counting. When TMR's APB clock prescaler factor is 1, the CK\_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

Follow the configuration steps below:

- Set counting frequency through TMRx\_DIV register.
- Set counting cycles through TMRx\_PR register.
- Enable a counter by setting the TMREN bit in the TMRx\_CTRL1 register.



**Figure 163. Control circuit with CK\_INT, TMRx\_DIV = 0x0 and TMRx\_PR = 0x16**

### Counting mode

TMR16 and TMR17 support multiple counting modes to meet various application scenarios. Each of them consists of a 16-bit upcounter.

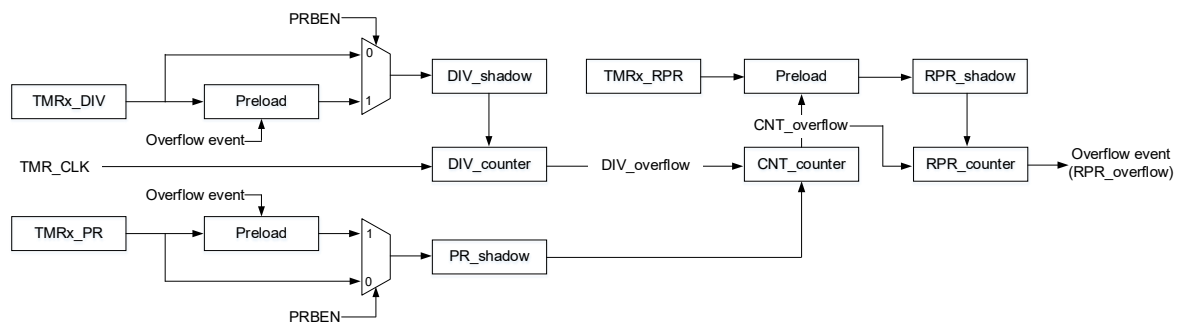
The TMRx\_PR register is used to define counting period of counter. The value in the TMRx\_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN = 1), the value in the TMRx\_PR register is transferred to the shadow register only at an overflow event.

TMRx\_DIV register is used to define the counter frequency of the counter. The counter counts once every DIV[15:0] + 1 clock cycle. Similar to TMRx\_PR register, after enabling periodic buffer, the value of the TMRx\_DIV register are transferred into the shadow register at each overflow event.

Reading the TMRx\_CNT register returns the current counter value. Writing the TMRx\_CNT register will update the current counter value.

An overflow event is enabled by default. It can be disabled by setting OVFEN = 1 in the TMRx\_CTRL1 register. The OVFS bit in the TMRx\_CTRL1 register is used to select the source of an overflow event, which is, by default, counter overflow or underflow, setting OVFSWTR, reset signal generated by slave mode timer controller in reset mode. Once the OVFS is set, an overflow event is generated only when overflow or underflow occurs.

Setting the TMREN bit (TMREN = 1) enables the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR\_EN is set 1 clock cycle after the TMREN is set

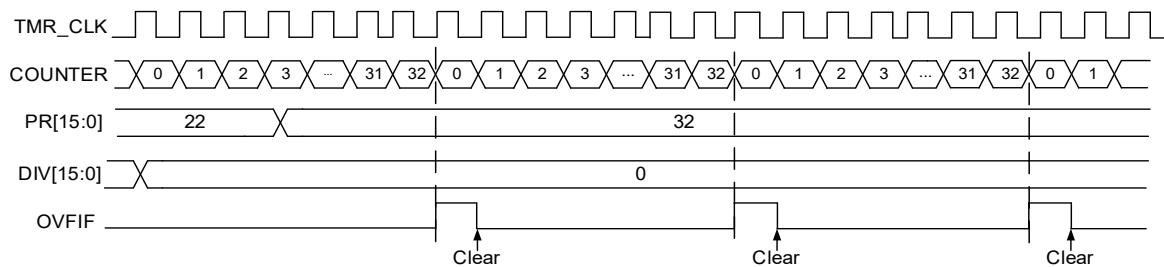


**Figure 164. Basic structure of a counter**

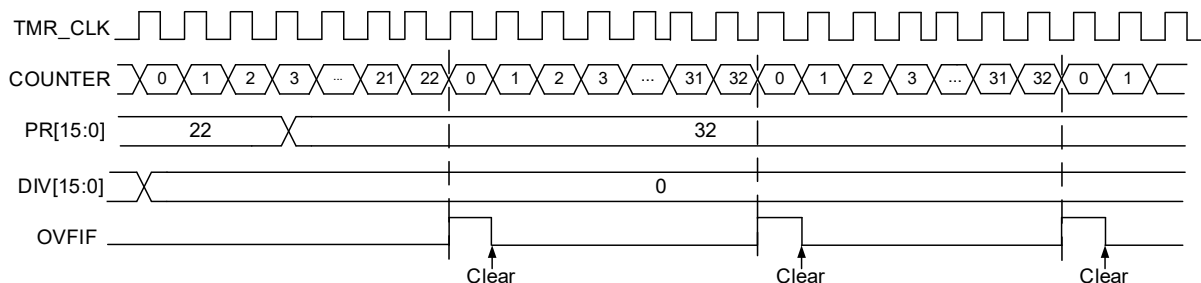
### Upcounting mode

This mode is enabled by setting  $TWCMSEL[1:0] = 2'b00$  and  $OWCDIR = 1'b0$  in the  $TMRx\_CTRL1$  register.

In upcounting mode, the counter counts from 0 to the value programmed in the  $TMRx\_PR$  register, restarts from 0, and generates a counter overflow event, with setting  $OVFIF$  bit to 1. If the overflow event is disabled, the counter is no longer reloaded with the prescaler and re-loaded value on counter overflow; otherwise, the prescaler and re-loaded value will be updated on an overflow event.



**Figure 165. Overflow event when  $PRBEN = 0$**

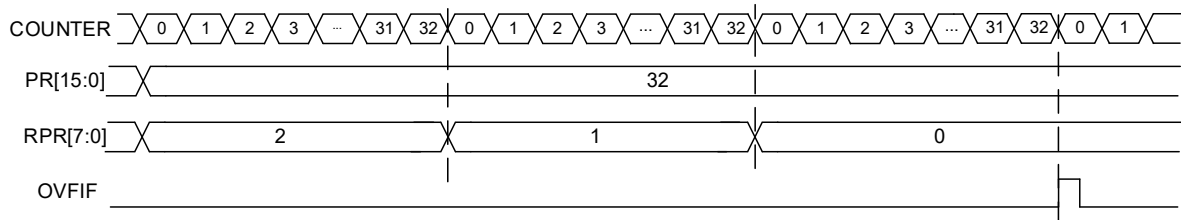


**Figure 166. Overflow event when  $PRBEN = 1$**

### Repetition counter mode:

The  $TMRx\_RPR$  register is used to set repetition counting mode. This mode is enabled when the repetition counter value is not equal to 0. In this mode, an overflow event is generated when a

counter overflow occurs ( $RPR[7:0] + 1$ ). The repetition counter is decremented at each counter overflow. An overflow event is generated when the repetition counter reaches 0. The frequency of the overflow event can be adjusted by setting the repetition counter value.



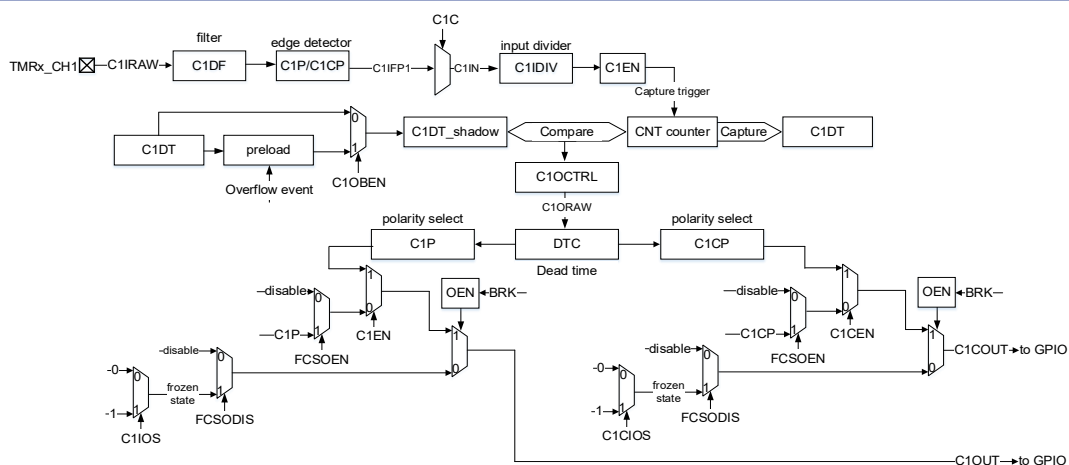
**Figure 167. OVFIF when RPR = 2**

### TMR input function

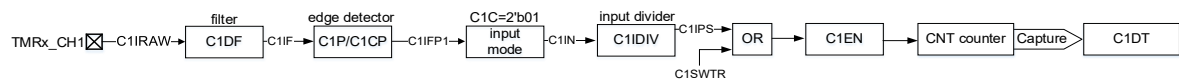
Each timer of TMR16 and TMR17 has one independent channel that can be configured as input or output.

As input, each channel input signal is processed as follows:

- TMRx\_CHx outputs the pre-processed CxIRAW. The C1INSE bit is used to select TMRx\_CHx as the source of C1IRAW.
- CxIRAW inputs digital filter and outputs filtered CxIF signal. The digital filter uses the CxDF bit to to program sampling frequency and sampling times.
- CxIF inputs edge detector, and outputs the CxIFPx signal after edge selection. The edge selection depends on both CxP and CxCP bits. It is possible to select input rising edge, falling edge or both edges.
- CxIFPx inputs capture signal selector, and outputs the CxIN signal after capture signal selection. The capture signal selection is defined by CxC bits. It is possible to select CxIFPx as CxIN source.
- CxIN outputs the CxIPS signal that is divided by input channel divider. The divider factor can be defined as No division, /2, /4 or /8, by the CxIDIV bit.



**Figure 168. Input/output channel 1 main circuit**



**Figure 169. Channel 1 input stage**

### Input mode

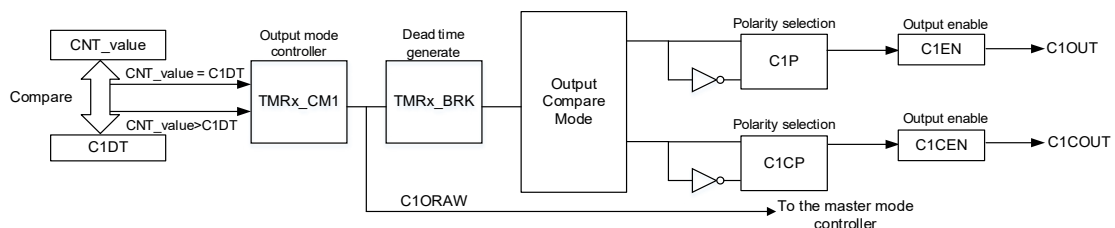
In input mode, the TMRx\_CxDT registers latch the current counter values after the selected trigger signal is detected, and the capture compare interrupt flag bit (CxIF) is set. An interrupt/DMA request will be generated if the CxIEN bit and CxDEN bit are enabled. If the selected trigger signal is detected when the CxIF is set to 1, a capture overflow event is generated, the previous counter value will be overwritten with the current counter value, and the CxRF is set to 1.

To capture the rising edge of C1IN input, following the configuration procedure mentioned below:

- Set C1C = 01 in the TMRx\_CM1 register to select the C1IN as channel 1 input.
- Set C1IN signal filter bandwidth (CxDF[3:0]).
- Set the active edge on the C1IN channel by writing C1P = 0 (rising edge) in the TMRx\_CCTR register.
- Program C1IN signal capture frequency divider (C1DIV[1:0])
- Enable channel 1 input capture (C1EN = 1)
- If needed, enable the relevant interrupt or DMA request by setting the C1IEN bit in the TMRx\_IDEN register or the C1DEN bit in the TMRx\_IDEN register

### TMR output function

The TMR output consists of a comparator and an output controller. It is used to program the period, duty cycle and polarity of the output signal, as shown in Figure 170.



**Figure 170. Channel output stage**

### Output mode

Write CxC[1:0]≠2'b00 to configure the channel as output to implement multiple output modes. In this case, the counter value is compared with the value in the TMRx\_CxDT register, and the intermediate signal CxORAW is generated according to the output mode selected by CxOCTRL[2:0], which is sent to IO after being processed by the output control circuit. The period of the output signal is configured by the TMRx\_PR register, while the duty cycle by the TMRx\_CxDT register.

Output compare modes include:

### PWM mode A:

Enable PWM mode A by setting  $CxOCTRL = 3'b110$ . In upcounting mode, C1ORAW outputs high when  $TMRx\_C1DT > TMRx\_CVAL$ ; otherwise, it is low. In downcounting mode, C1ORAW outputs low when  $TMRx\_C1DT < TMRx\_CVAL$ ; otherwise, it is high.

To use PWM mode A, the following procedures are recommended:

- Set PWM periods through  $TMRx\_PR$  register.
- Set PWM duty cycles through  $TMRx\_CxD$ .
- Select PWM mode A by setting  $CxOCTRL = 3'b110$  in the  $TMRx\_CM1/CM2$  register.
- Set counting frequency through  $TMRx\_DIV$  register.
- Select counting mode by setting the  $TWCMSEL[1:0]$  bit in the  $TMRx\_CTRL1$  register.
- Select output polarity through the  $CxP$  and  $CxCP$  bits in the  $TMRx\_CCTRL$  register.
- Enable channel output through the  $CxEN$  and  $CxCEN$  bits in the  $TMRx\_CCTRL$  register.
- Enable  $TMRx$  output through the  $OEN$  bit in the  $TMRx\_BRK$  register.
- Configure GPIOs corresponding to TMR output channels as multiplexed mode.
- Enable  $TMRx$  to start counting through the  $TMREN$  bit in the  $TMRx\_CTRL1$  register.

### PWM mode B:

Enable PWM mode B by setting  $CxOCTRL = 3'b111$ . In upcounting mode, C1ORAW outputs low when  $TMRx\_C1DT > TMRx\_CVAL$ ; otherwise, it is high. In downcounting mode, C1ORAW outputs high when  $TMRx\_C1DT < TMRx\_CVAL$ ; otherwise, it is low.

### Forced output mode:

Enable forced output mode by setting  $CxOCTRL = 3'b100/101$ . In this case, the CxORAW is forced to be the programmed level, regardless of the counter value. Despite this, the channel flag bit and DMA request still depend on the compare result.

### Output compare mode:

Enable output compare mode by setting  $CxOCTRL = 3'b001/010/011$ . In this case, when the counter value matches the value of the  $CxD$  register, the CxORAW is forced high ( $CxOCTRL = 3'b001$ ), low ( $CxOCTRL = 3'b010$ ) or toggling ( $CxOCTRL = 3'b011$ ).

### One-pulse mode

This is a particular case of PWM mode. Enable one-pulse by setting  $OCMEN = 1$ . In this mode, the comparison match is performed in the current counting period. The  $TMREN$  bit is cleared as soon as the current counting is completed. Therefore, only one pulse is output. When in upcounting mode, the configuration must follow the rule:  $CVAL < CxD \leq PR$ ; in downcounting mode,  $CVAL > CxD$  is required.

### Fast output mode

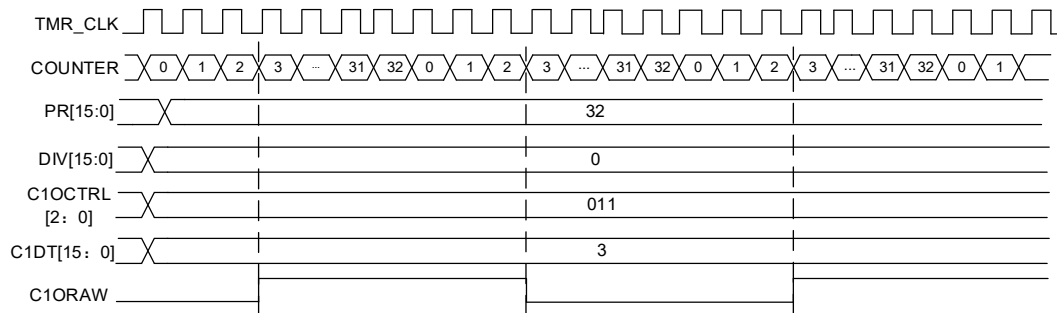
Enable this mode by setting  $CxOIEN = 1$ . If enabled, the CxORAW signal will not change when the counter value matches the  $CxD$ , but change at the beginning of the current counting period. In other words, the comparison result is advanced, so the comparison result between the counter value and the  $TMRx\_CxD$  register will determine the level of CxORAW in advance.

Figure 171 gives an example of output compare mode (toggle) with  $C1DT = 0x3$ . When the counter value is equal to  $0x3$ , C1OUT toggles.

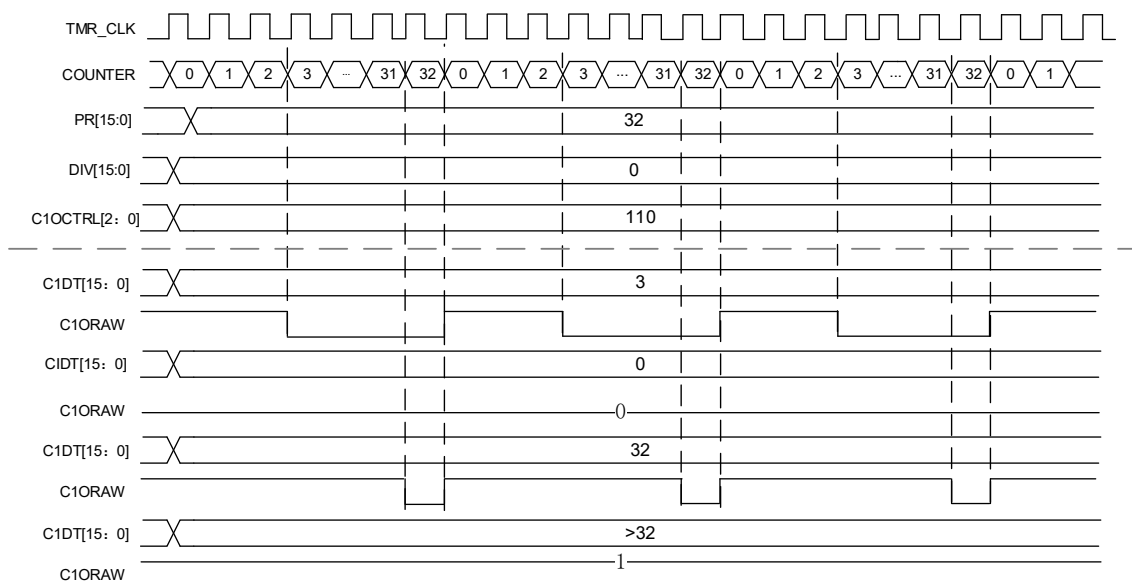


Figure 172 gives an example of the combination between upcounting mode and PWM mode A. The output signal behaves when PR = 0x32 but CxDT is configured with a different value.

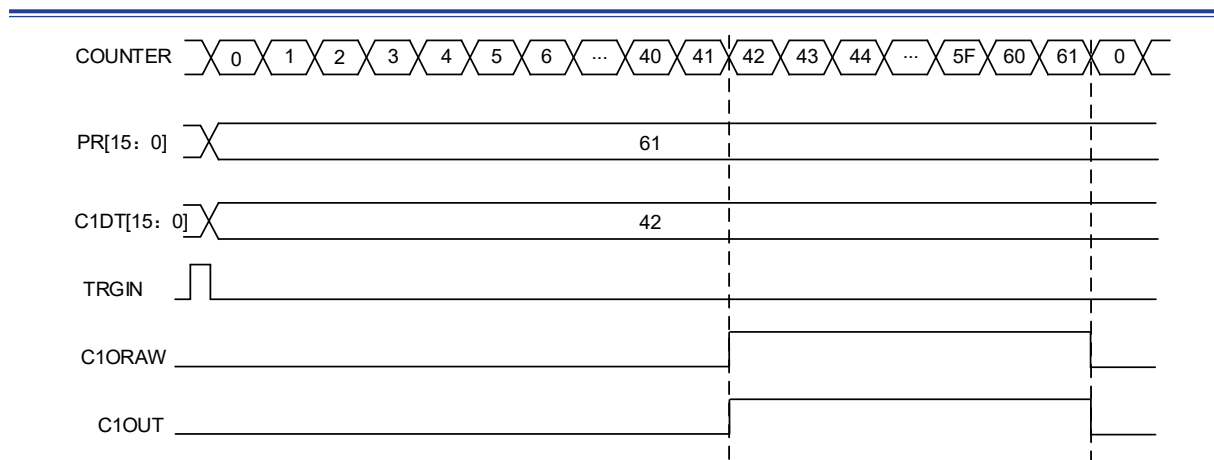
Figure 173 gives an example of the combination between upcounting mode and one-pulse PWM mode B. The counter only counts only one cycle, and the output signal sends only one pulse.



**Figure 171. C1ORAW toggles when counter value matches the C1DT value**



**Figure 172. Upcounting mode and PWM mode A**



**Figure 173. One-pulse mode**

### Dead-time insertion

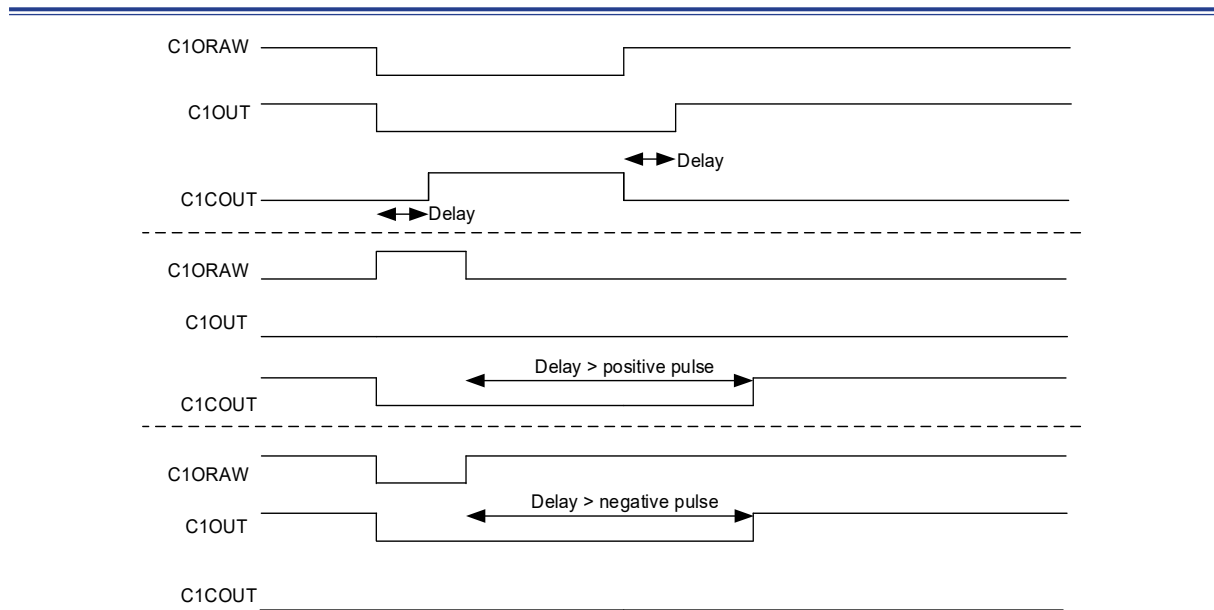
The channel 1 of the TMR16 and TMR17 timers contains a set of reverse channel output. This function is enabled by the CxCEN bit and its polarity is defined by CxCP. Refer to Table 62 for more information about the output state of CxOUT and CxCOUT.

The dead-time is activated when switching to IDLEF state (OEN falling down to 0).

Setting both CxEN and CxCEN bits, and using DTC[7:0] bit to insert dead-time of different durations. After the dead-time insertion, the rising edge of the CxOUT is delayed compared to the rising edge of the reference signal; the rising edge of the CxCOUT is delayed compared to the falling edge of the reference signal.

If the delay is greater than the width of the active output, then the C1OUT and C1COUT will not generate corresponding pulses. Therefore, the dead-time should be less than the width of the active output.

Figure 174 gives an example of dead-time insertion when CxP = 0, CxCP = 0, OEN = 1, CxEN = 1 and CxCEN = 1.



**Figure 174. Complementary output with dead-time insertion**

#### TMR brake function

When the brake function is enabled (BRKEN = 1), the CxOUT and CxCOUT are jointly controlled by OEN, FCSODIS, FCSOEN, CxIOS and CxCIOS. But, CxOUT and CxCOUT cannot be set both to active level at the same time. Please refer Table to 58 for more details.

The brake source can be the brake input pin or a clock failure event. The polarity is controlled by the BRKV bit.

When a brake event occurs, there are the following actions:

- The OEN bit is cleared asynchronously, and the channel output state is selected by setting the FCSODIS bit. This function works even if the MCU oscillator is off.
- Once OEN = 0, the channel output level is defined by the CxIOS bit. If FCSODIS = 0, the timer output is disabled; otherwise, the output enable remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state, that is, 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 active, then the dead-time generator is activated. The CxIOS and CxCIOS bits are used to program the level after dead-time. Even in this case, the CxIOS and CxCIOS cannot be driven to their active level at the same time. It should be note that because of synchronization on OEN, the dead-time duration is usually longer than usual (around 2 clk\_tmr clock cycles).
  - If FCSODIS = 0, the timer releases the enable output; otherwise, it keeps the enable output; the enable output becomes high as soon as one of the CxEN and CxCEN bits becomes high.
- If the brake interrupt or DMA request is enabled, the brake statue flag is set, and a brake interrupt or DMA request can be generated.
- If AOEN = 1, the OEN bit is automatically set again at the next overflow event.

Note: When the brake input is active, the OEN cannot be set, nor the status flag, BRKIF can be cleared.

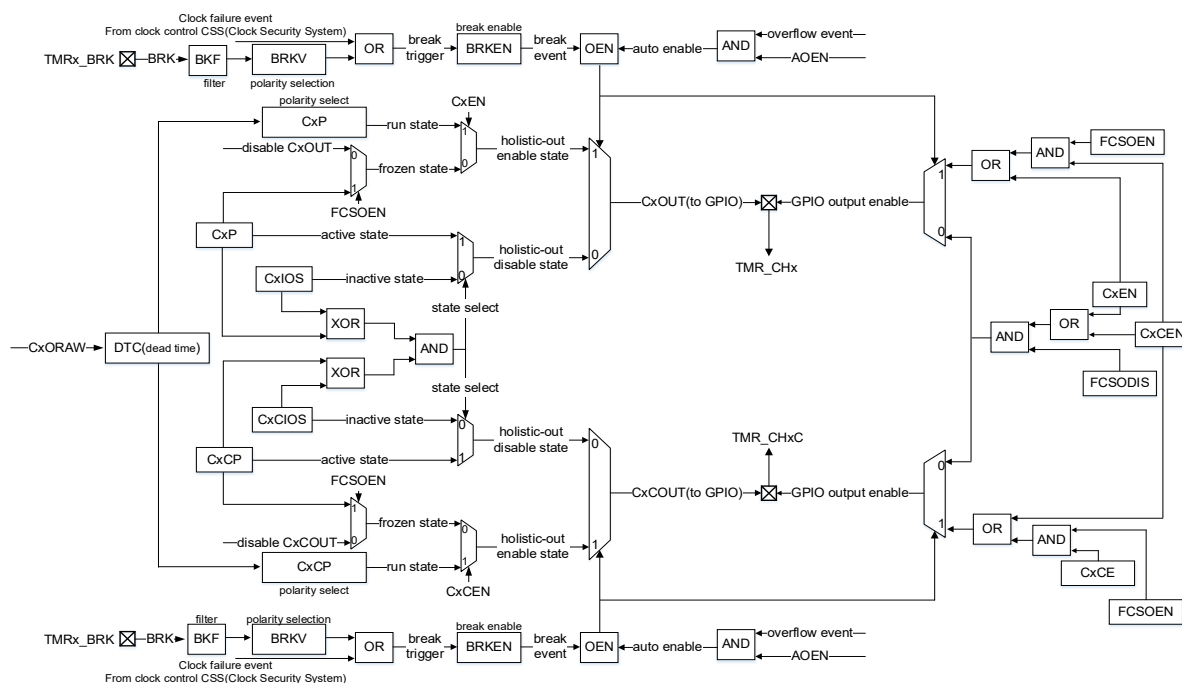


Figure 175. Example of TMR output control

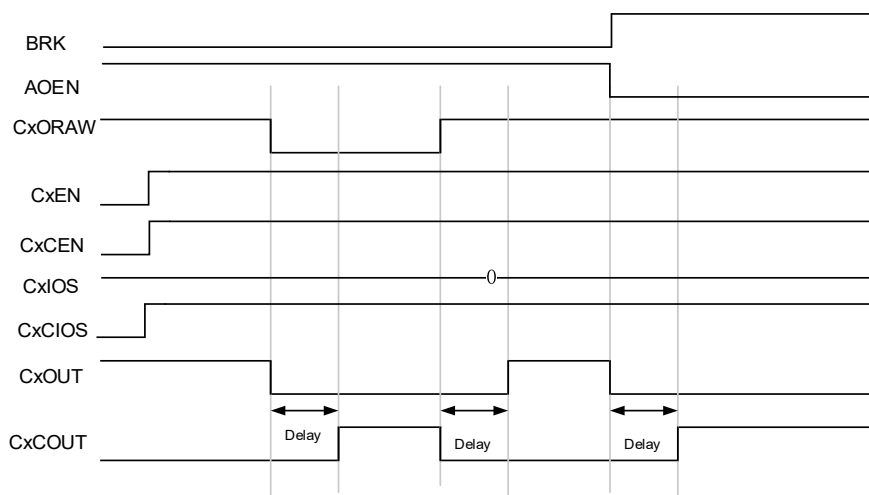


Figure 176. Example of TMR brake function

## Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the TMRx counter stops counting by setting the TMRx\_PAUSE in the DEBUG module.

## TMR16 and TMR17 registers

These peripheral registers must be accessed by word (32 bits).

TMR16 and TMR17 registers are mapped into a 16-bit addressable space.

**Table 57. TMR16 and TMR17 register map and reset value**

| Register     | Offset | Reset value |
|--------------|--------|-------------|
| TMRx_CTRL1   | 0x00   | 0x0000      |
| TMRx_CTRL2   | 0x04   | 0x0000      |
| TMRx_IDEN    | 0x0C   | 0x0000      |
| TMRx_ISTS    | 0x10   | 0x0000      |
| TMRx_SWEVT   | 0x14   | 0x0000      |
| TMRx_CM1     | 0x18   | 0x0000      |
| TMRx_CCTRL   | 0x20   | 0x0000      |
| TMRx_CVAL    | 0x24   | 0x0000      |
| TMRx_DIV     | 0x28   | 0x0000      |
| TMRx_PR      | 0x2C   | 0x0000      |
| TMRx_RPR     | 0x30   | 0x0000      |
| TMRx_C1DT    | 0x34   | 0x0000      |
| TMRx_BRK     | 0x44   | 0x0000      |
| TMRx_DMACTRL | 0x48   | 0x0000      |
| TMRx_DMADT   | 0x4C   | 0x0000      |

### TMR16 and TMR17 control register 1 (TMRx\_CTRL1)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:10 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 9:8   | CLKDIV   | 0x0         | rw   | <p>Clock division</p> <p>This field is used to define the relationship between digital filter sampling frequency (<math>f_{DTS}</math>) and timer clock frequency (<math>f_{CK\_INT}</math>). It is also used to set the ratio relationship between dead time base (<math>T_{DTS}</math>) and timer clock period (<math>T_{CK\_INT}</math>)</p> <p>00: No division, <math>f_{DTS} = f_{CK\_INT}</math><br/>           01: Divided by 2, <math>f_{DTS} = f_{CK\_INT}/2</math><br/>           10: Divided by 4, <math>f_{DTS} = f_{CK\_INT}/4</math><br/>           11: Reserved</p> |
| Bit 7     | PRBEN    | 0x0         | rw   | <p>Period buffer enable</p> <p>0: Period buffer is disabled<br/>           1: Period buffer is enabled</p>   |
| Bit 6:4   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 3     | OCMEN    | 0x0         | rw   | <p>One cycle mode enable</p> <p>This bit is use to select whether to stop counting at an update event</p> <p>0: The counter does not stop at an update event<br/>           1: The counter stops at an update event</p>  |

| Bit   | Name  | Reset value | Type | Description  |
|-------|-------|-------------|------|--|
| Bit 2 | OVFS  | 0x0         | rw   | Overflow event source<br>This bit is used to select overflow event or DMA request sources.<br>0: Counter overflow, setting the OVFSWTR bit or overflow event generated by slave timer controller<br>1: Only counter overflow generates an overflow event |
| Bit 1 | OVFEN | 0x0         | rw   | Overflow event enable<br>0: Enabled<br>1: Disabled   |
| Bit 0 | TMREN | 0x0         | rw   | TMR enable<br>0: Disabled<br>1: Enabled  |

#### TMR16 and TMR17 control register 2 (TMRx\_CTRL2)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 30:10 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 9     | C1CIOS   | 0x0         | rw   | Channel 1 complementary idle output state<br>OEN = 0 after dead-time:<br>0: C1OUTL = 0<br>1: C1OUTL = 1   |
| Bit 8     | C1IOS    | 0x0         | rw   | Channel 1 idle output state<br>OEN = 0 after dead-time:<br>0: C1OUT = 0<br>1: C1OUT = 1   |
| Bit 7:4   | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 3     | DRS      | 0x0         | rw   | DMA request source<br>0: Capture/compare event<br>1: Overflow event   |
| Bit 2     | CCFS     | 0x0         | rw   | Channel control bit flash selection<br>This bit only acts on channels that have complementary output.<br>If the channel control bits are buffered:<br>0: Control bits are updated by setting the HALL bit<br>1: Control bits are updated by setting the HALL bit or a rising edge on TRGIN. |
| Bit 1     | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 0     | CBCTRL   | 0x0         | rw   | Channel buffer control<br>This bit acts on channels that have complementary output.<br>0: CxEN, CxCEN and CxOCTRL bits are not buffered.<br>1: CxEN, CxCEN and CxOCTRL bits are buffered.   |

#### TMR16 and TMR17 DMA/interrupt enable register (TMRx\_IDEN)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:10 | Reserved | 0x0         | resd | Kept at its default value.                                     |
| Bit 9     | C1DEN    | 0x0         | rw   | Channel 1 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 8     | OVFDEN   | 0x0         | rw   | Overflow event DMA request enable<br>0: Disabled<br>1: Enabled |

| Bit     | Name     | Reset value | Type | Description   |
|---------|----------|-------------|------|---|
| Bit 7   | BRKIE    | 0x0         | rw   | Brake interrupt enable<br>0: Disabled<br>1: Enabled     |
| Bit 6   | Reserved | 0x0         | resd | Kept at its default value.                              |
| Bit 5   | HALLIEN  | 0x0         | rw   | HALL interrupt enable<br>0: Disabled<br>1: Enabled      |
| Bit 4:2 | Reserved | 0x0         | resd | Kept at its default value.                              |
| Bit 1   | C1IEN    | 0x0         | rw   | Channel 1 interrupt enable<br>0: Disabled<br>1: Enabled |
| Bit 0   | OVFIEN   | 0x0         | rw   | Overflow interrupt enable<br>0: Disabled<br>1: Enabled  |

#### TMR16 and TMR17 interrupt status register (TMRx\_ISTS)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:10 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 9     | C1RF     | 0x0         | rw0c | Channel 1 recapture flag<br>This bit indicates whether a recapture is detected when C1IF=1. This bit is set by hardware, and cleared by writing "0".<br>0: No capture is detected<br>1: Capture is detected.   |
| Bit 8     | Reserved | 0x0         | resd | Default value  |
| Bit 7     | BRKIF    | 0x0         | rw0c | Brake interrupt flag<br>This bit indicates whether the brake input is active or not. It is set by hardware and cleared by writing "0"<br>0: Inactive level<br>1: Active level  |
| Bit 6     | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 5     | HALLIF   | 0x0         | rw0c | HALL interrupt flag<br>This bit is set by hardware on HALL event. It is cleared by writing "0".<br>0: No Hall event occurs.<br>1: Hall event is detected.<br>HALL even: CxEN, CxCEN and CxOCTRL are updated.   |
| Bit 4:2   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 1     | C1IF     | 0x0         | rw0c | Channel 1 interrupt flag<br>If the channel 1 is configured as input mode:<br>This bit is set by hardware on a capture event. It is cleared by software or read access to the TMRx_C1DT<br>0: No capture event occurs<br>1: Capture event is generated<br>If the channel 1 is configured as output mode:<br>This bit is set by hardware on a compare event. It is cleared by software.<br>0: No compare event occurs<br>1: Compare event is generated |

| Bit   | Name  | Reset value | Type | Description  |
|-------|-------|-------------|------|--|
| Bit 0 | OVFIF | 0x0         | rw0c | <p>Overflow interrupt flag</p> <p>This bit is set by hardware on an overflow event. It is cleared by software.</p> <p>0: No overflow event occurs</p> <p>1: Overflow event is generated. If OVFEN = 0 and OVFS = 0 in the TMRx_CTRL1 register:</p> <ul style="list-style-type: none"> <li>– An overflow event is generated when OVFG = 1 in the TMRx_SWEVE register;</li> <li>– An overflow event is generated when the counter CVAL is reinitialized by a trigger event.</li> </ul> |

#### TMR16 and TMR17 software event register (TMRx\_SWEVT)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:8 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 7    | BRKSWTR  | 0x0         | wo   | <p>Brake event triggered by software</p> <p>This bit is set by software to generate a brake event.</p> <p>0: No effect</p> <p>1: Generate a brake event.</p>  |
| Bit 6    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 5    | HALLSWTR | 0x0         | wo   | <p>HALL event triggered by software</p> <p>This bit is set by software to generate a HALL event.</p> <p>0: No effect</p> <p>1: Generate a HALL event.</p> <p>Note: This bit acts only on channels that have complementary output.</p> |
| Bit 4:2  | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 1    | C1SWTR   | 0x0         | wo   | <p>Channel 1 event triggered by software</p> <p>This bit is set by software to generate a channel 1 event.</p> <p>0: No effect</p> <p>1: Generate a channel 1 event.</p>  |
| Bit 0    | OVFSWTR  | 0x0         | wo   | <p>Overflow event triggered by software</p> <p>This bit is set by software to generate an overflow event.</p> <p>0: No effect</p> <p>1: Generate an overflow event.</p>   |

#### TMR16 and TMR17 channel mode register1 (TMRx\_CM1)

The channel can be used in input (capture mode) or output (compare mode). The direction of a channel is defined by the corresponding CxC bits. All the other bits of this register have different functions in input and output modes. The CxOx describes its function in output mode when the channel is in output mode, while the CxIx describes its function in output mode when the channel is in input mode. Attention must be given to the fact that the same bit can have different functions in input mode and output mode.

##### Output compare mode:

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:8 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 7    | C1OSEN   | 0x0         | rw   | <p>Channel 1 output switch enable</p> <p>0: C1ORAW is not affected by EXT input.</p> <p>1: Once a high level is detect on EXT input, clear C1ORAW.</p> |



| Bit     | Name    | Reset value | Type | Description  |
|---------|---------|-------------|------|--|
| Bit 6:4 | C1OCTRL | 0x0         | rw   | <p>Channel 1 output control</p> <p>This field defines the behavior of the original signal C1ORAW.</p> <p>000: Disconnected. C1ORAW is disconnected from C1OUT;</p> <p>001: C1ORAW is high when TMRx_CVAL = TMRx_C1DT</p> <p>010: C1ORAW is low when TMRx_CVAL = TMRx_C1DT</p> <p>011: Switch C1ORAW level when TMRx_CVAL = TMRx_C1DT</p> <p>100: C1ORAW is forced low</p> <p>101: C1ORAW is forced high.</p> <p>110: PWM mode A</p> <ul style="list-style-type: none"> <li>– OWCDIR = 0, C1ORAW is high once TMRx_C1DT &gt; TMRx_CVAL, else low;</li> <li>– OWCDIR = 1, C1ORAW is low once TMRx_C1DT &lt; TMRx_CVAL, else high;</li> </ul> <p>111: PWM mode B</p> <ul style="list-style-type: none"> <li>– OWCDIR = 0, C1ORAW is low once TMRx_C1DT &gt; TMRx_CVAL, else high;</li> <li>– OWCDIR = 1, C1ORAW is high once TMRx_C1DT &lt; TMRx_CVAL, else low.</li> </ul> <p>Note: In the configurations other than 000', the C1OUT is connected to C1ORAW. The C1OUT output level is not only subject to the changes of C1ORAW, but also the output polarity set by CCTRL.</p> |
| Bit 3   | C1OBEN  | 0x0         | rw   | <p>Channel 1 output buffer enable</p> <p>0: Buffer function of TMRx_C1DT is disabled. The new value written to the TMRx_C1DT takes effect immediately.</p> <p>1: Buffer function of TMRx_C1DT is enabled. The value to be written to the TMRx_C1DT is stored in the buffer register, and can be sent to the TMRx_C1DT register only on an overflow event.</p>  |
| Bit 2   | C1OIEN  | 0x0         | rw   | <p>Channel 1 output enable immediately</p> <p>In PWM mode A or B, this bit is used to accelerate the channel 1 output's response to the trigger event.</p> <p>0: Need to compare the CVAL with C1DT before generating an output</p> <p>1: No need to compare the CVAL and C1DT. An output is generated immediately when a trigger event occurs.</p>  |
| Bit 1:0 | C1C     | 0x0         | rw   | <p>Channel 1 configuration</p> <p>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':</p> <p>00: Output</p> <p>01: Input, C1IN is mapped on C1IFP1</p> <p>10: Reserved</p> <p>11: Reserved</p>  |

**Input capture mode:**

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:8  | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 11:10 | C2IDIV   | 0x0         | rw   | Channel 2 input divider  |
| Bit 7:4   | C1DF     | 0x0         | rw   | Channel 1 digital filter<br>This field defines the digital filter of the channel 1. N stands for the number of filtering, indicating that the input edge can pass the filter only after N sampling events.<br>0000: No filter, sampling is done at $f_{DTS}$<br>0001: $f_{SAMPLING} = f_{CK\_INT}$ , N = 2<br>0010: $f_{SAMPLING} = f_{CK\_INT}$ , N = 4<br>0011: $f_{SAMPLING} = f_{CK\_INT}$ , N = 8<br>0100: $f_{SAMPLING} = f_{DTS}/2$ , N = 6<br>0101: $f_{SAMPLING} = f_{DTS}/2$ , N = 8<br>0110: $f_{SAMPLING} = f_{DTS}/4$ , N = 6<br>0111: $f_{SAMPLING} = f_{DTS}/4$ , N = 8<br>1000: $f_{SAMPLING} = f_{DTS}/8$ , N = 6<br>1001: $f_{SAMPLING} = f_{DTS}/8$ , N = 8<br>1010: $f_{SAMPLING} = f_{DTS}/16$ , N = 5<br>1011: $f_{SAMPLING} = f_{DTS}/16$ , N = 6<br>1100: $f_{SAMPLING} = f_{DTS}/16$ , N = 8<br>1101: $f_{SAMPLING} = f_{DTS}/32$ , N = 5<br>1110: $f_{SAMPLING} = f_{DTS}/32$ , N = 6<br>1111: $f_{SAMPLING} = f_{DTS}/32$ , N = 8 |
| Bit 3:2   | C1IDIV   | 0x0         | rw   | Channel 1 input divider<br>This field defines Channel 1 input divider.<br>00: No divider. An input capture is generated at each active edge.<br>01: An input compare is generated every 2 active edges<br>10: An input compare is generated every 4 active edges<br>11: An input compare is generated every 8 active edges<br>Note: the divider is reset once C1EN = '0'   |
| Bit 1:0   | C1C      | 0x0         | rw   | Channel 1 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':<br>00: Output<br>01: Input, C1IN is mapped on C1IFP1<br>10: Reserved<br>11: Reserved   |

**TMR16 and TMR17 channel control register (TMRx\_CTRL)**

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 15:4 | Reserved | 0x0         | resd | Kept its default value.   |
| Bit 3    | C1CP     | 0x0         | rw   | Channel 1 complementary polarity<br>0: C1COUT is active high.<br>1: C1COUT is active low. |
| Bit 2    | C1CEN    | 0x0         | rw   | Channel 1 complementary enable<br>0: Output is disabled.<br>1: Output is enabled.         |

| Bit   | Name | Reset value | Type | Description   |
|-------|------|-------------|------|---|
| Bit 1 | C1P  | 0x0         | rw   | Channel 1 polarity<br>When the channel 1 is configured as output mode:<br>0: C1OUT is active high<br>1: C1OUT is active low<br>When the channel 1 is configured as input mode:<br>0: C1IN active edge is on its rising edge. When used as external trigger, C1IN is not inverted.<br>1: C1IN active edge is on its falling edge. When used as external trigger, C1IN is inverted. |
| Bit 0 | C1EN | 0x0         | rw   | Channel 1 enable<br>0: Input or output is disabled<br>1: Input or output is enabled   |

**Table 58. Complementary output channel CxOUT and CxCOUT control bits with brake function**

| Control bit |             |           |          |           | Output state <sup>(1)</sup>   |  |
|-------------|-------------|-----------|----------|-----------|---|--|
| OEN bit     | FCSODIS bit | FCSEN bit | CxEN bit | CxCEN bit | CxOUT output state  | CxCOUT output state  |
| 1           | X           | 0         | 0        | 0         | Output disabled<br>(no driven by the timer)<br>CxOUT = 0, Cx_EN = 0         | Output disabled<br>(no driven by the timer)<br>CxCOUT = 0, CxCEN = 0             |
|             |             | 0         | 0        | 1         | Output disabled<br>(no driven by the timer)<br>CxOUT = 0, Cx_EN = 0         | CxORAW + polarity,<br>CxCOUT = CxORAW xor CxCP,<br>CxCEN = 1                     |
|             |             | 0         | 1        | 0         | CxORAW + polarity<br>CxOUT = CxORAW xor CxP,<br>Cx_EN = 1                   | Output disabled<br>(no driven by the timer)<br>CxCOUT = 0, CxCEN = 0             |
|             |             | 0         | 1        | 1         | CxORAW + polarity + dead-time,<br>Cx_EN = 1                                 | CxORAW inverted + polarity +<br>dead-time,<br>CxCEN = 1                          |
|             |             | 1         | 0        | 0         | Output disabled<br>(no driven by the timer)<br>CxOUT = CxP, Cx_EN = 0       | Output disabled<br>(no driven by the timer)<br>CxCOUT = CxCP, CxCEN = 0          |
|             |             | 1         | 0        | 1         | Off-state<br>(Output enabled with inactive level)<br>CxOUT = CxP, Cx_EN = 1 | CxORAW + polarity,<br>CxCOUT = CxORAW xor CxCP,<br>CxCEN = 1                     |
|             |             | 1         | 1        | 0         | CxORAW + polarity,<br>CxOUT = CxORAW xor CxP,<br>Cx_EN = 1                  | Off-state<br>(Output enabled with inactive<br>level)<br>CxCOUT = CxCP, CxCEN = 1 |
|             |             | 1         | 1        | 1         | CxORAW + polarity + dead-time,<br>Cx_EN = 1                                 | CxORAW inverted + polarity +<br>dead-time,<br>CxCEN = 1                          |

| Control bit |   |   |   |   | Output state <sup>(1)</sup>  |
|-------------|---|---|---|---|--|
| 0           | 0 | X | 0 | 0 | Output disabled<br>(corresponding IO is not driven by the timer, IO floating)  |
|             | 0 |   | 0 | 1 | Asynchronously: CxOUT = CxP, Cx_EN = 0, CxCOUT = CxCP, CxCEN = 0;  |
|             | 0 |   | 1 | 0 | If the clock is present: after a dead-time, CxOUT = CxIOS, CxCOUT = CxCIOS, assuming that CxIOS and CxCIOS do not correspond to CxOUT and CxCOUT active level. |
|             | 0 |   | 1 | 1 |  |
|             | 1 |   | 0 | 0 | CxEN = CxCEN = 0: (corresponding IO is not driven by the timer, IO floating)   |
|             | 1 |   | 0 | 1 | In other cases, Off-state (Output enabled with inactive level)   |
|             | 1 |   | 1 | 0 | Asynchronously: CxOUT = CxP, Cx_EN = 1, CxCOUT = CxCP, CxCEN = 1;  |
|             | 1 |   | 1 | 1 | If the clock is present: after a dead-time, CxOUT = CxIOS, CxCOUT = CxCIOS, assuming that CxIOS and CxCIOS do not correspond to CxOUT and CxCOUT active level. |

Note: 1. If the two outputs of a channel are not used (CxEN = CxCEN = 0), CxIOS, CxCIOS, CxP and CxCP must be cleared.

2. The state of the external I/O pins connected to the complementary CxOUT and CxCOUT channels depends on the CxOUT and CxCOUT channel state and the GPIO and the IOMUX registers.

#### TMR16 and TMR17 counter value (TMRx\_CVAL)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---------------|
| Bit 15:0 | CVAL | 0x0         | rw   | Counter value |

#### TMR16 and TMR17 division value (TMRx\_DIV)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | DIV  | 0x0         | rw   | Divider value<br>The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK} / (DIV[15:0] + 1)$ .<br>The value of this register is transferred to the actual prescaler register when an overflow event occurs. |

#### TMR16 and TMR17 period register (TMRx\_PR)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | PR   | 0x0         | rw   | Period value<br>This defines the period value of the TMRx counter. The timer stops working when the period value is 0. |

#### TMR16 and TMR17 repetition period register (TMRx\_RPR)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:8 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 7:0  | RPR      | 0x0         | rw   | Repetition of period value<br>This field is used to reduce the generation rate of overflow events. An overflow event is generated when the repetition counter reaches 0. |

### TMR16 and TMR17 channel 1 data register (TMRx\_C1DT)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 15:0 | C1DT | 0x0         | rw   | <p>Channel 1 data register</p> <p>When the channel 1 is configured as input mode:<br/>The C1DT is the CVAL value stored by the last channel 1 input event (C1IN)</p> <p>When the channel 1 is configured as output mode:<br/>C1DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C1OBEN bit, and the corresponding output is generated on C1OUT as configured.</p> |

### TMR16 and TMR17 brake register (TMRx\_BRK)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:17 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 19:16 | BKF      | 0x0         | rw   | <p>Brake input filter</p> <p>This field is used to set the filter for brake input. The filter number N indicates that the input edge can pass through filter only after N sampling events.</p> <p>0000: No filter, sampling done at <math>f_{DTS}</math><br/> 0001: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 2<br/> 0010: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 4<br/> 0011: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 8<br/> 0100: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 6<br/> 0101: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 8<br/> 0110: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 6<br/> 0111: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 8<br/> 1000: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 6<br/> 1001: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 8<br/> 1010: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 5<br/> 1011: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 6<br/> 1100: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 8<br/> 1101: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 5<br/> 1110: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 6<br/> 1111: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 8</p> |
| Bit 15    | OEN      | 0x0         | rw   | <p>Output enable</p> <p>This bit acts on the channels as output. It is used to enable CxOUT and CxCOUT outputs.</p> <p>0: Disabled<br/>1: Enabled</p>  |
| Bit 14    | AOEN     | 0x0         | rw   | <p>Automatic output enable</p> <p>OEN is set automatically at an overflow event.</p> <p>0: Disabled<br/>1: Enabled</p>   |
| Bit 13    | BRKV     | 0x0         | rw   | <p>Brake input validity</p> <p>This bit is used to select the active level of a brake input.</p> <p>0: Brake input is active low.<br/>1: Brake input is active high.</p>   |
| Bit 12    | BRKEN    | 0x0         | rw   | <p>Brake enable</p> <p>This bit is used to enable brake input.</p> <p>0: Brake input is disabled.<br/>1: Brake input is enabled.</p>   |

| Bit      | Name    | Reset value | Type | Description   |
|----------|---------|-------------|------|---|
| Bit 11   | FCSOEN  | 0x0         | rw   | <p>Frozen channel status when holistic output enable</p> <p>This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN = 1.</p> <p>0: CxOUT/CxCOUT outputs are disabled.</p> <p>1: CxOUT/CxCOUT outputs are enabled. Output inactive level.</p>   |
| Bit 10   | FCSODIS | 0x0         | rw   | <p>Frozen channel status when holistic output disable</p> <p>This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN = 0.</p> <p>0: CxOUT/CxCOUT outputs are disabled.</p> <p>1: CxOUT/CxCOUT outputs are enabled. Output idle level.</p>  |
| Bit 9: 8 | WPC     | 0x0         | rw   | <p>Write protection configuration</p> <p>This field is used to enable write protection.</p> <p>00: Write protection is OFF.</p> <p>01: Write protection level 3, and the following bits are write protected:</p> <p>TMRx_BRK: DTC, BRKEN, BRKV and AOEN</p> <p>TMRx_CTRL2: CxIOS and CxCIOS</p> <p>10: Write protection level 2. The following bits and all bits in level 3 are write protected:</p> <p>TMRx_CCTRL: CxP and CxCP</p> <p>TMRx_BRK: FCSODIS and FCSOEN</p> <p>11: Write protection level 1. The following bits and all bits in level 2 are write protected:</p> <p>TMRx_CMx: C2OCTRL and C2OBEN</p> <p>Note: Once WPC &gt; 0, its content remains frozen until the next system reset.</p> |
| Bit 7: 0 | DTC     | 0x00        | rw   | <p>Dead-time configuration</p> <p>This field defines the duration of the dead-time insertion. The 3-bit MSB of DTC[7:0] is used for function selection:</p> <p>0xx: <math>DT = DTC[7:0] \times TDTS</math></p> <p>10x: <math>DT = (64 + DTC[5:0]) \times TDTS \times 2</math></p> <p>110: <math>DT = (32 + DTC[4:0]) \times TDTS \times 8</math></p> <p>111: <math>DT = (32 + DTC[4:0]) \times TDTS \times 16</math></p>  |

Note: Based on lock configuration, BKF, AOEN, BRKV, BRKEN, FCSODIS, FCSOEN and DTC[7:0] can all be write protected. Thus it is necessary to configure write protection when writing to the TMRx\_BRK register for the first time.

### TMR16 and TMR17 DMA control register (TMRx\_DMACTRL)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15:13 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 12:8  | DTB      | 0x0         | rw   | DMA transfer bytes<br>This field defines the number of DMA transfers:<br>00000: 1 byte    00001: 2 bytes<br>00010: 3 bytes    00011: 4 bytes<br>.....<br>10000: 17 bytes    10001: 18 bytes |
| Bit 7:5   | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 4:0   | ADDR     | 0x0         | rw   | DMA transfer address offset<br>ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register:<br>00000: TMRx_CTRL1<br>00001: TMRx_CTRL2<br>00010: TMRx_STCTRL<br>.....  |

### TMR16 and TMR17 DMA data register (TMRx\_DMADT)

| Bit      | Name  | Reset value | Type | Description   |
|----------|-------|-------------|------|---|
| Bit 15:0 | DMADT | 0x0         | rw   | DMA data register<br>A write/read operation to the DMADT register accesses any TMR register located at the following address:<br>TMRx peripheral address + ADDR × 4 to TMRx peripheral address + ADDR × 4 + DTB × 4 |

## Advanced-control timers (TMR1)

### TMR1 introduction

The advanced-control timer TMR1 consists of a 16-bit counter supporting up and down counting modes, four capture/compare registers, and four independent channels to achieve embedded dead-time, input capture and programmable PWM output.

### TMR1 main features

- Source of counter clock: internal clock, external clock an internal trigger input
- 16-bit up, down, up/down, repetition and encoder mode counter
- Five independent channels for input capture, output compare, PWM generation, one-pulse mode output and embedded dead-time
- Three independent channels for complementary output
- TMR brake function
- Synchronization control between master and slave timers
- Interrupt/DMA is generated at overflow event, trigger event, brake signal input and channel event
- Support TMR burst DMA transfer

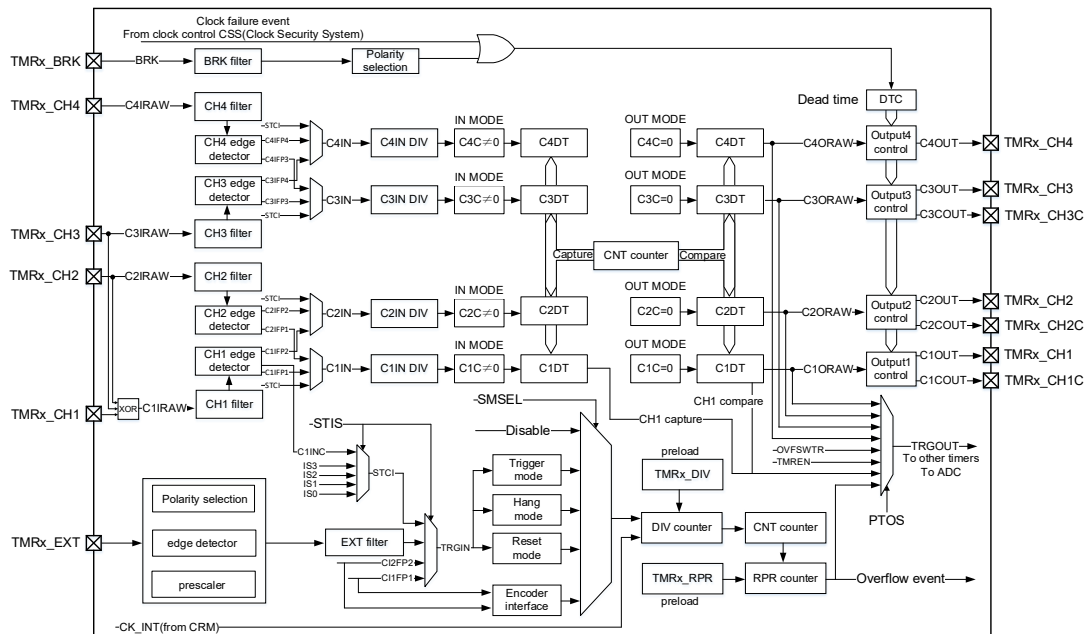


Figure 177. Block diagram of advanced-control timer

## TMR1 function overview

### Counting clock

The count clock of TMR1 can be provided by the internal clock (CK\_INT), external clock (external clock mode A and B) and internal trigger input (ISx).

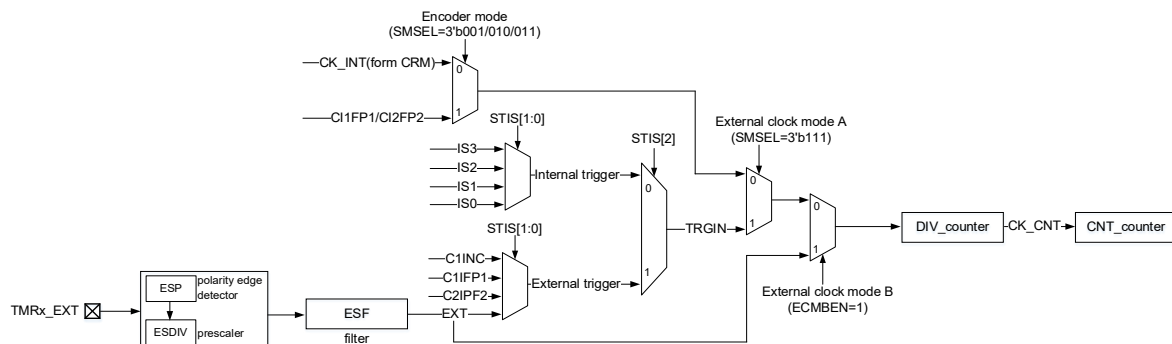


Figure 178. Counting clock

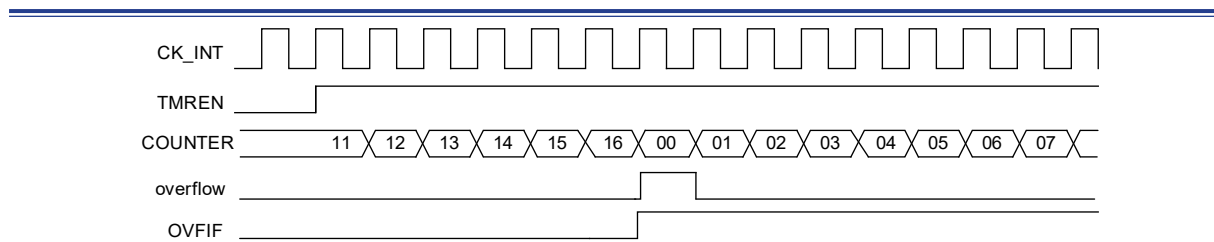
### Internal clock (CK\_INT)

By default, the CK\_INT which is divided by the prescaler, is used to drive the counter to start counting. When TMR's APB clock prescaler factor is 1, the CK\_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

Follow the configuration steps below:



- Set CK\_INT frequency by setting the CLKDIV[1:0] in TMRx\_CTRL1 register.
- Select a counting mode by setting the TWCMSSEL[1:0] in TMRx\_CTRL1 register. If an unidirectional aligned counting mode is selected, it is necessary to select a counting direction through the OWCDIR in TMRx\_CTRL1 register.
- Set counting frequency through TMRx\_DIV register.
- Set counting cycles through TMRx\_PR register.
- Enable a counter by setting the TMREN bit in the TMRx\_CTRL1 register.



**Figure 179. Control circuit with CK\_INT, TMRx\_DIV = 0x0 and TMRx\_PR = 0x16**

### External clock (TRGIN/EXT)

The counter clock can be provided by two external clock sources, namely, TRGIN and EXT signals.

SMSSEL = 3'b111: External clock mode A is selected. By setting the STIS[2:0] bit, select an external clock source TRGIN signal to drive the counter to start counting.

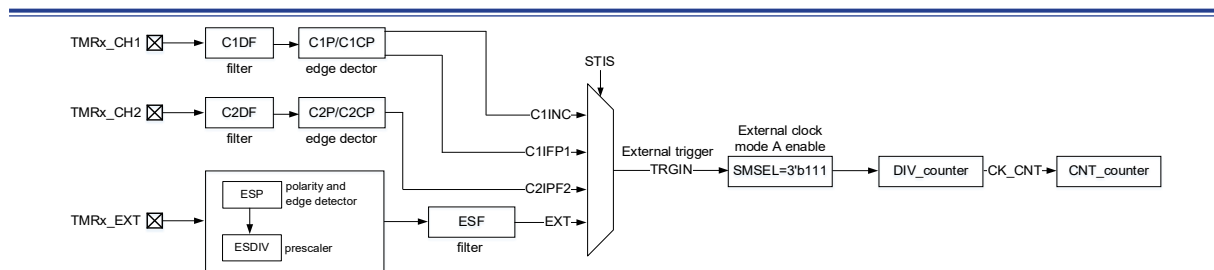
The external clock sources include: C1INC (STIS = 3'b100, channel 1 rising edge and falling edge), C1IFP1 (STIS = 3'b101, a signal after channel 1 filter and polarity selection), C2IFP2 (STIS = 3'b110, a signal after channel 2 filter and polarity selection) and EXT (STIS = 3'b111, external input signal after polarity selection, frequency division and filter).

ECMBEN = 1: External clock mode B is selected. The counter is driven by external input that has gone through polarity selection, frequency division and filtering. The external clock mode B is equivalent to the external clock mode A, and the EXT signal is used as an external force TRGIN,

To use external clock mode A, follow the steps below:

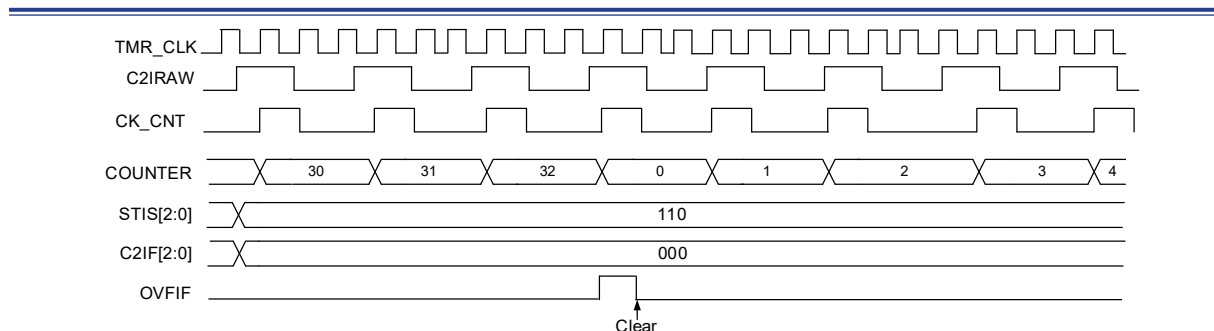
- Set external source TRGIN parameters
  - If the TMRx\_CH1 is used as a source of TRGIN, it is necessary to configure channel 1 input filter (C1DF[3:0] in TMRx\_CM1 register) and channel 1 input polarity (C1P/C1CP in TMRx\_CCTRL register);
  - If the TMRx\_CH2 is used as source of TRGIN, it is necessary to configure channel 1 input filter (C2DF[3:0] in TMRx\_CM1 register) and channel 2 input polarity (C2P/C2CP in TMRx\_CCTR register);
  - If the TMRx\_EXT is used as a source of TRGIN, it is necessary to configure the external signal polarity (ESP in TMRx\_STCTRL register), external signal frequency division (ESDIV[1:0] in TMRx\_STCTRL) and external signal filter (ESF[3:0] in TMRx\_STCTRL register).
- Set TRGIN signal source through the STIS[1:0] bit in TMRx\_STCTRL register.
- Enable external clock mode A by setting SMSSEL = 3'b111 in TMRx\_STCTR register.
- Set counting frequency through the DIV[15:0] in TMRx\_DIV register.

- Set counting period through the PR[15:0] in TMRx\_PR register.
  - Enable counter through the TMREN bit in TMRx\_CTRL1 register.
- To use external clock mode B, follow the steps below:
- Set external signal polarity through the ESP bit in TMRx\_STCTRL register.
  - Set external signal frequency division through the ESDIV[1:0] bit in TMRx\_STCTRL register.
  - Set external signal filter through the ESF[3:0] bit in TMRx\_STCTRL register.
  - Enable external clock mode B through the ECMBEN bit in TMRx\_STCTR register.
  - Set counting frequency through the DIV[15:0] bit in TMRx\_DIV register.
  - Set counting period through the PR[15:0] bit in TMRx\_PR register.
  - Enable counter through the TMREN in TMRx\_CTRL1 register.

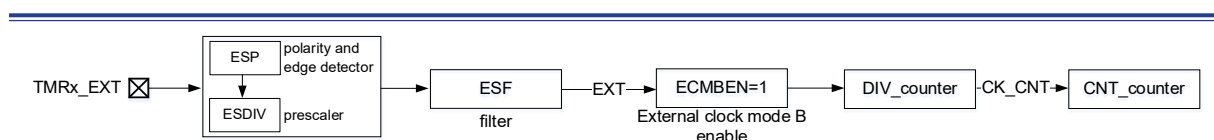


**Figure 180. Block diagram of external clock mode A**

Note: The delay between the signal on the input side and the actual clock of the counter is due to the synchronization circuit.

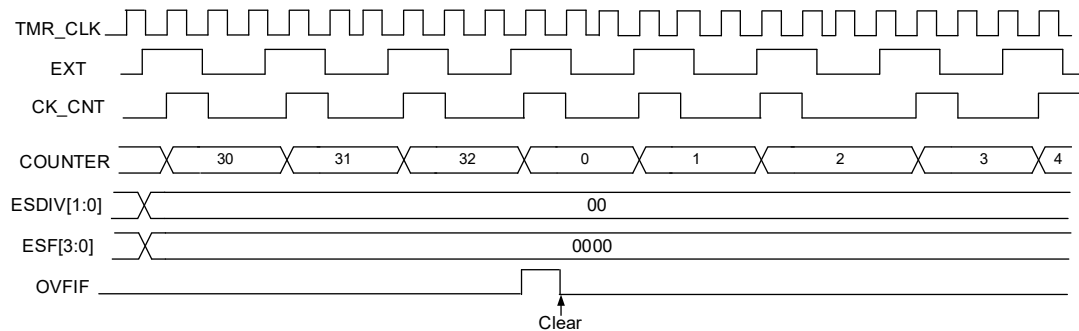


**Figure 181. Counting in external clock mode A, PR = 0x32, DIV = 0x0**



**Figure 182. Block diagram of external clock mode B**

Note: The delay between the EXT signal on the input side and the actual clock of the counter is due to the synchronization circuit.



**Figure 183. Counting in external clock mode B, PR = 0x32, DIV = 0x0**

### Internal trigger input (ISx)

Timer synchronization allows interconnection between several timers. The TMR\_CLK of one timer can be provided by the TRGOUT signal output by another timer. Set the STIS[2:0] bit to select internal trigger signal to enable counting.

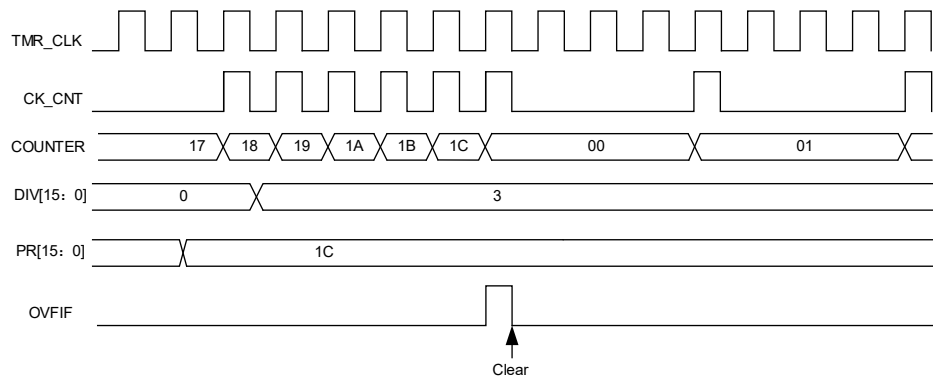
Each timer consists of a 16-bit prescaler, which is used to generate the CK\_CNT that enables the counter to count. The frequency division relationship between the CK\_CNT and TMR\_CLK can be adjusted by setting the value of the TMRx\_DIV register. The prescaler value can be modified at any time, but it takes effect only when the next overflow event occurs.

Below is the configuration procedure for internal trigger input:

- Set counting cycles through TMRx\_PR register.
- Set counting frequency through TMRx\_DIV register.
- Set counting modes through the TWCMSSEL[1:0] in TMRx\_CTRL1 register.
- Select internal trigger by setting STIS[2:0] = 3'b000 ~ 3'b011 in TMRx\_STCTRL register.
- Select external clock mode A by setting SMSEL[2:0] = 3'b111 in TMRx\_STCTRL register.
- Enable TMRx to start counting through the TMREN in TMRx\_CTRL1 register.

**Table 59. TMRx internal trigger connection**

| Slave timer | IS0 (STIS = 000) | IS1 (STIS = 001) | IS2 (STIS = 010) | IS3 (STIS = 011) |
|-------------|------------------|------------------|------------------|------------------|
| TMR1        | TMR15            | TMR2             | TMR3             | —                |
| TMR2        | TMR1             | TMR15            | TMR3             | USB_OTG_SOF      |
| TMR3        | TMR1             | TMR2             | TMR15            | —                |
| TMR15       | TMR2             | TMR3             | TMR16            | TMR17_OC         |



**Figure 184. Counter timing with prescaler value changing from 1 to 4**

### Counting mode

The advanced-control timer consists of a 16-bit counter supporting up, down, up/down counting modes.

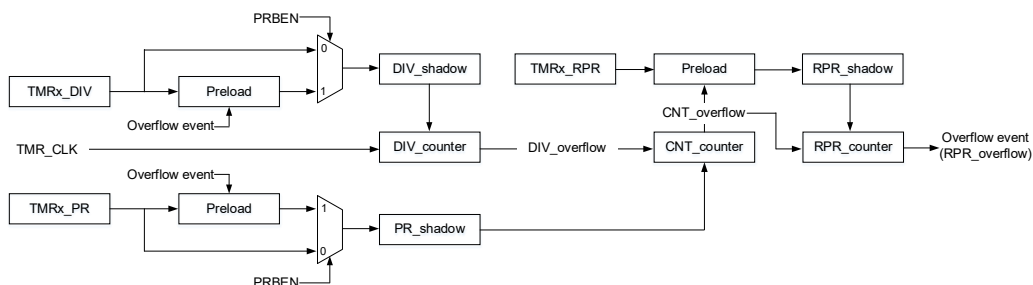
The TMRx\_PR register is used to define counting period of counter. The value in the TMRx\_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN = 1), the value in the TMRx\_PR register is transferred to the shadow register only at an overflow event.

TMRx\_DIV register is used to define the counter frequency of the counter. The counter counts once every DIV[15:0] + 1 clock cycle. Similar to TMRx\_PR register, after enabling periodic buffer, the value of the TMRx\_DIV register are transferred into the shadow register at each overflow event.

Reading the TMRx\_CNT register returns the current counter value. Writing the TMRx\_CNT register will update the current counter value.

An overflow event is enabled by default. It can be disabled by setting OVFEN = 1 in the TMRx\_CTRL1 register. The OVFS bit in the TMRx\_CTRL1 register is used to select the source of an overflow event, which is, by default, counter overflow or underflow, setting OVFSWTR, reset signal generated by slave mode timer controller in reset mode. Once the OVFS is set, an overflow event is generated only when overflow or underflow occurs.

Setting the TMREN bit (TMREN = 1) enables the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR\_EN is set 1 clock cycle after the TMREN is set.

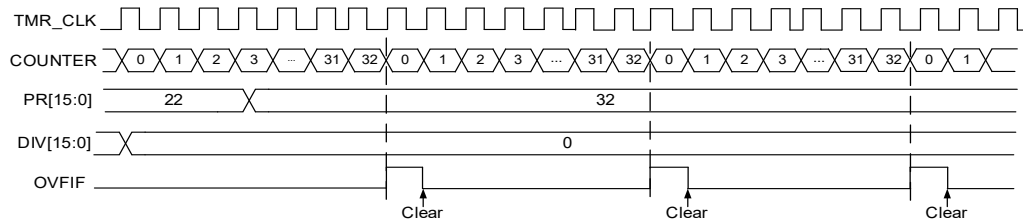


**Figure 185. Basic structure of a counter**

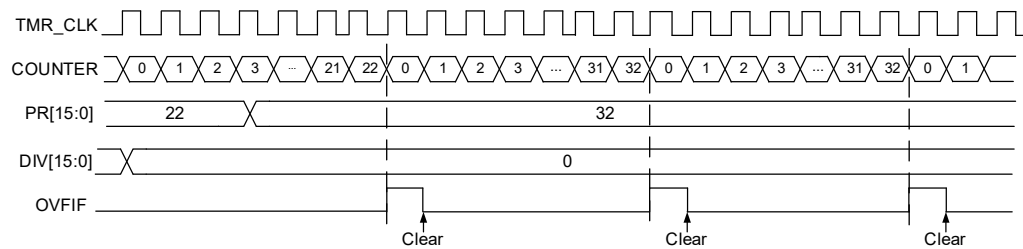
### Upcounting mode

This mode is enabled by setting  $TWCMSEL[1:0] = 2'b00$  and  $OWCDIR = 1'b0$  in the  $TMRx\_CTRL1$  register.

In upcounting mode, the counter counts from 0 to the value programmed in the  $TMR1\_PR$  register, restarts from 0, and generates a counter overflow event, with setting  $OVFIF$  bit to 1. If the overflow event is disabled, the counter is no longer reloaded with the prescaler and re-loaded value on counter overflow; otherwise, the prescaler and re-loaded value will be updated on an overflow event.



**Figure 186. Overflow event when  $PRBEN = 0$**

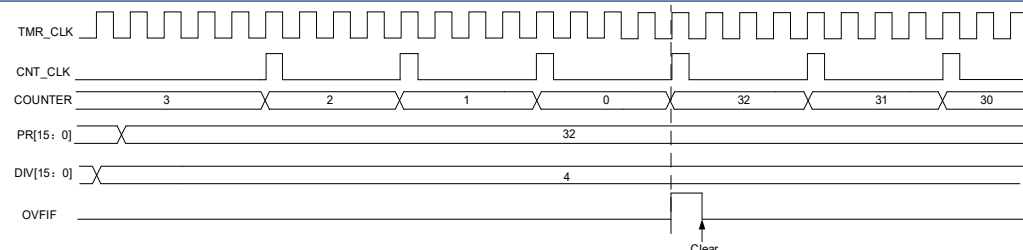


**Figure 187. Overflow event when  $PRBEN = 1$**

### Downcounting mode

This mode is enabled by setting  $TWCMSEL[1:0] = 2'b00$  and  $OWCDIR = 1'b1$  in the  $TMRx\_CTRL1$  register.

In downcounting mode, the counter counts from the value programmed in the  $TMRx\_PR$  register down to 0, and restarts from the value programmed in the  $TMRx\_PR$  register, and generates a counter underflow event.



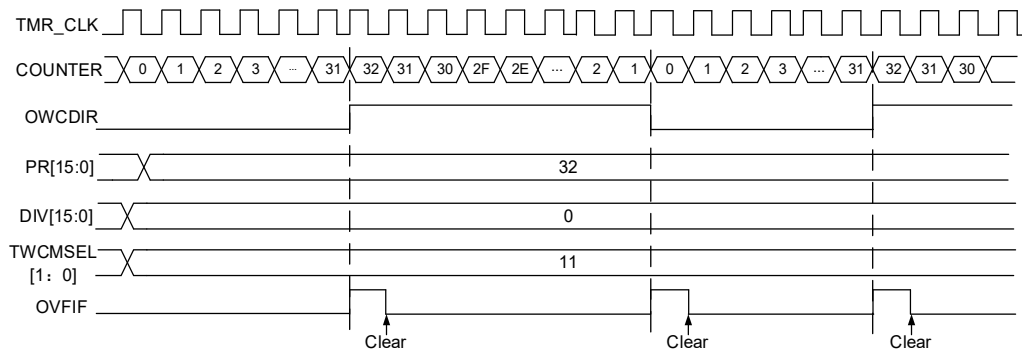
**Figure 188. Counter timing diagram with internal clock divided by 4**

### Up/down counting mode

Up/down counting mode can be enabled by setting  $TWCMSEL[1:0] \neq 2'b00$  in the  $TMRx\_CTRL1$  register. In up/down counting mode, the counter counts up/down alternatively. When the counter counts from the value programmed in the  $TMRx\_PR$  register down to 1, an underflow event is generated, and then restarts counting from 0; When the counter counts from 0 to the value of the  $TMRx\_PR$  register -1, an overflow event is generated, and then restarts counting from the value of the  $TMRx\_PR$  register. The  $OWCDIR$  bit indicates the current counting direction.

The  $TWCMSEL[1:0]$  bit in the  $TMRx\_CTRL1$  register is used to select the condition under which the  $CxIF$  flag is set in two-way counting mode. In other words, when  $TWCMSEL[1:0] = 2'b01$  (counting mode 1) is selected, the  $CxIF$  flag is set only when the counter counts down; when  $TWCMSEL[1:0] = 2'b10$  (counting mode 2) is selected, the  $CxIF$  flag is set only when the counter counts up; when  $TWCMSEL[1:0] = 2'b11$  (counting mode 3) is selected, the  $CxIF$  flag is set when the counter counts up and down.

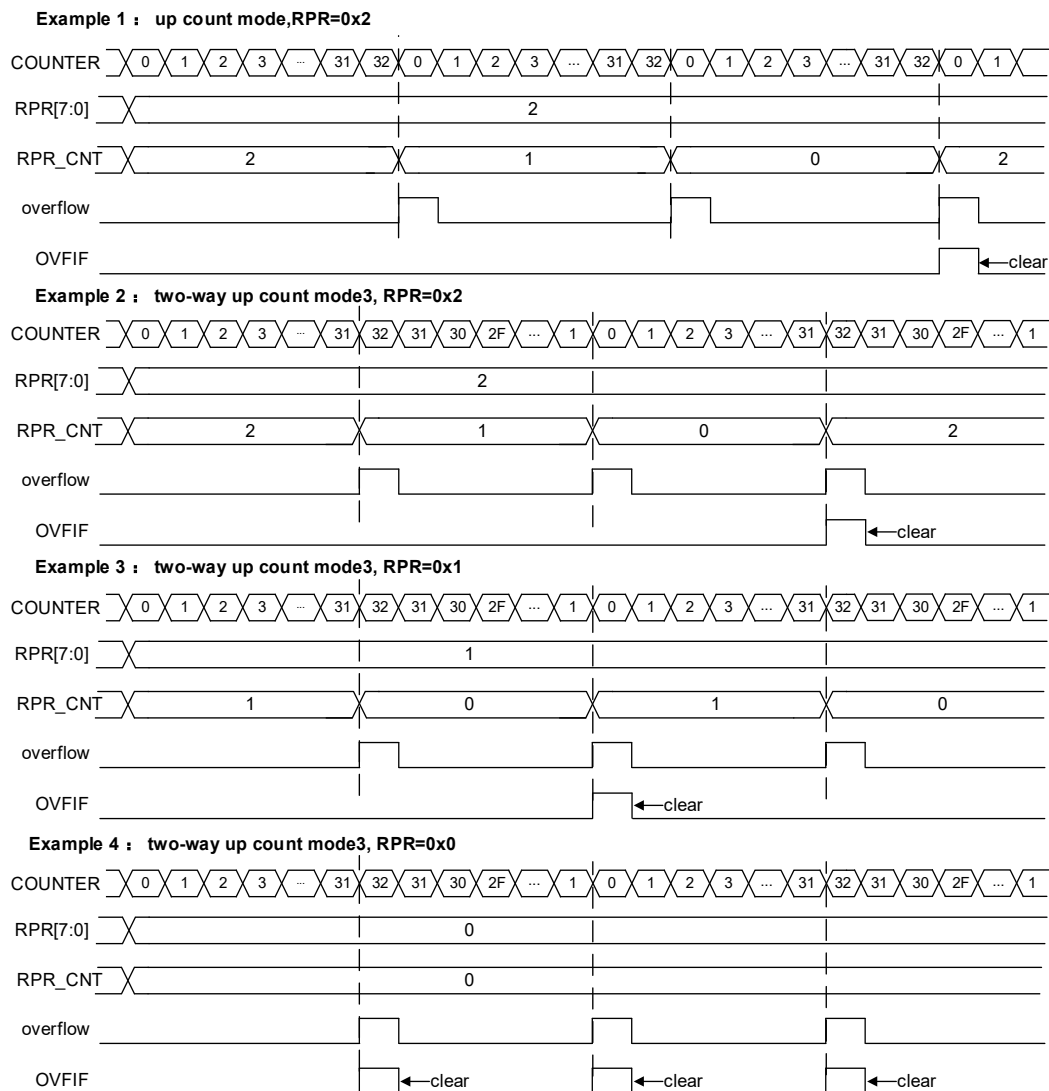
Note: The  $OWCDIR$  is ready-only in up/down counting mode.



**Figure 189. Counter timing diagram with internal clock divided by 1 and  $TMRx\_PR = 0x32$**

### Repetition counter mode:

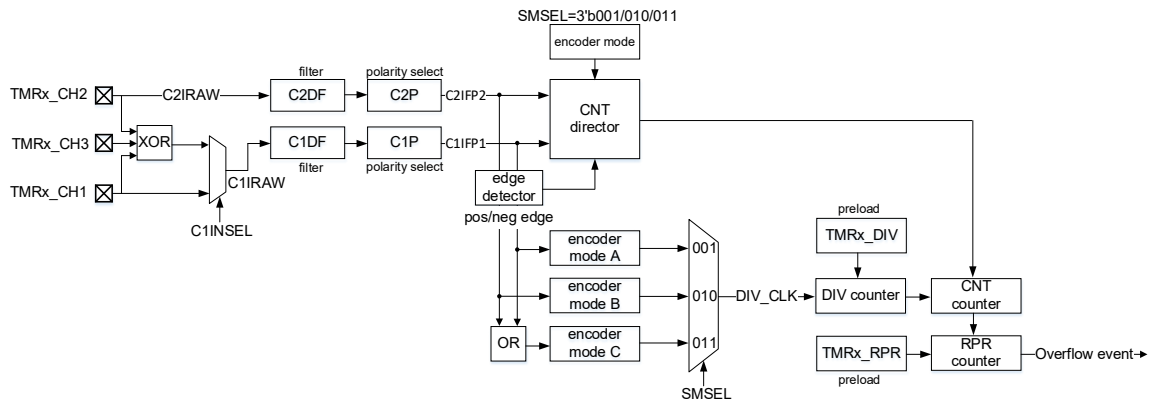
The  $TMRx\_RPR$  register is used to set repetition counting mode. This mode is enabled when the repetition counter value is not equal to 0. In this mode, an overflow event is generated when a counter overflow occurs ( $RPR[7:0] + 1$ ). The repetition counter is decremented at each counter overflow. An overflow event is generated when the repetition counter reaches 0. The frequency of the overflow event can be adjusted by setting the repetition counter value.



**Figure 190. OVIF when RPR = 2**

### Encoder interface mode

To enable the encoder interface mode, write  $SMSEL[2:0] = 3'b001/3'b010/3'b011$ . In this mode, the two inputs (TMRx\_CH1/TMRx\_CH2) are required. Depending on the level on one input, the counter counts up or down on the edge of the other input. The OWCDIR bit indicates the direction of the counter, as shown in the table below:



**Figure 191. Structure of encoder mode**

Encoder mode A:  $SMSEL = 3'b001$ . The counter counts on the selected C1IFP1 edge (rising and falling edges), and the counting direction is dependent on the edge direction of C1IFP1 and the level of C2IFP2.

Encoder mode B:  $SMSEL = 3'b010$ . The counter counts on the selected C2IFP2 edge (rising and falling edges), and the counting direction is dependent on the edge direction of C2IFP2 and the level of C1IFP1.

Encoder mode C:  $SMSEL = 3'b011$ . The counter counts on both C1IFP1 and C2IFP2 edges (rising and falling edges). The counting direction is dependent on the C1IFP1 edge direction and C2IFP2 level, and C2IFP2 edge direction and C1IFP1 level.

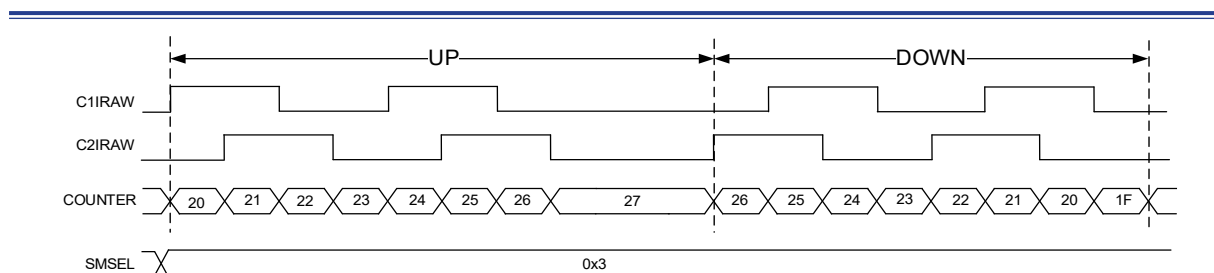
To use encoder mode, follow the procedures below:

- Set channel 1 input signal filtering through the C1DF[3:0] bit in the TMRx\_CM1 register;  
Set channel 1 input signal active level through the C1P bit in the TMRx\_CCTRL register.
- Set channel 2 input signal filtering through the C2DF[3:0] bit in the TMRx\_CM1 register;  
Set channel 2 input signal active level through the C2P bit in the TMRx\_CCTRL register.
- Set channel 1 as input mode through the C1C[1:0] bit in the TMRx\_CM1 register;  
Set channel 2 as input mode through the C2C[1:0] bit in the TMRx\_CM1 register.
- Select encoder mode A ( $SMSEL = 3'b001$ ), encoder mode B ( $SMSEL = 3'b010$ ), or encoder mode C ( $SMSEL = 3'b011$ ) by setting the SMSEL[2:0] bit in the TMRx\_STCTRL register.  
Set counting cycles through the PR[15:0] bit in the TMRx\_PR register.
- Set counting frequency through the DIV[15:0] bit in the TMRx\_DIV register.
- Configure the corresponding IOs of TMRx\_CH1 and TMRx\_CH2 as multiplexed mode.
- Enable counter through the TMREN bit in the TMRx\_CTRL1 register.



**Table 60. Counting direction versus encoder signals**

| Active edge                     | Level on opposite signal<br>(C1IFP1 to C2IFP2,<br>C2IFP2 to C1IFP1) | C1IFP1 signal |          | C2IFP2 signal |          |
|---------------------------------|---|---------------|----------|---------------|----------|
|                                 |   | Rising        | Falling  | Rising        | Falling  |
| Count on C1IFP1 only            | High  | Down          | Up       | No count      | No count |
|                                 | Low   | Up            | Down     | No count      | No count |
| Count on C2IFP2 only            | High  | No count      | No count | Up            | Down     |
|                                 | Low   | No count      | No count | Down          | Up       |
| Count on both C1IFP1 and C2IFP2 | High  | Down          | Up       | Up            | Down     |
|                                 | Low   | Up            | Down     | Down          | Up       |



**Figure 192. Example of encoder interface mode C**

### TMR input function

The TMR1 has four independent channels. Each channel can be configured as input or output. As input, each channel input signal is processed as follows:

- TMRx\_CHx outputs CxIRAW after being preprocessed. Select the TMRx\_CH1 or XOR-ed TMRx\_CH1, TMRx\_CH2, and TMRx\_CH3 for CxIRAW through the C1INSE bit.
- CxIRAW inputs digital filter and outputs filtered CxIF signal. The digital filter uses the CxDF bit to program sampling frequency and sampling times.
- CxIF inputs edge detector, and outputs the CxIFPx signal after edge selection. The edge selection depends on both CxP and CxCP bits. It is possible to select input rising edge, falling edge or both edges.
- CxIFPx inputs capture signal selector, and outputs the CxIN signal after capture signal selection. The capture signal selection is defined by CxC bits. It is possible to select CxIFPx, CyIFPx or STCI as CxIN source. Of those, CyIFPx (x≠y) is the CyIFPy signal that is from Y channel and processed by channel-x edge detector. The STCI comes from slave timer controller, and its source is selected by STIS bit.
- CxIN outputs the CxIPS signal that is divided by input channel divider. The divider factor can be defined as No division, /2, /4 or /8, by the CxIDIV bit.

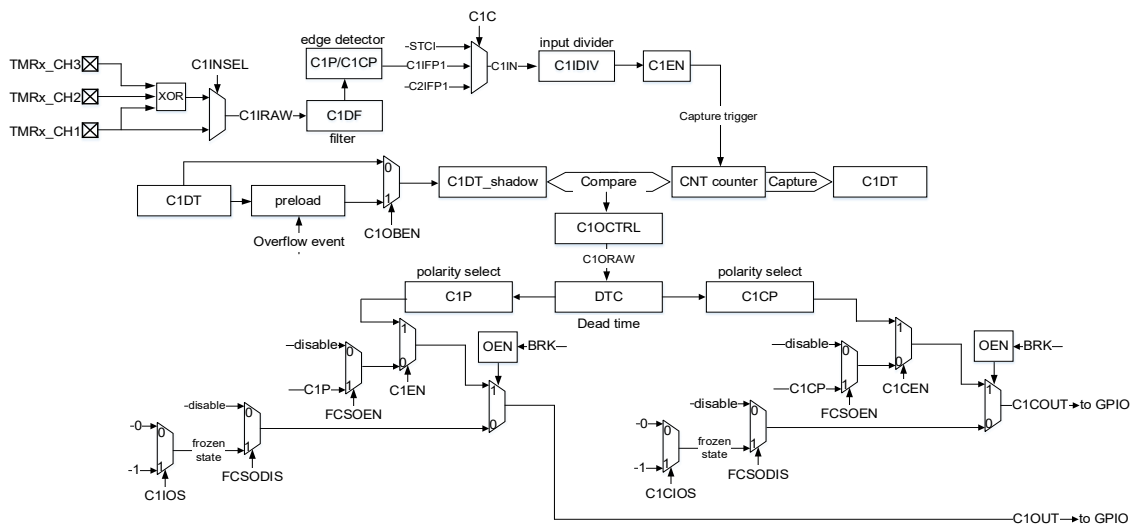


Figure 193. Input/output channel 1 main circuit

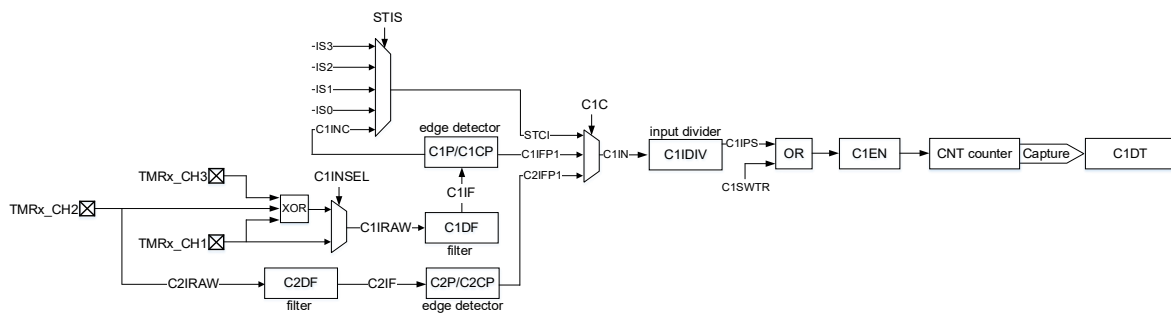


Figure 194. Channel 1 input stage

### Input mode

In input mode, the TMRx\_CxDT registers latch the current counter values after the selected trigger signal is detected, and the capture compare interrupt flag bit (CxIF) is set to 1. An interrupt/DMA request will be generated if the CxIEN bit and CxDEN bit are enabled. If the selected trigger signal is detected when the CxIF is set to 1, a capture overflow event is generated. The previous counter value will be overwritten with the current counter value, and the CxRF is set to 1

To capture the rising edge of C1IN input, following the procedure below:

- Set C1C = 01 in the TMRx\_CM1 register to select the C1IN as channel 1 input.
- Set C1IN signal filter bandwidth (CxDF[3:0]).
- Set the active edge on the C1IN channel by writing C1P = 0 (rising edge) in the TMRx\_CCTR register.
- Program C1IN signal capture frequency divider (C1DIV[1:0])
- Enable channel 1 input capture (C1EN = 1)

- If needed, enable the relevant interrupt or DMA request by setting the C1IEN bit in the TMRx\_IDEN register or the C1DEN bit in the TMRx\_IDEN register.

### Timer Input XOR function

The timer input pins (TMRx\_CH1, TMRx\_CH2 and TMRx\_CH3) are connected to the channel 1 (selected by setting the C1INSE in the TMRx\_CTRL2 register) through an XOR gate.

The XOR gate can be used to connect Hall sensors. For example, connect the three XOR inputs to the three Hall sensors respectively so as to calculate the position and speed of the rotation by analyzing three Hall sensor signals.

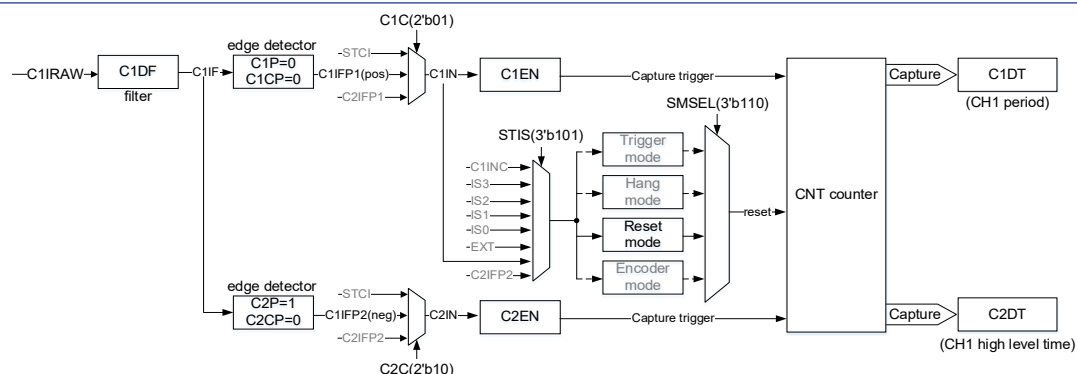
### PWM input

PWM input mode is applied to channel 1 and 2. To use this mode, both C1IN and C2IN are mapped on the same TMRx\_CHx, and the CxIFPx of either channel 1 or channel 2 must be configured as trigger input and slave mode controller is configured in reset mode.

The PWM input mode can be used to measure the period and duty cycle of the PWM input signal. For example, the user can measure the period and duty cycle of the PWM applied on channel 1 using the following procedures:

- Set C1C = 2'b01: select C1IN for C1IFP1.
- Set C1P = 1'b0, select C1IFP1 rising edge active.
- Set C2C = 2'b10, select C2IN for C1IFP2.
- Set C2P = 1'b1, select C1IFP2 falling edge active.
- Set STIS = 3'b101, select the slave mode timer trigger signal as C1IFP1.
- Set SMSEL = 3'b100: configure the slave mode controller in reset mode.
- Set C1EN = 1'b1 and C2EN = 1'b1. Enable channel 1 and input capture.

After above configuration, the rising edge of channel 1 input signal will trigger the capture and stores the capture value into C1DT register, and it will reset the counter at the same time. The falling edge of the channel 1 input signal triggers the capture and stores the capture value into C2DT register. The period of the channel 1 input signal is calculated through C1DT, and its duty cycle through C2DT.



**Figure 195. PWM input mode configuration example**

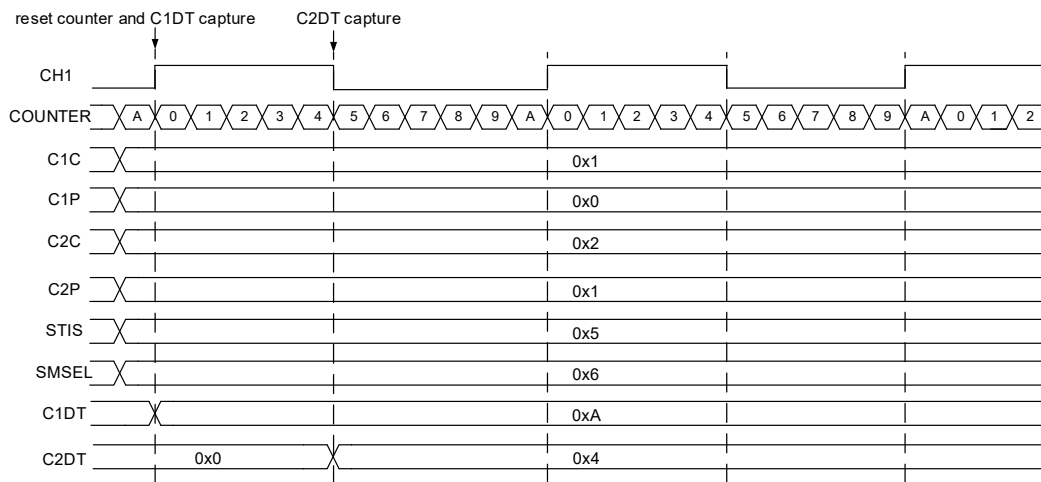


Figure 196. PWM input mode

### TMR output function

The TMR output consists of a comparator and an output controller. It is used to program the period, duty cycle and polarity of the output signal. The advanced-control timer output function varies from one channel to one channel.

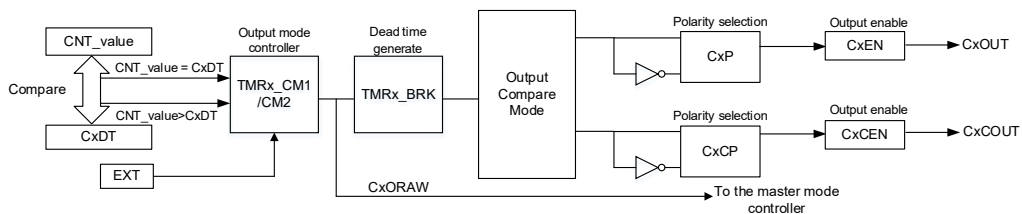


Figure 197. Channel output stage (channel 1 to 3)

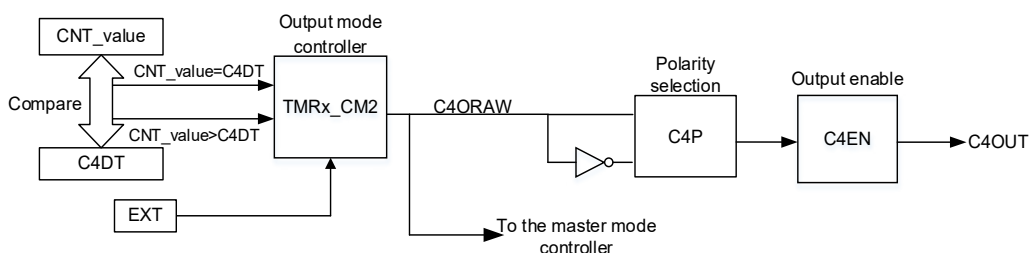


Figure 198. Channel 4 output stage

### Output mode

Write  $CxC[1:0] \neq 2'b00$  to configure the channel as output to implement multiple output modes. In this case, the counter value is compared with the value in the  $TMRx\_CxDt$  register, and the intermediate signal  $CxORAW$  is generated according to the output mode selected by

CxOCTRL[2:0], which is sent to IO after being processed by the output control circuit. The period of the output signal is configured by the TMRx\_PR register, while the duty cycle by the TMRx\_CxDT register.

Output compare modes include:

#### **PWM mode A:**

Enable PWM mode A by setting CxOCTRL = 3'b110. In upcounting mode, C1ORAW outputs high when TMRx\_C1DT > TMRx\_CVAL; otherwise, it is low. In downcounting mode, C1ORAW outputs low when TMRx\_C1DT < TMRx\_CVAL; otherwise, it is high.

To use PWM mode A, the following procedures are recommended:

- Set PWM periods through TMRx\_PR register.
- Set PWM duty cycles through TMRx\_CxD.
- Select PWM mode A by setting CxOCTRL = 3'b110 in the TMRx\_CM1/CM2 register.
- Set counting frequency through TMRx\_DIV register.
- Select counting mode by setting the TWCMSSEL[1:0] bit in the TMRx\_CTRL1 register.
- Select output polarity through the CxP and CxCP bits in the TMRx\_CCTRL register.
- Enable channel output through the CxEN and CxCEN bits in the TMRx\_CCTRL register.
- Enable TMRx output through the OEN bit in the TMRx\_BRK register.
- Configure GPIOs corresponding to TMR output channels as multiplexed mode.
- Enable TMRx to start counting through the TMREN bit in the TMRx\_CTRL1 register.

#### **PWM mode B:**

Enable PWM mode B by setting CxOCTRL = 3'b111. In upcounting mode, C1ORAW outputs low when TMRx\_C1DT > TMRx\_CVAL; otherwise, it is high. In downcounting mode, C1ORAW outputs high when TMRx\_C1DT < TMRx\_CVAL; otherwise, it is low.

#### **Forced output mode:**

Enable forced output mode by setting CxOCTRL = 3'b100/101. In this case, the CxORAW is forced to be the programmed level, regardless of the counter value. Despite this, the channel flag bit and DMA request still depend on the compare result.

#### **Output compare mode:**

Enable output compare mode by setting CxOCTRL = 3'b001/010/011. In this case, when the counter value matches the value of the CxDT register, the CxORAW is forced high (CxOCTRL = 3'b001), low (CxOCTRL = 3'b010) or toggling (CxOCTRL = 3'b011).

#### **One-pulse mode:**

This is a particular case of PWM mode. Enable one-pulse by setting OCMEN = 1. In this mode, the comparison match is performed in the current counting period. The TMREN bit is cleared as soon as the current counting is completed. Therefore, only one pulse is output. When in upcounting mode, the configuration must follow the rule: CVAL < CxDT ≤ PR; in downcounting mode, CVAL > CxDT is required.

#### **Fast output mode:**

Enable this mode by setting CxOIEN = 1. If enabled, the CxORAW signal will not change when the counter value matches the CxDT, but change at the beginning of the current counting period. In

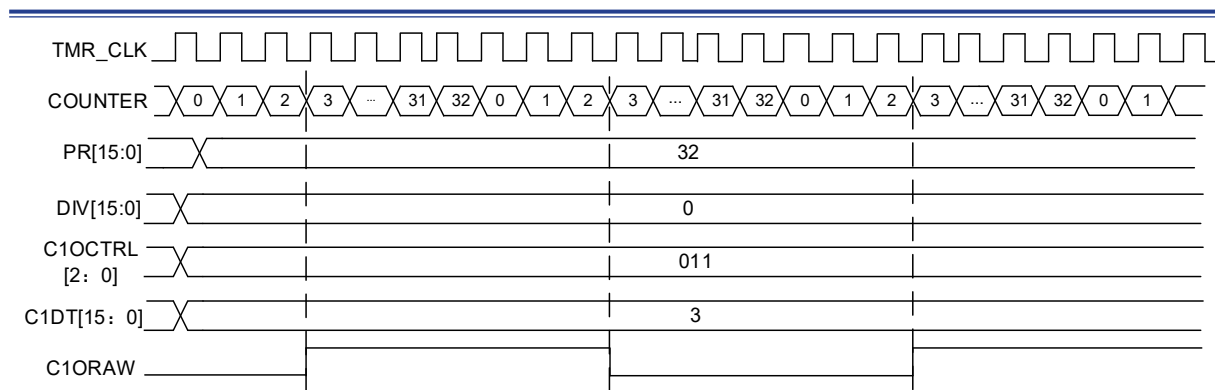
other words, the comparison result is advanced, so the comparison result between the counter value and the TMRx\_CxDt register will determine the level of CxORAW in advance.

Figure 199 gives an example of output compare mode (toggle) with C1DT = 0x3. When the counter value is equal to 0x3, C1OUT toggles.

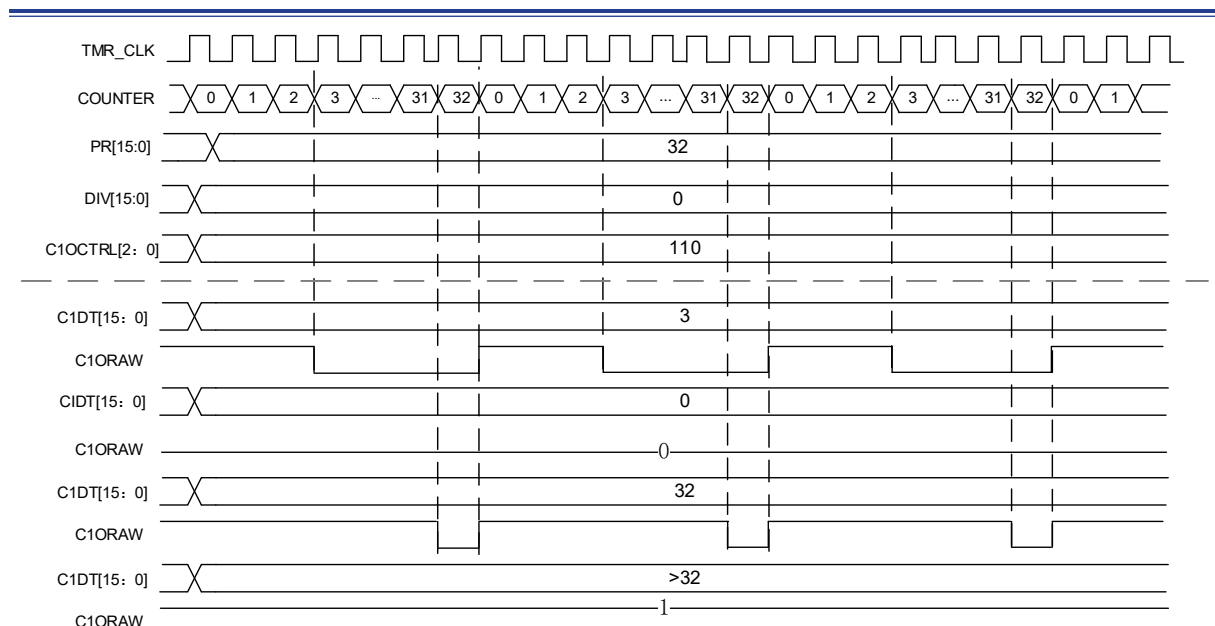
Figure 200 gives an example of the combination between upcounting mode and PWM mode A. The output signal behaves when PR = 0x32 but CxDT is configured with a different value.

Figure 201 gives an example of the combination between up/down counting mode and PWM mode A. The output signal behaves when PR = 0x32 but CxDT is configured with a different value.

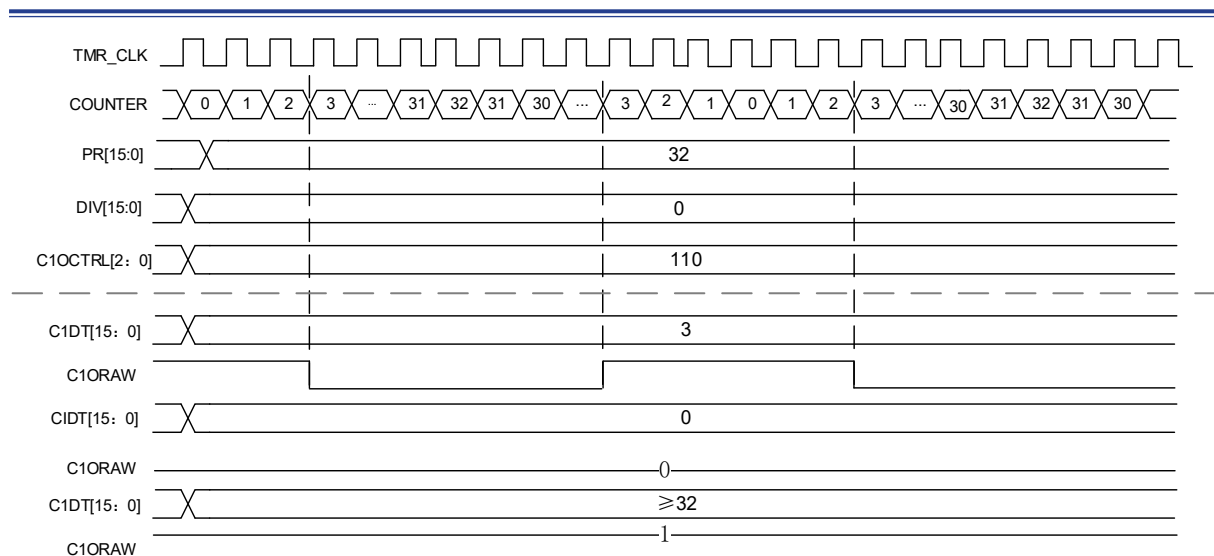
Figure 202 gives an example of the combination between upcounting mode and one-pulse PWM mode B. The counter only counts only one cycle, and the output signal sends only one pulse.



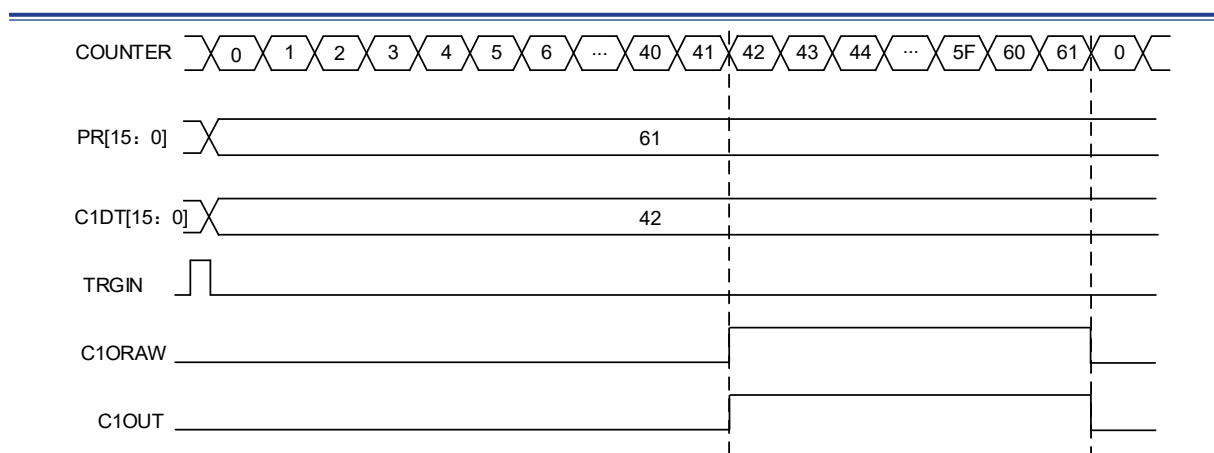
**Figure 199. C1ORAW toggles when counter value matches the C1DT value**



**Figure 200. Upcounting mode and PWM mode A**



**Figure 201. Up/down counting mode and PWM mode**

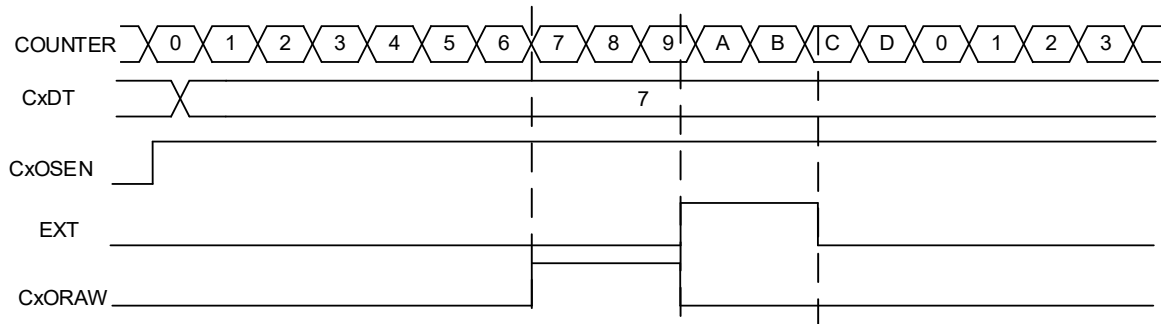


**Figure 202. One-pulse mode**

### CxORAW clear

When the CxOSEN bit is set, the CxORAW signal for a given channel is cleared by applying a high level to the EXT input. The CxORAW signal remains unchanged until the next overflow event.

This function can only be used in output capture or PWM modes, and does not work in forced output mode. Figure 203 shows the example of clearing CxORAW. When the EXT input is high, the CxORAW signal, which was originally high, is driven low; when the EXT is low, the CxORAW signal outputs the corresponding level according to the comparison result between the counter value and CxDT value.



**Figure 203. Clearing CxORAW (PWM mode A) by EXT input**

### Master mode timer event output

When TMR is used as a master timer, one of the following source of signals can be selected as TRGOUT output to a slave mode timer. This is done by setting the PTOS bit in the TMRxCTRL2 register.

- PTOS = 3'b000, TRGOUT output software overflow event (OVFSWTR bit in TMRx\_SWEVT register) or reset event
- PTOS = 3'b001, TRGOUT output counter enable
- PTOS = 3'b010, TRGOUT output counter overflow event
- PTOS = 3'b011, TRGOUT output capture and compare event
- PTOS = 3'b100, TRGOUT output C1ORAW
- PTOS = 3'b101, TRGOUT output C2ORAW
- PTOS = 3'b110, TRGOUT output C3ORAW
- PTOS = 3'b111, TRGOUT output C4ORAW

### Dead-time insertion

The channel 1 to 3 of the advanced-control timers contains a set of reverse channel output. This function is enabled by the CxCEN bit and its polarity is defined by CxCP. Refer to Table 62 for more information about the output state of CxOUT and CxCOUT.

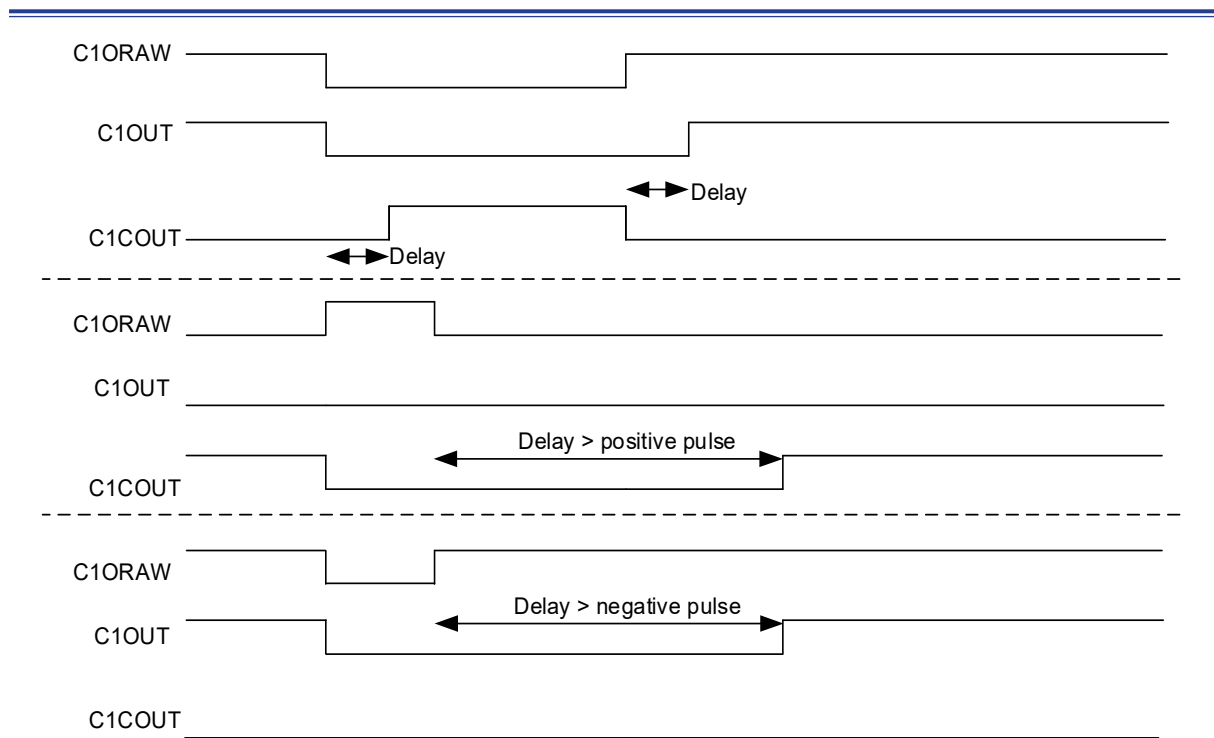
The dead-time is activated when switching to IDLEF state (OEN falling down to 0).

Setting both CxEN and CxCEN bits, and using DTC[7:0] bit to insert dead-time of different durations. After the dead-time insertion, the rising edge of the CxOUT is delayed compared to the rising edge of the reference signal; the rising edge of the CxCOUT is delayed compared to the falling edge of the reference signal.

If the delay is greater than the width of the active output, then the C1OUT and C1COUT will not generate corresponding pulses. Therefore the dead-time should be less than the width of the active output.

Figure 204 gives an example of dead-time insertion when CxP = 0, CxCP = 0, OEN = 1, CxEN = 1 and CxCEN = 1.





**Figure 204. Complementary output with dead-time insertion**

### TMR brake function

When the brake function is enabled ( $BRKEN = 1$ ), the  $CxOUT$  and  $CxCOUT$  are jointly controlled by  $OEN$ ,  $FCSODIS$ ,  $FCSOEN$ ,  $CxIOS$  and  $CxCIOS$ . But,  $CxOUT$  and  $CxCOUT$  cannot be set both to active level at the same time. Please refer to Table 62 for more details.

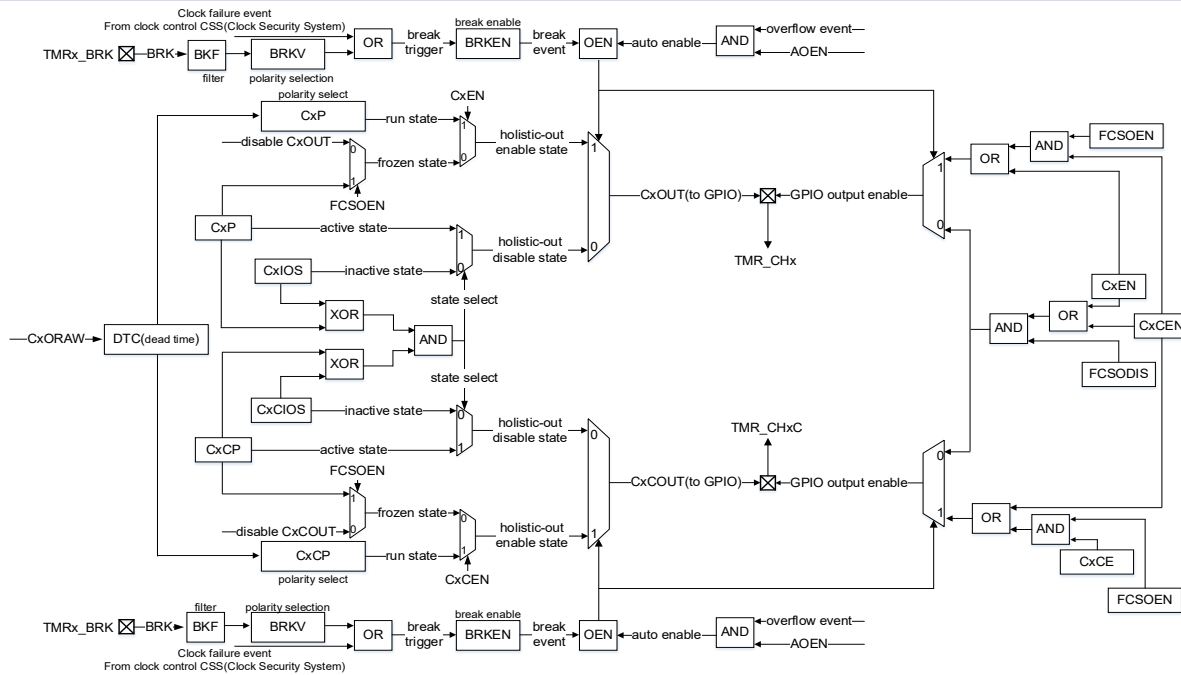
The brake source can be the brake input pin or a clock failure event. The polarity is controlled by the  $BRKV$  bit.

When a brake event occurs, there are the following actions:

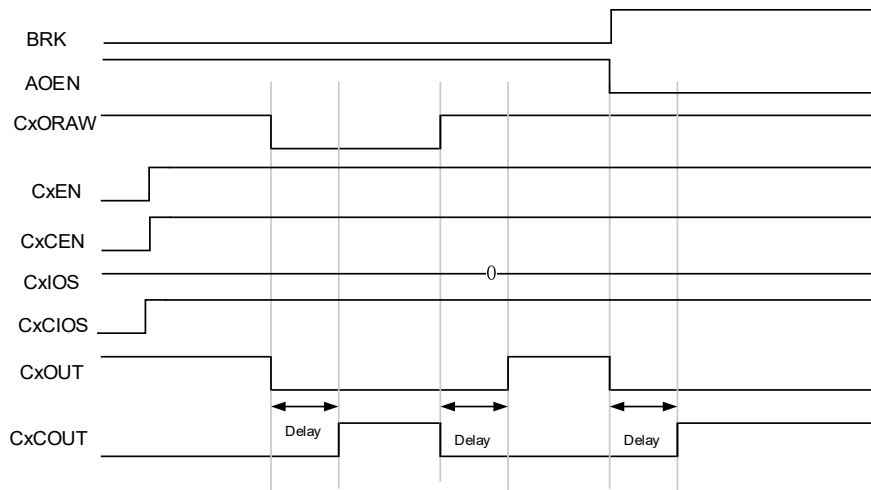
- The  $OEN$  bit is cleared asynchronously, and the channel output state is selected by setting the  $FCSODIS$  bit. This function works even if the MCU oscillator is off.
- Once  $OEN = 0$ , the channel output level is defined by the  $CxIOS$  bit. If  $FCSODIS = 0$ , the timer output is disabled; otherwise, the output enable remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state, that is, 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 active, then the dead-time generator is activated. The  $CxIOS$  and  $CxCIOS$  bits are used to program the level after dead-time. Even in this case, the  $CxIOS$  and  $CxCIOS$  cannot be driven to their active level at the same time. It should be note that because of synchronization on  $OEN$ , the dead-time duration is usually longer than usual (around 2  $clk\_tmr$  clock cycles)
  - If  $FCSODIS = 0$ , the timer releases the enable output; otherwise, it keeps the enable output; the enable output becomes high as soon as one of the  $CxEN$  and  $CxCEN$  bits becomes high.

- If the brake interrupt or DMA request is enabled, the brake statue flag is set, and a brake interrupt or DMA request can be generated.
- If AOEN = 1, the OEN bit is automatically set again at the next overflow event.

Note: When the brake input is active, the OEN cannot be set, nor the status flag, BRKIF can be cleared.



**Figure 205. Example of TMR output control**



**Figure 206. Example of TMR brake function**

### TMR synchronization

The timers are linked together internally for timer synchronization. Master timer is selected by setting the PTOS[2:0] bit; Slave timer is selected by setting the SMSEL[2:0] bit.

Slave modes include:

#### Slave mode: Reset mode

The counter and its prescaler can be reset by a selected trigger signal. An overflow event can be generated when OVFS = 0.

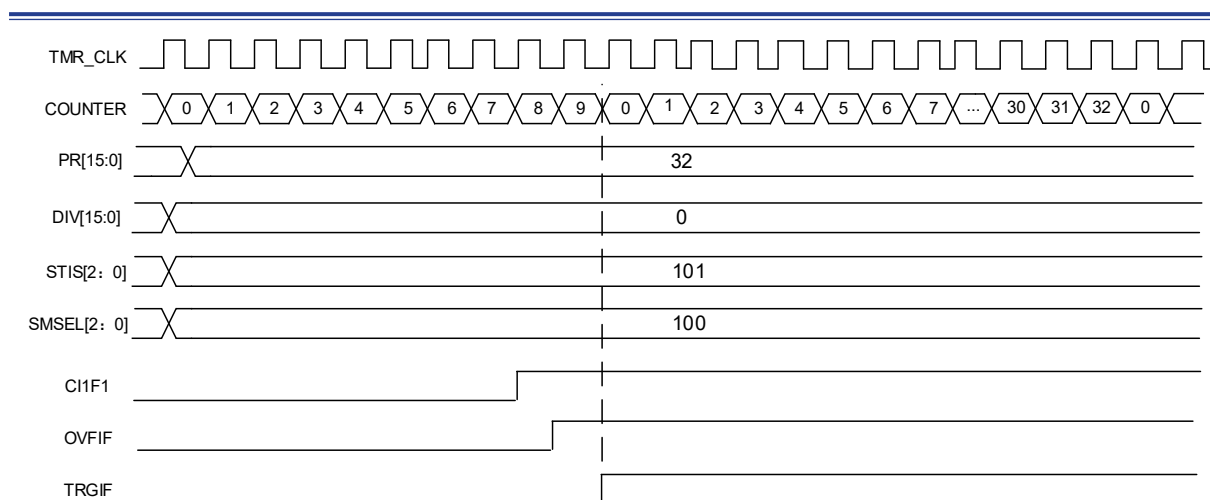


Figure 207. Example of reset mode

#### Slave mode: Suspend mode

In this mode, the counter is controlled by a selected trigger input. The counter starts counting when the trigger input is high and stops as soon as the trigger input is low.

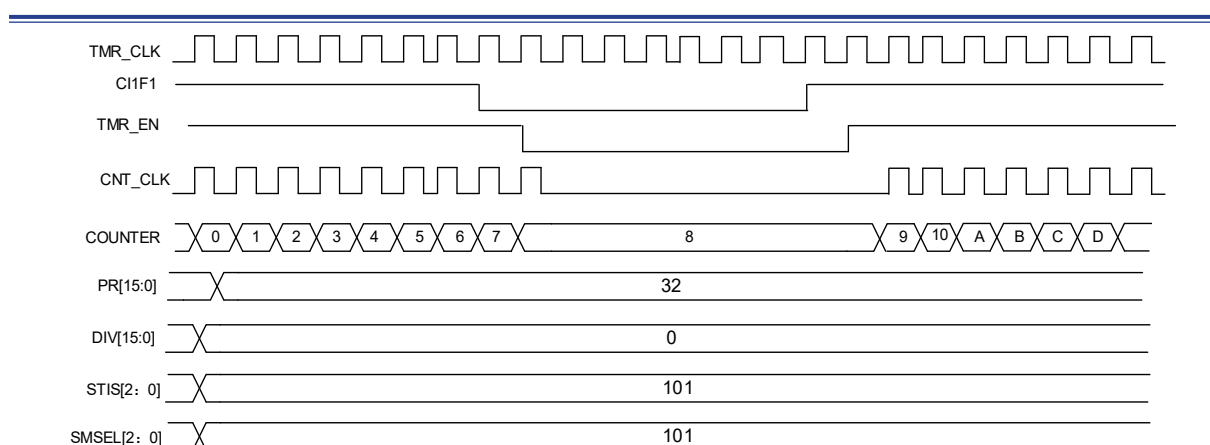
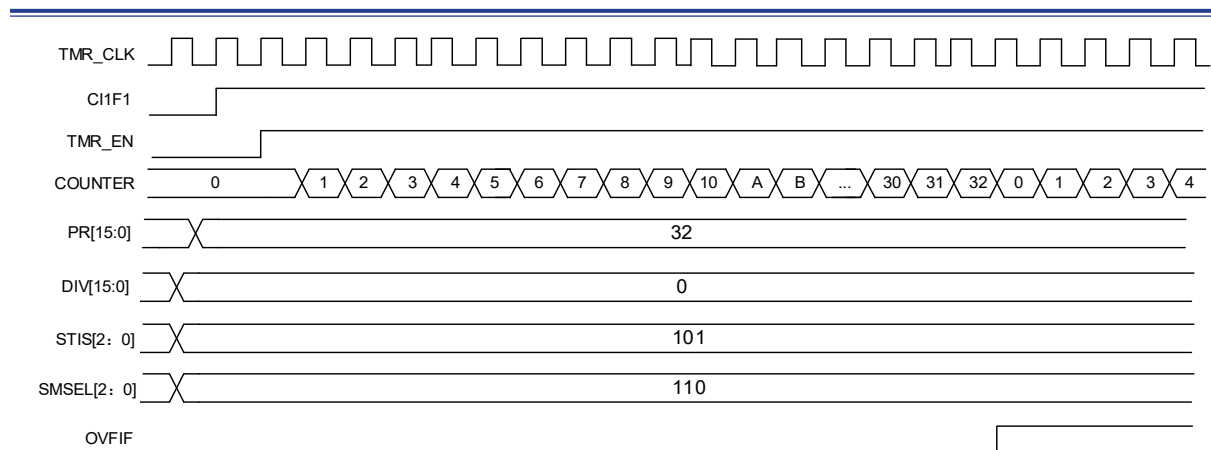


Figure 208. Example of suspend mode

### Slave mode: Trigger mode

The counter can start counting on the rising edge of a selected trigger input (TMR\_EN = 1).



**Figure 209. Example of trigger mode**

Refer to Section 14.2.3.5 for more examples.

### Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the TMRx counter stops counting by setting the TMRx\_PAUSE in the DEBUG module.

### TMR1 registers

These peripheral registers must be accessed by word (32 bits).

TMR1 and TMR8 register are mapped into a 16-bit addressable space.

**Table 61. TMR1 register map and reset value**

| Register    | Offset | Reset value |
|-------------|--------|-------------|
| TMR1_CTRL1  | 0x00   | 0x0000      |
| TMR1_CTRL2  | 0x04   | 0x0000      |
| TMR1_STCTRL | 0x08   | 0x0000      |
| TMR1_IDEN   | 0x0C   | 0x0000      |
| TMR1_ISTS   | 0x10   | 0x0000      |
| TMR1_SWEVT  | 0x14   | 0x0000      |
| TMR1_CM1    | 0x18   | 0x0000      |
| TMR1_CM2    | 0x1C   | 0x0000      |
| TMR1_CCTRL  | 0x20   | 0x0000      |
| TMR1_CVAL   | 0x24   | 0x0000      |
| TMR1_DIV    | 0x28   | 0x0000      |
| TMR1_PR     | 0x2C   | 0x0000      |
| TMR1_RPR    | 0x30   | 0x0000      |
| TMR1_C1DT   | 0x34   | 0x0000      |
| TMR1_C2DT   | 0x38   | 0x0000      |
| TMR1_C3DT   | 0x3C   | 0x0000      |

| Register     | Offset | Reset value |
|--------------|--------|-------------|
| TMR1_C4DT    | 0x40   | 0x0000      |
| TMR1_BRK     | 0x44   | 0x0000      |
| TMR1_DMACTRL | 0x48   | 0x0000      |
| TMR1_DMADT   | 0x4C   | 0x0000      |
| TMR1_CM3     | 0x70   | 0x0000      |
| TMR1_C5DT    | 0x74   | 0x0000      |

#### TMR1 control register 1 (TMR1\_CTRL1)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:10 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 9:8   | CLKDIV   | 0x0         | rw   | <p>Clock division</p> <p>This field is used to define the relationship between digital filter sampling frequency (<math>f_{DTS}</math>) and timer clock frequency (<math>f_{CK\_INT}</math>). it is also used to set the ratio relationship between dead time base (<math>T_{DTS}</math>) and timer clock period (<math>T_{CK\_INT}</math>)</p> <p>00: No division, <math>f_{DTS} = f_{CK\_INT}</math><br/> 01: Divided by 2, <math>f_{DTS} = f_{CK\_INT}/2</math><br/> 10: Divided by 4, <math>f_{DTS} = f_{CK\_INT}/4</math><br/> 11: Reserved</p> |
| Bit 7     | PRBEN    | 0x0         | rw   | <p>Period buffer enable</p> <p>0: Period buffer is disabled<br/> 1: Period buffer is enabled</p>   |
| Bit 6:5   | TWCMSEL  | 0x0         | rw   | <p>Two-way counting mode selection</p> <p>00: One-way counting mode, depending on the OWCDIR bit<br/> 01: Two-way counting mode 1, count up and down alternately, the CxIF bit is set only when the counter counts down<br/> 10: Two-way counting mode 2, count up and down alternately, the CxIF bit is set only when the counter counts up<br/> 11: Two-way counting mode 3, count up and down alternately, the CxIF bit is set when the counter counts up / down</p>  |
| Bit 4     | OWCDIR   | 0x0         | rw   | <p>One-way count direction</p> <p>0: Up<br/> 1: Down</p>   |
| Bit 3     | OCMEN    | 0x0         | rw   | <p>One cycle mode enable</p> <p>This bit is use to select whether to stop counting at an update event</p> <p>0: The counter does not stop at an update event<br/> 1: The counter stops at an update event</p>  |
| Bit 2     | OVFS     | 0x0         | rw   | <p>Overflow event source</p> <p>This bit is used to select overflow event or DMA request sources.</p> <p>0: Counter overflow, setting the OVFSWTR bit or overflow event generated by slave timer controller<br/> 1: Only counter overflow generates an overflow event</p>  |
| Bit 1     | OVFEN    | 0x0         | rw   | <p>Overflow event enable</p> <p>0: Enabled<br/> 1: Disabled</p>  |

| Bit   | Name  | Reset value | Type | Description                             |
|-------|-------|-------------|------|---|
| Bit 0 | TMREN | 0x0         | rw   | TMR enable<br>0: Disabled<br>1: Enabled |

#### TMR1 control register2 (TMR1\_CTRL2)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:15 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 14    | C4IOS    | 0x0         | rw   | Channel 4 idle output state   |
| Bit 13    | C3CIOS   | 0x0         | rw   | Channel 3 complementary idle output state   |
| Bit 12    | C3IOS    | 0x0         | rw   | Channel 3 idle output state   |
| Bit 11    | C2CIOS   | 0x0         | rw   | Channel 2 complementary idle output state   |
| Bit 10    | C2IOS    | 0x0         | rw   | Channel 2 idle output state   |
| Bit 9     | C1CIOS   | 0x0         | rw   | Channel 1 complementary idle output state<br>OEN = 0 after dead-time:<br>0: C1OUTL = 0<br>1: C1OUTL = 1   |
| Bit 8     | C1IOS    | 0x0         | rw   | Channel 1 idle output state<br>OEN = 0 after dead-time:<br>0: C1OUT = 0<br>1: C1OUT = 1   |
| Bit 7     | C1INSEL  | 0x0         | rw   | C1IN selection<br>0: CH1 pin is connected to C1IRAW input<br>1: The XOR result of CH1, CH2 and CH3 pins is connected to C1IRAW input  |
| Bit 6: 4  | PTOS     | 0x0         | rw   | Master TMR output selection<br>This field is used to select the TMRx signal sent to the slave timer.<br>000: Software overflow or Reset<br>001: Enable<br>010: Overflow<br>011: Compare pulse<br>100: C1ORAW signal<br>101: C2ORAW signal<br>110: C3ORAW signal<br>111: C4ORAW signal       |
| Bit 3     | DRS      | 0x0         | rw   | DMA request source<br>0: Capture/compare event<br>1: Overflow event   |
| Bit 2     | CCFS     | 0x0         | rw   | Channel control bit flash selection<br>This bit only acts on channels that have complementary output.<br>If the channel control bits are buffered:<br>0: Control bits are updated by setting the HALL bit<br>1: Control bits are updated by setting the HALL bit or a rising edge on TRGIN. |
| Bit 1     | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 0     | CBCTRL   | 0x0         | rw   | Channel buffer control<br>This bit acts on channels that have complementary output.<br>0: CxEN, CxCEN and CxOCTRL bits are not buffered.<br>1: CxEN, CxCEN and CxOCTRL bits are buffered.   |

### TMR1 slave timer control register (TMR1\_STCTRL)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15    | ESP      | 0x0         | rw   | External signal polarity<br>0: High or rising edge<br>1: Low or falling edge   |
| Bit 14    | ECMBEN   | 0x0         | rw   | External clock mode B enable<br>This bit is used to enable external clock mode B<br>0: Disabled<br>1: Enabled  |
| Bit 13:12 | ESDIV    | 0x0         | rw   | External signal divide<br>This field is used to select the frequency division of an external trigger<br>00: Normal<br>01: Divided by 2<br>10: Divided by 4<br>11: Divided by 8   |
| Bit 11:8  | ESF      | 0x0         | rw   | External signal filter<br>This field is used to filter an external signal. The external signal can be sampled only after it has been generated N times<br>0000: No filter, sampling by $f_{DTS}$<br>0001: $f_{SAMPLING} = f_{CK\_INT}$ , $N = 2$<br>0010: $f_{SAMPLING} = f_{CK\_INT}$ , $N = 4$<br>0011: $f_{SAMPLING} = f_{CK\_INT}$ , $N = 8$<br>0100: $f_{SAMPLING} = f_{DTS}/2$ , $N = 6$<br>0101: $f_{SAMPLING} = f_{DTS}/2$ , $N = 8$<br>0110: $f_{SAMPLING} = f_{DTS}/4$ , $N = 6$<br>0111: $f_{SAMPLING} = f_{DTS}/4$ , $N = 8$<br>1000: $f_{SAMPLING} = f_{DTS}/8$ , $N = 6$<br>1001: $f_{SAMPLING} = f_{DTS}/8$ , $N = 8$<br>1010: $f_{SAMPLING} = f_{DTS}/16$ , $N = 5$<br>1011: $f_{SAMPLING} = f_{DTS}/16$ , $N = 6$<br>1100: $f_{SAMPLING} = f_{DTS}/16$ , $N = 8$<br>1101: $f_{SAMPLING} = f_{DTS}/32$ , $N = 5$<br>1110: $f_{SAMPLING} = f_{DTS}/32$ , $N = 6$<br>1111: $f_{SAMPLING} = f_{DTS}/32$ , $N = 8$ |
| Bit 7     | STS      | 0x0         | rw   | Subordinate TMR synchronization<br>If enabled, master and slave timer can be synchronized.<br>0: Disabled<br>1: Enabled  |
| Bit 6: 4  | STIS     | 0x0         | rw   | Subordinate TMR input selection<br>This field is used to select the subordinate TMR input.<br>000: Internal selection 0 (IS0)<br>001: Internal selection 1 (IS1)<br>010: Internal selection 2 (IS2)<br>011: Internal selection 3 (IS3)<br>100: C1IRAW input detector (C1INC)<br>101: Filtered input 1 (C1IF1)<br>110: Filtered input 2 (C1IF2)<br>111: External input (EXT)<br>Please refer to Table 59 for more information on ISx for each timer.  |
| Bit 3     | Reserved | 0x0         | resd | Kept at its default value.   |

| Bit     | Name  | Reset value | Type | Description   |
|---------|-------|-------------|------|---|
| Bit 2:0 | SMSEL | 0x0         | rw   | Subordinate TMR mode selection<br>000: Slave mode is disabled<br>001: Encoder mode A<br>010: Encoder mode B<br>011: Encoder mode C<br>100: Reset mode – Rising edge of the TRGIN input reinitializes the counter<br>101: Suspend mode – The counter starts counting when the TRGIN is high<br>110: Trigger mode – A trigger event is generated at the rising edge of the TRGIN input<br>111: External clock mode A – Rising edge of the TRGIN input clocks the counter<br><br>Note: Please refer to count mode section for the details on encoder mode A/B/C. |

#### TMR1 DMA/interrupt enable register (TMR1\_IDEN)

| Bit    | Name     | Reset value | Type | Description  |
|--------|----------|-------------|------|--|
| Bit 15 | Reserved | 0x0         | resd | Kept at its default value.                                     |
| Bit 14 | TDEN     | 0x0         | rw   | Trigger DMA request enable<br>0: Disabled<br>1: Enabled        |
| Bit 13 | HALLDE   | 0x0         | rw   | HALL DMA request enable<br>0: Disabled<br>1: Enabled           |
| Bit 12 | C4DEN    | 0x0         | rw   | Channel 4 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 11 | C3DEN    | 0x0         | rw   | Channel 3 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 10 | C2DEN    | 0x0         | rw   | Channel 2 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 9  | C1DEN    | 0x0         | rw   | Channel 1 DMA request enable<br>0: Disabled<br>1: Enabled      |
| Bit 8  | OVFDEN   | 0x0         | rw   | Overflow event DMA request enable<br>0: Disabled<br>1: Enabled |
| Bit 7  | BRKIE    | 0x0         | rw   | Brake interrupt enable<br>0: Disabled<br>1: Enabled            |
| Bit 6  | TIEN     | 0x0         | rw   | Trigger interrupt enable<br>0: Disabled<br>1: Enabled          |
| Bit 5  | HALLIEN  | 0x0         | rw   | HALL interrupt enable<br>0: Disabled<br>1: Enabled             |
| Bit 4  | C4IEN    | 0x0         | rw   | Channel 4 interrupt enable<br>0: Disabled<br>1: Enabled        |



| Bit   | Name  | Reset value | Type | Description   |
|-------|-------|-------------|------|---|
| Bit 3 | C3IEN | 0x0         | rw   | Channel 3 interrupt enable<br>0: Disabled<br>1: Enabled |
| Bit 2 | C2IEN | 0x0         | rw   | Channel 2 interrupt enable<br>0: Disabled<br>1: Enabled |
| Bit 1 | C1IEN | 0x0         | rw   | Channel 1 interrupt enable<br>0: Disabled<br>1: Enabled |
| Bit 0 | OVIEN | 0x0         | rw   | Overflow interrupt enable<br>0: Disabled<br>1: Enabled  |

#### TMR1 interrupt status register (TMR1\_ISTS)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15:13 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 12    | C4RF     | 0x0         | rw0c | Channel 4 recapture flag<br>Please refer to C1RF description.   |
| Bit 11    | C3RF     | 0x0         | rw0c | Channel 3 recapture flag<br>Please refer to C1RF description.   |
| Bit 10    | C2RF     | 0x0         | rw0c | Channel 2 recapture flag<br>Please refer to C1RF description.   |
| Bit 9     | C1RF     | 0x0         | rw0c | Channel 1 recapture flag<br>This bit indicates whether a recapture is detected when C1IF = 1. This bit is set by hardware, and cleared by writing "0".<br>0: No capture is detected<br>1: Capture is detected.  |
| Bit 8     | Reserved | 0x0         | resd | Default value   |
| Bit 7     | BRKIF    | 0x0         | rw0c | Brake interrupt flag<br>This bit indicates whether the brake input is active or not. It is set by hardware and cleared by writing "0"<br>0: Inactive level<br>1: Active level   |
| Bit 6     | TRGIF    | 0x0         | rw0c | Trigger interrupt flag<br>This bit is set by hardware on a trigger event. It is cleared by writing "0".<br>0: No trigger event occurs<br>1: Trigger event is generated.<br>Trigger event: an active edge is detected on TRGIN input, or any edge in suspend mode. |
| Bit 5     | HALLIF   | 0x0         | rw0c | HALL interrupt flag<br>This bit is set by hardware on HALL event. It is cleared by writing "0".<br>0: No Hall event occurs.<br>1: Hall event is detected.<br>HALL event: CxEN, CxCEN and CxOCTRL are updated.   |
| Bit 4     | C4IF     | 0x0         | rw0c | Channel 4 interrupt flag<br>Please refer to C1IF description.   |
| Bit 3     | C3IF     | 0x0         | rw0c | Channel 3 interrupt flag<br>Please refer to C1IF description.   |

| Bit   | Name  | Reset value | Type | Description  |
|-------|-------|-------------|------|--|
| Bit 2 | C2IF  | 0x0         | rw0c | Channel 2 interrupt flag<br>Please refer to C1IF description.  |
| Bit 1 | C1IF  | 0x0         | rw0c | Channel 1 interrupt flag<br>If the channel 1 is configured as input mode:<br>This bit is set by hardware on a capture event. It is cleared by software or read access to the TMRx_C1DT<br>0: No capture event occurs<br>1: Capture event is generated<br>If the channel 1 is configured as output mode:<br>This bit is set by hardware on a compare event. It is cleared by software.<br>0: No compare event occurs<br>1: Compare event is generated |
| Bit 0 | OVFIF | 0x0         | rw0c | Overflow interrupt flag<br>This bit is set by hardware on an overflow event. It is cleared by software.<br>0: No overflow event occurs<br>1: Overflow event is generated. If OVFE = 0 and OVFS = 0 in the TMRx_CTRL1 register:<br>– An overflow event is generated when OVFG = 1 in the TMRx_SWEVE register;<br>– An overflow event is generated when the counter CVAL is reinitialized by a trigger event.  |

#### TMR1 software event register (TMR1\_SWEVT)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 15:8 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 7    | BRKSWTR  | 0x0         | wo   | Brake event triggered by software<br>This bit is set by software to generate a brake event.<br>0: No effect<br>1: Generate a brake event.  |
| Bit 6    | TRGSWTR  | 0x0         | rw   | Trigger event triggered by software<br>This bit is set by software to generate a trigger event.<br>0: No effect<br>1: Generate a trigger event.  |
| Bit 5    | HALLSWTR | 0x0         | wo   | HALL event triggered by software<br>This bit is set by software to generate a HALL event.<br>0: No effect<br>1: Generate a HALL event.<br>Note: This bit acts only on channels that have complementary output. |
| Bit 4    | C4SWTR   | 0x0         | wo   | Channel 4 event triggered by software<br>Please refer to C1M description.  |
| Bit 3    | C3SWTR   | 0x0         | wo   | Channel 3 event triggered by software<br>Please refer to C1M description.  |
| Bit 2    | C2SWTR   | 0x0         | wo   | Channel 2 event triggered by software<br>Please refer to C1M description.  |
| Bit 1    | C1SWTR   | 0x0         | wo   | Channel 1 event triggered by software<br>This bit is set by software to generate a channel 1 event.<br>0: No effect<br>1: Generate a channel 1 event.  |

| Bit   | Name    | Reset value | Type | Description  |
|-------|---------|-------------|------|--|
| Bit 0 | OVFSWTR | 0x0         | wo   | Overflow event triggered by software<br>This bit is set by software to generate an overflow event.<br>0: No effect<br>1: Generate an overflow event. |

#### TMR1 channel mode register1 (TMR1\_CM1)

The channel can be used in input (capture mode) or output (compare mode). The direction of a channel is defined by the corresponding CxC bits. All the other bits of this register have different functions in input and output modes. The CxOx describes its function in output mode when the channel is in output mode, while the CxIx describes its function in output mode when the channel is in input mode. Attention must be given to the fact that the same bit can have different functions in input mode and output mode.

##### Output compare mode:

| Bit       | Name    | Reset value | Type | Description   |
|-----------|---------|-------------|------|---|
| Bit 15    | C2OSEN  | 0x0         | rw   | Channel 2 output switch enable  |
| Bit 14:12 | C2OCTRL | 0x0         | rw   | Channel 2 output control  |
| Bit 11    | C2OBEN  | 0x0         | rw   | Channel 2 output buffer enable  |
| Bit 10    | C2OIEN  | 0x0         | rw   | Channel 2 output enable immediately   |
| Bit 9:8   | C2C     | 0x0         | rw   | Channel 2 configuration<br>This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN = '0':<br>00: Output<br>01: Input, C2IN is mapped on C2IFP2<br>10: Input, C2IN is mapped on C1IFP2<br>11: Input, C2IN is mapped on STI. This mode works only when the internal trigger input is selected by STIS register. |
| Bit 7     | C1OSEN  | 0x0         | rw   | Channel 1 output switch enable<br>0: C1ORAW is not affected by EXT input.<br>1: Once a high level is detect on EXT input, clear C1ORAW.   |

| Bit      | Name    | Reset value | Type | Description   |
|----------|---------|-------------|------|---|
| Bit 6: 4 | C1OCTRL | 0x0         | rw   | <p>Channel 1 output control</p> <p>This field defines the behavior of the original signal C1ORAW.</p> <p>000: Disconnected. C1ORAW is disconnected from C1OUT;</p> <p>001: C1ORAW is high when TMRx_CVAL = TMRx_C1DT</p> <p>010: C1ORAW is low when TMRx_CVAL = TMRx_C1DT</p> <p>011: Switch C1ORAW level when TMRx_CVAL = TMRx_C1DT</p> <p>100: C1ORAW is forced low</p> <p>101: C1ORAW is forced high.</p> <p>110: PWM mode A</p> <ul style="list-style-type: none"> <li>– OWCDIR = 0, C1ORAW is high once TMRx_C1DT &gt; TMRx_CVAL, else low;</li> <li>– OWCDIR = 1, C1ORAW is low once TMRx_C1DT &lt; TMRx_CVAL, else high;</li> </ul> <p>111: PWM mode B</p> <ul style="list-style-type: none"> <li>– OWCDIR = 0, C1ORAW is low once TMRx_C1DT &gt; TMRx_CVAL, else high;</li> <li>– OWCDIR = 1, C1ORAW is high once TMRx_C1DT &lt; TMRx_CVAL, else low.</li> </ul> <p>Note: In the configurations other than '000', the C1OUT is connected to C1ORAW. The C1OUT output level is not only subject to the changes of C1ORAW, but also the output polarity set by CCTRL.</p> |
| Bit 3    | C1OBEN  | 0x0         | rw   | <p>Channel 1 output buffer enable</p> <p>0: Buffer function of TMRx_C1DT is disabled. The new value written to the TMRx_C1DT takes effect immediately.</p> <p>1: Buffer function of TMRx_C1DT is enabled. The value to be written to the TMRx_C1DT is stored in the buffer register, and can be sent to the TMRx_C1DT register only on an overflow event.</p>   |
| Bit 2    | C1OIEN  | 0x0         | rw   | <p>Channel 1 output enable immediately</p> <p>In PWM mode A or B, this bit is used to accelerate the channel 1 output's response to the trigger event.</p> <p>0: Need to compare the CVAL with C1DT before generating an output</p> <p>1: No need to compare the CVAL and C1DT. An output is generated immediately when a trigger event occurs.</p>   |
| Bit 1: 0 | C1C     | 0x0         | rw   | <p>Channel 1 configuration</p> <p>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':</p> <p>00: Output</p> <p>01: Input, C1IN is mapped on C1IFP1</p> <p>10: Input, C1IN is mapped on C2IFP1</p> <p>11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.</p>  |

#### Input capture mode:

| Bit       | Name   | Reset value | Type | Description  |
|-----------|--------|-------------|------|--|
| Bit 15:12 | C2DF   | 0x0         | rw   | Channel 2 digital filter   |
| Bit 11:10 | C2IDIV | 0x0         | rw   | Channel 2 input divider  |
| Bit 9: 8  | C2C    | 0x0         | rw   | Channel 2 configuration<br>This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN = '0':<br>00: Output<br>01: Input, C2IN is mapped on C2IFP2<br>10: Input, C2IN is mapped on C1IFP2<br>11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.  |
| Bit 7: 4  | C1DF   | 0x0         | rw   | Channel 1 digital filter<br>This field defines the digital filter of the channel 1. N stands for the number of filtering, indicating that the input edge can pass the filter only after N sampling events.<br>0000: No filter, sampling is done at $f_{DTS}$<br>0001: $f_{SAMPLING} = f_{CK\_INT}$ , N = 2<br>0010: $f_{SAMPLING} = f_{CK\_INT}$ , N = 4<br>0011: $f_{SAMPLING} = f_{CK\_INT}$ , N = 8<br>0100: $f_{SAMPLING} = f_{DTS}/2$ , N = 6<br>0101: $f_{SAMPLING} = f_{DTS}/2$ , N = 8<br>0110: $f_{SAMPLING} = f_{DTS}/4$ , N = 6<br>0111: $f_{SAMPLING} = f_{DTS}/4$ , N = 8<br>1000: $f_{SAMPLING} = f_{DTS}/8$ , N = 6<br>1001: $f_{SAMPLING} = f_{DTS}/8$ , N = 8<br>1010: $f_{SAMPLING} = f_{DTS}/16$ , N = 5<br>1011: $f_{SAMPLING} = f_{DTS}/16$ , N = 6<br>1100: $f_{SAMPLING} = f_{DTS}/16$ , N = 8<br>1101: $f_{SAMPLING} = f_{DTS}/32$ , N = 5<br>1110: $f_{SAMPLING} = f_{DTS}/32$ , N = 6<br>1111: $f_{SAMPLING} = f_{DTS}/32$ , N = 8 |
| Bit 3: 2  | C1IDIV | 0x0         | rw   | Channel 1 input divider<br>This field defines Channel 1 input divider.<br>00: No divider. An input capture is generated at each active edge.<br>01: An input compare is generated every 2 active edges<br>10: An input compare is generated every 4 active edges<br>11: An input compare is generated every 8 active edges<br>Note: the divider is reset once C1EN = '0'.  |
| Bit 1: 0  | C1C    | 0x0         | rw   | Channel 1 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN = '0':<br>00: Output<br>01: Input, C1IN is mapped on C1IFP1<br>10: Input, C1IN is mapped on C2IFP1<br>11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.  |

#### TMR1 channel mode register 2 (TMR1\_CM2)

The channel can be used in input (capture mode) or output (compare mode). The direction of a channel is defined by the corresponding CxC bits. All the other bits of this register have different functions in input and output modes. The CxOx describes its function in output mode when the channel is in output mode, while the CxIx describes its function in output mode when the channel is

in input mode. Attention must be given to the fact that the same bit can have different functions in input mode and output mode.

#### Output compare mode:

| Bit       | Name    | Reset value | Type | Description   |
|-----------|---------|-------------|------|---|
| Bit 15    | C4OSEN  | 0x0         | rw   | Channel 4 output switch enable  |
| Bit 14:12 | C4OCTRL | 0x0         | rw   | Channel 4 output control  |
| Bit 11    | C4OBEN  | 0x0         | rw   | Channel 4 output buffer enable  |
| Bit 10    | C4OIEN  | 0x0         | rw   | Channel 4 output enable immediately   |
| Bit 9: 8  | C4C     | 0x0         | rw   | Channel 4 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C4EN = '0':<br>00: Output<br>01: Input, C4IN is mapped on C4IFP4<br>10: Input, C4IN is mapped on C3IFP4<br>11: Input, C4IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS. |
| Bit 7     | C3OSEN  | 0x0         | rw   | Channel 3 output switch enable  |
| Bit 6:4   | C3OCTRL | 0x0         | rw   | Channel 3 output control  |
| Bit 3     | C3OBEN  | 0x0         | rw   | Channel 3 output buffer enable  |
| Bit 2     | C3OIEN  | 0x0         | rw   | Channel 3 output enable immediately   |
| Bit 1:0   | C3C     | 0x0         | rw   | Channel 3 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C3EN = '0':<br>00: Output<br>01: Input, C3IN is mapped on C3IFP3<br>10: Input, C3IN is mapped on C4IFP3<br>11: Input, C3IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS. |

#### Input capture mode:

| Bit       | Name   | Reset value | Type | Description   |
|-----------|--------|-------------|------|---|
| Bit 15:12 | C4DF   | 0x0         | rw   | Channel 4 digital filter  |
| Bit 11:10 | C4IDIV | 0x0         | rw   | Channel 4 input divider   |
| Bit 9:8   | C4C    | 0x0         | rw   | Channel 4 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C4EN = '0':<br>00: Output<br>01: Input, C4IN is mapped on C4IFP4<br>10: Input, C4IN is mapped on C3IFP4<br>11: Input, C4IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS. |
| Bit 7:4   | C3DF   | 0x0         | rw   | Channel 3 digital filter  |
| Bit 3:2   | C3IDIV | 0x0         | rw   | Channel 3 input divider   |
| Bit 1:0   | C3C    | 0x0         | rw   | Channel 3 configuration<br>This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C3EN = '0':<br>00: Output<br>01: Input, C3IN is mapped on C3IFP3<br>10: Input, C3IN is mapped on C4IFP3<br>11: Input, C3IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS. |

### TMR1 Channel control register (TMR1\_CCTRL)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15:14 | Reserved | 0x0         | resd | Kept its default value.   |
| Bit 13    | C4P      | 0x0         | rw   | Channel 4 polarity<br>Please refer to C1P description.  |
| Bit 12    | C4EN     | 0x0         | rw   | Channel 4 enable<br>Please refer to C1EN description.   |
| Bit 11    | C3CP     | 0x0         | rw   | Channel 3 complementary polarity<br>Please refer to C1P description.  |
| Bit 10    | C3CEN    | 0x0         | rw   | Channel 3 complementary enable<br>Please refer to C1EN description.   |
| Bit 9     | C3P      | 0x0         | rw   | Channel 3 polarity<br>Please refer to C1P description.  |
| Bit 8     | C3EN     | 0x0         | rw   | Channel 3 enable<br>Please refer to C1EN description.   |
| Bit 7     | C2CP     | 0x0         | rw   | Channel 2 complementary polarity<br>Please refer to C1P description.  |
| Bit 6     | C2CEN    | 0x0         | rw   | Channel 2 complementary enable<br>Please refer to C1EN description.   |
| Bit 5     | C2P      | 0x0         | rw   | Channel 2 polarity<br>Please refer to C1P description.  |
| Bit 4     | C2EN     | 0x0         | rw   | Channel 2 enable<br>Please refer to C1EN description.   |
| Bit 3     | C1CP     | 0x0         | rw   | Channel 1 complementary polarity<br>0: C1COUT is active high.<br>1: C1COUT is active low.   |
| Bit 2     | C1CEN    | 0x0         | rw   | Channel 1 complementary enable<br>0: Output is disabled.<br>1: Output is enabled.   |
| Bit 1     | C1P      | 0x0         | rw   | Channel 1 polarity<br>When the channel 1 is configured as output mode:<br>0: C1OUT is active high<br>1: C1OUT is active low<br>When the channel 1 is configured as input mode:<br>0: C1IN active edge is on its rising edge. When used as external trigger, C1IN is not inverted.<br>1: C1IN active edge is on its falling edge. When used as external trigger, C1IN is inverted. |
| Bit 0     | C1EN     | 0x0         | rw   | Channel 1 enable<br>0: Input or output is disabled<br>1: Input or output is enabled   |

**Table 62. Complementary output channel CxOUT and CxCOUT control bits with brake function**

| Control bit |             |            |          |           | Output state <sup>(1)</sup>  |  |
|-------------|-------------|------------|----------|-----------|--|--|
| OEN bit     | FCSODIS bit | FCSEEN bit | CxEN bit | CxCEN bit | CxOUT output state   | CxCOUT output state  |
| 1           | X           | 0          | 0        | 0         | Output disabled<br>(no driven by the timer)<br>CxOUT = 0, Cx_EN = 0  | Output disabled<br>(no driven by the timer)<br>CxCOUT = 0, CxCEN = 0             |
|             |             | 0          | 0        | 1         | Output disabled<br>(no driven by the timer)<br>CxOUT = 0, Cx_EN = 0  | CxORAW + polarity,<br>CxCOUT = CxORAW xor CxCP,<br>CxCEN = 1                     |
|             |             | 0          | 1        | 0         | CxORAW + polarity<br>CxOUT = CxORAW xor CxP,<br>Cx_EN = 1  | Output disabled<br>(no driven by the timer)<br>CxCOUT = 0, CxCEN = 0             |
|             |             | 0          | 1        | 1         | CxORAW+polarity+dead-time,<br>Cx_EN = 1  | CxORAW inverted + polarity +<br>dead-time,<br>CxCEN = 1                          |
|             |             | 1          | 0        | 0         | Output disabled<br>(no driven by the timer)<br>CxOUT = CxP, Cx_EN = 0  | Output disabled<br>(no driven by the timer)<br>CxCOUT = CxCP, CxCEN = 0          |
|             |             | 1          | 0        | 1         | Off-state<br>(Output enabled with inactive level)<br>CxOUT = CxP, Cx_EN = 1  | CxORAW + polarity,<br>CxCOUT = CxORAW xor CxCP,<br>CxCEN = 1                     |
|             |             | 1          | 1        | 0         | CxORAW + polarity,<br>CxOUT = CxORAW xor CxP,<br>Cx_EN = 1   | Off-state<br>(Output enabled with inactive<br>level)<br>CxCOUT = CxCP, CxCEN = 1 |
|             |             | 1          | 1        | 1         | CxORAW+ polarity+dead-time,<br>Cx_EN = 1   | CxORAW inverted + polarity +<br>dead-time,<br>CxCEN = 1                          |
| 0           | 0           | X          | 0        | 0         | Output disabled<br>(corresponding IO is not driven by the timer, IO floating)  |  |
|             | 0           |            | 0        | 1         | Asynchronously: CxOUT = CxP, Cx_EN = 0, CxCOUT = CxCP,<br>CxCEN = 0;   |  |
|             | 0           |            | 1        | 0         | If the clock is present: after a dead-time, CxOUT = CxIOS, CxCOUT<br>= CxCIOS, assuming that CxIOS and CxCIOS do not correspond to     |  |
|             | 0           |            | 1        | 1         | CxOUT and CxCOUT active level.   |  |
|             | 1           |            | 0        | 0         | CxEN = CxCEN = 0: Output disabled<br>(corresponding IO is not driven by the timer, IO floating)  |  |
|             | 1           |            | 0        | 1         | In other cases: Off-state (Output enabled with inactive level)<br>Asynchronously: CxOUT = CxP, Cx_EN = 1, CxCOUT = CxCP,<br>CxCEN = 1; |  |
|             | 1           |            | 1        | 0         | If the clock is present: after a dead-time, CxOUT = CxIOS, CxCOUT<br>= CxCIOS, assuming that CxIOS and CxCIOS do not correspond to     |  |
|             | 1           |            | 1        | 1         | CxOUT and CxCOUT active level.   |  |

Note: 1. If the two outputs of a channel are not used (CxEN = CxCEN = 0), CxIOS, CxCIOS, CxP and CxCP must be cleared.

2. The state of the external I/O pins connected to the complementary CxOUT and CxCOUT channels depends on the CxOUT and CxCOUT channel state and the GPIO and the IOMUX registers.



### TMR1 counter value (TMR1\_CVAL)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---------------|
| Bit 15:0 | CVAL | 0x0         | rw   | Counter value |

### TMR1 division value (TMR1\_DIV)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | DIV  | 0x0         | rw   | Divider value<br>The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK} / (DIV[15:0] + 1)$ .<br>The value of this register is transferred to the actual prescaler register when an overflow event occurs. |

### TMR1 period register (TMR1\_PR)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | PR   | 0x0         | rw   | Period value<br>This defines the period value of the TMRx counter. The timer stops working when the period value is 0. |

### TMR1 repetition period register (TMR1\_RPR)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | RPR  | 0x0         | rw   | Repetition of period value<br>This field is used to reduce the generation rate of overflow events. An overflow event is generated when the repetition counter reaches 0. |

### TMR1 channel 1 data register (TMR1\_C1DT)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | C1DT | 0x0         | rw   | Channel 1 data register<br>When the channel 1 is configured as input mode:<br>The C1DT is the CVAL value stored by the last channel 1 input event (C1IN)<br>When the channel 1 is configured as output mode:<br>C1DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C1OBEN bit, and the corresponding output is generated on C1OUT as configured. |

### TMR1 channel 2 data register (TMR1\_C2DT)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 15:0 | C2DT | 0x0         | rw   | Channel 2 data register<br>When the channel 2 is configured as input mode:<br>The C2DT is the CVAL value stored by the last channel 2 input event (C1IN)<br>When the channel 2 is configured as output mode:<br>C2DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C2OBEN bit, and the corresponding output is generated on C2OUT as configured. |

#### TMR1 channel 3 data register (TMR1\_C3DT)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 15:0 | C3DT | 0x0         | rw   | <p>Channel 3 data register</p> <p>When the channel 3 is configured as input mode:<br/>The C3DT is the CVAL value stored by the last channel 3 input event (C1IN)</p> <p>When the channel 3 is configured as output mode:<br/>C3DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C3OBEN bit, and the corresponding output is generated on C3OUT as configured.</p> |

#### TMR1 channel 4 data register (TMRx\_C4DT)

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 15:0 | C4DT | 0x0         | rw   | <p>Channel 4 data register</p> <p>When the channel 4 is configured as input mode:<br/>The C4DT is the CVAL value stored by the last channel 4 input event (C1IN)</p> <p>When the channel 3 is configured as output mode:<br/>C4DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C4OBEN bit, and the corresponding output is generated on C4OUT as configured.</p> |

#### TMR1 brake register (TMR1\_BRK)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:17 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 19:16 | BKF      | 0x0         | rw   | <p>Brake input filter</p> <p>This field is used to set the filter for brake input. The filter number N indicates that the input edge can pass through filter only after N sampling events.</p> <p>0000: No filter, sampling is done at <math>f_{DTS}</math></p> <p>0001: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 2</math></p> <p>0010: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 4</math></p> <p>0011: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 8</math></p> <p>0100: <math>f_{SAMPLING} = f_{DTS}/2</math>, <math>N = 6</math></p> <p>0101: <math>f_{SAMPLING} = f_{DTS}/2</math>, <math>N = 8</math></p> <p>0110: <math>f_{SAMPLING} = f_{DTS}/4</math>, <math>N = 6</math></p> <p>0111: <math>f_{SAMPLING} = f_{DTS}/4</math>, <math>N = 8</math></p> <p>1000: <math>f_{SAMPLING} = f_{DTS}/8</math>, <math>N = 6</math></p> <p>1001: <math>f_{SAMPLING} = f_{DTS}/8</math>, <math>N = 8</math></p> <p>1010: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 5</math></p> <p>1011: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 6</math></p> <p>1100: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 8</math></p> <p>1101: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 5</math></p> <p>1110: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 6</math></p> <p>1111: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 8</math></p> |
| Bit 15    | OEN      | 0x0         | rw   | <p>Output enable</p> <p>This bit acts on the channels as output. It is used to enable CxOUT and CxCOU outputs.</p> <p>0: Disabled</p> <p>1: Enabled</p>  |

| Bit     | Name    | Reset value | Type | Description   |
|---------|---------|-------------|------|---|
| Bit 14  | AOEN    | 0x0         | rw   | Automatic output enable<br>OEN is set automatically at an overflow event.<br>0: Disabled<br>1: Enabled  |
| Bit 13  | BRKV    | 0x0         | rw   | Brake input validity<br>This bit is used to select the active level of a brake input.<br>0: Brake input is active low.<br>1: Brake input is active high.  |
| Bit 12  | BRKEN   | 0x0         | rw   | Brake enable<br>This bit is used to enable brake input.<br>0: Brake input is disabled.<br>1: Brake input is enabled.  |
| Bit 11  | FCSOEN  | 0x0         | rw   | Frozen channel status when holistic output enable<br>This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN = 1.<br>0: CxOUT/CxCOUT outputs are disabled.<br>1: CxOUT/CxCOUT outputs are enabled. Output inactive level.  |
| Bit 10  | FCSODIS | 0x0         | rw   | Frozen channel status when holistic output disable<br>This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN = 0.<br>0: CxOUT/CxCOUT outputs are disabled.<br>1: CxOUT/CxCOUT outputs are enabled. Output idle level.   |
| Bit 9:8 | WPC     | 0x0         | rw   | Write protection configuration<br>This field is used to enable write protection.<br>00: Write protection is OFF.<br>01: Write protection level 3, and the following bits are write protected:<br>TMRx_BRK: DTC, BRKEN, BRKV and AOEN<br>TMRx_CTRL2: CxIOS and CxCIOS<br>10: Write protection level 2. The following bits and all bits in level 3 are write protected:<br>TMRx_CCTRL: CxP and CxCP<br>TMRx_BRK: FCSODIS and FCSOEN<br>11: Write protection level 1. The following bits and all bits in level 2 are write protected:<br>TMRx_CMx: C2OCTRL and C2OBEN<br>Note: Once WPC > 0, its content remains frozen until the next system reset. |
| Bit 7:0 | DTC     | 0x00        | rw   | Dead-time configuration<br>This field defines the duration of the dead-time insertion. The 3-bit MSB of DTC[7:0] is used for function selection:<br>0xx: DT = DTC [7:0] × TDTS<br>10x: DT = (64 + DTC [5:0]) × TDTS × 2<br>110: DT = (32 + DTC [4:0]) × TDTS × 8<br>111: DT = (32 + DTC [4:0]) × TDTS × 16  |

Note: Based on lock configuration, BKF, AOEN, BRKV, BRKEN, FCSODIS, FCSOEN and DTC[7:0] can all be write protected. Thus it is necessary to configure write protection when writing to the TMRx\_BRK register for the first time.

### TMR1 DMA control register (TMR1\_DMACTRL)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15:13 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 12:8  | DTB      | 0x0         | rw   | DMA transfer bytes<br>This field defines the number of DMA transfers:<br>00000: 1 byte      00001: 2 bytes<br>00010: 3 bytes     00011: 4 bytes<br>.....<br>10000: 17 bytes    10001: 18 bytes |
| Bit 7:5   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 4:0   | ADDR     | 0x0         | rw   | DMA transfer address offset<br>ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register:<br>00000: TMRx_CTRL1<br>00001: TMRx_CTRL2<br>00010: TMRx_STCTRL<br>.....     |

### TMR1 DMA data register (TMR1\_DMADT)

| Bit      | Name  | Reset value | Type | Description   |
|----------|-------|-------------|------|---|
| Bit 15:0 | DMADT | 0x0         | rw   | DMA data register<br>A write/read operation to the DMADT register accesses any TMR register located at the following address:<br>TMRx peripheral address + ADDR × 4 to TMRx peripheral address + ADDR × 4 + DTB × 4 |

# 15 Window watchdog timer (WWDT)

## WWDT introduction

The window watchdog downcounter must be reloaded in a limited time window to prevent the watchdog circuits from generating a system reset. The window watch dog is used to detect the occurrence of system malfunctions.

The window watchdog timer is clocked by a divided APB1\_CLK. The precision of the APB1\_CLK enables the window watchdog to take accurate control of the limited window.

## WWDT main features

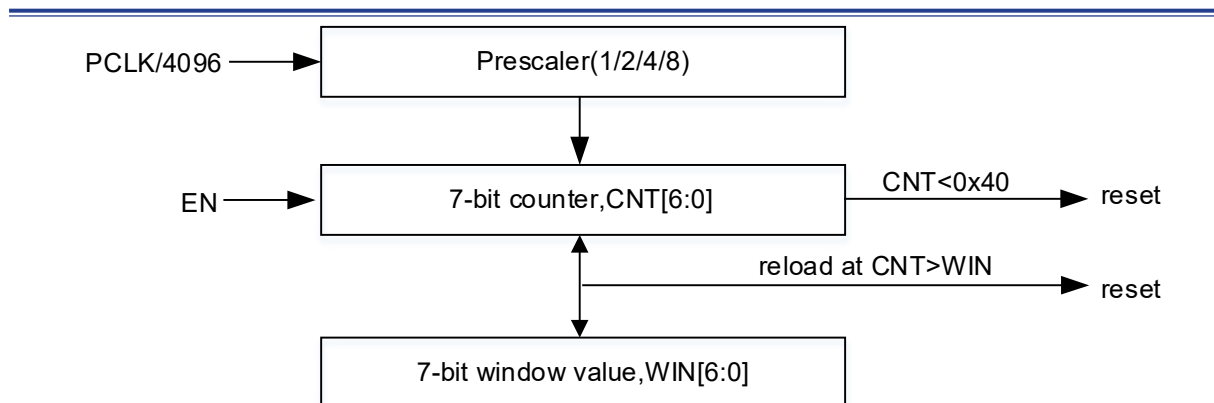
- 7-bit downcounter
- If the watchdog is enabled, a system reset is generated when the value of the downcounter is less than 0x40 or when the downcounter is reloaded outside the window.
- The downcounter can be reloaded by enabling the counter interrupt.

## WWDT function overview

If the watchdog is enabled, a system reset is generated at the following conditions:

When the 7-bit downcounter scrolls from 0x40 to 0x3F;

When the counter is reloaded while the 7-bit downcounter is greater than the value programmed in the window register.



**Figure 210. Window watchdog block diagram**

To prevent system reset, the counter must be reloaded only when its value is less than the value stored in the window register and greater than 0x40.

The WWDT counter is clocked by a divided APB1\_CLK, with the division factor being defined by the DIV[1:0] bit in the WWDT\_CFG register. The counter value determines the maximum counter period before the watchdog generates a reset. The WIN[6:0] bit can be used to configure the window value.

WWDT offers reload counter interrupt feature. If enabled, the WWDT will set the RLDF flag when the counter value reaches 0x40h, and an interrupt is generated accordingly. The interrupt service routine (ISTS) can be used to reload the counter to prevent a system reset. Note that if CNT[6] = 0, setting the WWDTEN bit will generate a system reset, so the CNT[6] bit must be always set (CNT[6] = 1) while writing to the WWDT\_CTRL register to prevent the occurrence of an immediate reset once the window watchdog is enabled.

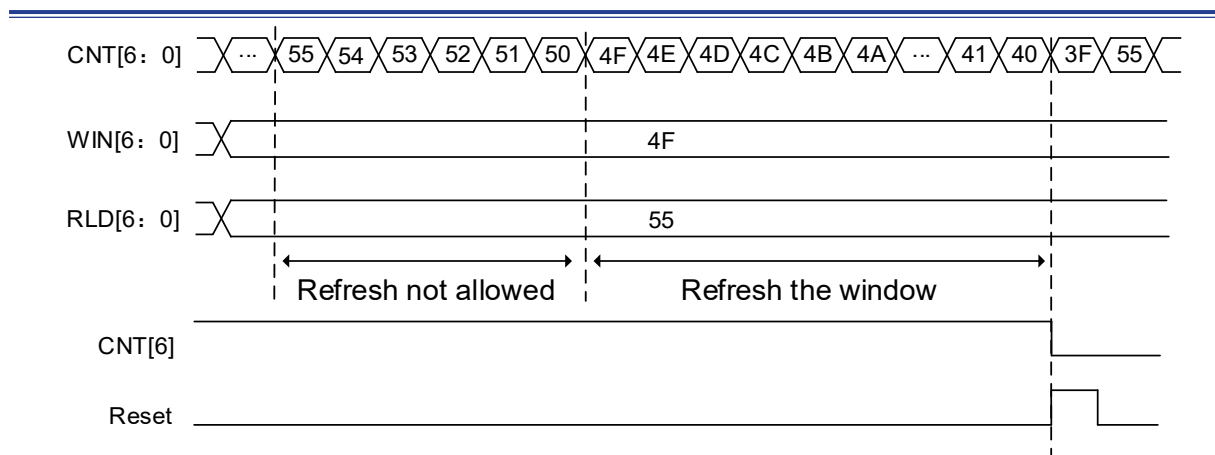
The formula to calculate the window watchdog time out:

$$T_{WWDT} = T_{PCLK1} \times 4096 \times 2^{DIV[1:0]} \times (CNT[5:0] + 1); (ms)$$

Where:  $T_{PCLK1}$  refers to APB1 clock period, in ms.

**Table 63. Minimum and maximum timeout value when PCLK1 = 72 MHz**

| Prescaler | Min. Timeout value | Max. Timeout value |
|-----------|--------------------|--------------------|
| 0         | 56.5 $\mu$ s       | 3.64 ms            |
| 1         | 113.5 $\mu$ s      | 7.28 ms            |
| 2         | 227.5 $\mu$ s      | 14.56 ms           |
| 3         | 455 $\mu$ s        | 29.12 ms           |



**Figure 211. Window watchdog timing diagram**

## Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the WWDT counter stops counting by setting the WWDT\_PAUSE in the DEBUG module. Refer to Chapter 30.2 for more information.

## WWDT registers

These peripheral registers must be accessed by word (32 bits).

**Table 64. WWDT register map and reset value**

| Register  | Offset | Reset value |
|-----------|--------|-------------|
| WWDT_CTRL | 0x00   | 0x7F        |
| WWDT_CFG  | 0x04   | 0x7F        |
| WWDT_STS  | 0x08   | 0x00        |

### Control register (WWDT\_CTRL)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:8 | Reserved | 0x000000    | resd | Kept at its default value.  |
| Bit 7    | WWDTEN   | 0x0         | rw1s | Window watchdog enable<br>0: Disabled<br>1: Enabled   |
| Bit 6:0  | CNT      | 0x7F        | rw   | This bit is set by software, but can be cleared only after reset.<br>Downcounter<br>When the counter counts down to 0x3F, a reset is generated. |

### Configuration register (WWDT\_CFG)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:10 | Reserved | 0x000000    | resd | Kept at its default value.  |
| Bit 9     | RLDIEN   | 0x0         | rw   | Reload counter interrupt<br>0: Disabled<br>1: Enabled   |
| Bit 8:7   | DIV      | 0x0         | rw   | Clock division value<br>00: PCLK1 divided by 4096<br>01: PCLK1 divided by 8192<br>10: PCLK1 divided by 16384<br>11: PCLK1 divided by 32768  |
| Bit 6:0   | WIN      | 0x7F        | rw   | Window value<br>If the counter is reloaded while its value is greater than the window register value, a reset is generated. The counter must be reloaded between 0x40 and WIN[6:0]. |

### Status register (WWDT\_STS)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31:1 | Reserved | 0x0000 0000 | resd | Kept at its default value.   |
| Bit 0    | RLDF     | 0x0         | rw0c | Reload counter interrupt flag<br>This flag is set when the downcounter reaches 0x40.<br>This bit is set by hardware and cleared by software. |

# 16 Watchdog timer (WDT)

## WDT introduction

The WDT is driven by a dedicated low-speed clock (LICK). Due to the lower clock accuracy of LICK, the WDT is best suited to the applications that have lower timing accuracy and can run independently outside the main application.

## WDT main features

- 12-bit downcounter
- The counter is clocked by LICK (can work in Stop and Standby modes)
- The counter can be configured to stop counting either in Deepsleep or Standby mode
- A system reset is generated under the following circumstances:
  - When the counter value is decremented to 0
  - When the counter is reloaded outside the window

## WDT function overview

### WDT enable:

Both software and hardware operations can be used to enable WDT. In other words, the WDT can be enabled by writing 0xCCCC to the WDT\_CMD register; or when the user enables the hardware watchdog through user system data area, the WDT will be automatically enabled after power-on reset.

### WDT reset:

When the counter value of the WDT counts down to 0, a WDT reset be generated. Thus the WDT\_CMD register must be written with the value 0xAAAA at regular intervals to reload the counter value to avoid the WDT reset.

### WDT write-protected:

The WDT\_DIV and WDT\_RLD registers are write-protected. Writing the value 0x5555 to the WDT\_CMD register will unlock write protection. The update status of these two registers are indicated by the DIVF and RLDF bits in the WDT\_STS register. If a different value is written to the WDT\_CMD register, these two registers will be re-protected. Writing the value 0xAAAA to the WDT\_CMD register also enables write protection.

### WDT clock:

The WDT counter is clocked by the LICK. The LICK is an internal RC clock with a typical value of 40 kHz, with its range falling between 30kHz and 60kHz. The timeout period is also within a certain range, so a margin should be taken into account when configuring timeout period. The LICK can be calibrated to obtain the WDT timeout with a relatively accuracy.

### WDT low power counting mode:

WDT can work in Sleep, Deepsleep and Standby modes. It is possible to stop counting in Deepsleep and Standby modes by setting the nWDT\_DEPSLP and nWDT\_STDBY bits in the User System Data area.



If the counter is disabled, it will stop decrementing as soon as the Deepsleep and Standby modes are entered. This means that the WDT would not perform a system reset in both low power modes. After waking up from these two modes, it continues downcounting from the value at the time of the entry of these modes.

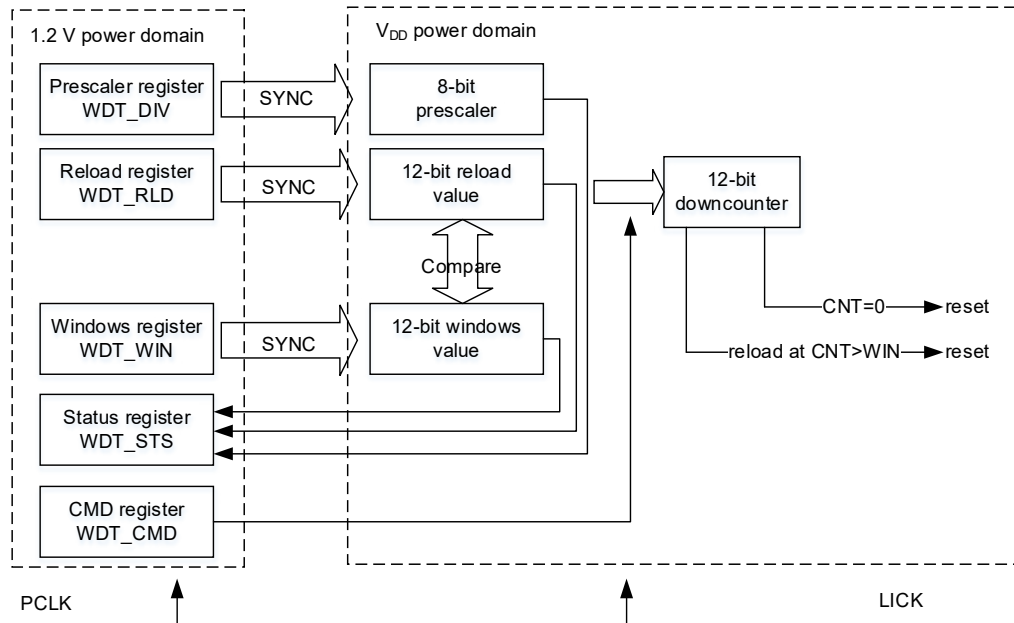


Figure 212. WDT block diagram

Table 65. WDT timeout period (LICK = 40 kHz)

| Prescaler divider | DIV[2:0] bits | Min.timeout (ms)<br>RLD[11:0] = 0x000 | Max. timeout (ms)<br>RLD[11:0] = 0xFFFF |
|-------------------|---------------|---------------------------------------|---|
| /4                | 0             | 0.1                                   | 409.6                                   |
| /8                | 1             | 0.2                                   | 819.2                                   |
| /16               | 2             | 0.4                                   | 1638.4                                  |
| /32               | 3             | 0.8                                   | 3276.8                                  |
| /64               | 4             | 1.6                                   | 6553.6                                  |
| /128              | 5             | 3.2                                   | 13107.2                                 |
| /256              | (6 or 7)      | 6.4                                   | 26214.4                                 |

## Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the WDT counter stops counting by setting the WDT\_PAUSE in the DEBUG module. Refer to Chapter 23.2 for more information.

## WDT registers

These peripheral registers must be accessed by words (32 bits).

**Table 66. WDT register and reset value**

| Register | Offset | Reset value |
|----------|--------|-------------|
| WDT_CMD  | 0x00   | 0x0000 0000 |
| WDT_DIV  | 0x04   | 0x0000 0000 |
| WDT_RLD  | 0x08   | 0x0000 0FFF |
| WDT_STS  | 0x0C   | 0x0000 0000 |
| WDT_WIN  | 0x10   | 0x0000 0FFF |

### Command register (WDT\_CMD)

(Reset in Standby mode)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.  |
| Bit 15:0  | CMD      | 0x0000      | wo   | Command register<br>0xAAAA: Reload counter<br>0x5555: Unlock write-protected WDT_DIV and WDT_RLD<br>0xCCCC: Enable WDT. If the hardware watchdog has been enabled, ignore this operation. |

### Divider register (WDT\_DIV)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31:3 | Reserved | 0x0000 0000 | resd | Kept at its default value.   |
| Bit 2:0  | DIV      | 0x0         | rw   | Clock division value<br>000: LICK divided by 4<br>001: LICK divided by 8<br>010: LICK divided by 16<br>011: LICK divided by 32<br>100: LICK divided by 64<br>101: LICK divided by 128<br>110: LICK divided by 256<br>111: LICK divided by 256<br>The write protection must be unlocked in order to enable write access to the register. The register can be read only when DIVF = 0. |

### Reload register (WDT\_RLD)

(Reset in Standby mode)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:12 | Reserved | 0x00000     | resd | Kept at its default value.  |
| Bit 11:0  | RLD      | 0xFFFF      | rw   | Reload value<br>The write protection must be unlocked in order to enable write access to the register. The register can be read only when RLDF = 0. |

## Status register (WDT\_STS)

(Reset in Standby mode)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:3 | Reserved | 0x0000 0000 | resd | Kept at its default value.  |
| Bit 2    | WINF     | 0x0         | ro   | Window value update complete flag<br>0: Window value update complete<br>1: Window value update is in progress<br>The WDT_WIN register can be written only when RLDF = 0.      |
| Bit 1    | RLDF     | 0x0         | ro   | Reload value update complete flag<br>0: Reload value update complete<br>1: Reload value update is in process.<br>The WDT_RLD register can be written only when RLDF = 0.      |
| Bit 0    | DIVF     | 0x0         | ro   | Division value update complete flag<br>0: Division value update complete<br>1: Division value update is in process.<br>The WDT_DIV register can be written only when DIVF = 0 |

## Window register (WDT\_WIN)

(Reset in Standby mode)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:12 | Reserved | 0x000000    | resd | Kept at its default value.   |
| Bit 11:0  | WIN      | 0xFFFF      | ro   | Window value<br>When the counter value is greater than the window value, the reload counter will perform a reset. The reload counter value falls between 0 and the window value. |

# 17 Enhanced real-time clock (ERTC)

## ERTC introduction

The real-time clock provides a calendar clock function. The time and date can be modified by modifying the ERTC\_TIME and ERTC\_DATE register.

The RTC module is in battery powered domain, which means that it keeps running and free from the influence of system reset as long as VBAT is powered (can only powered by VDD).

## ERTC main features

- Real-time calendar, one alarm. Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed. When the year register is a multiple of 4, it represents a leap year
- Periodic auto-wakeup
- Reference clock detection
- 1x programmable tamper detection, supporting time stamp feature
- Supports fine calibration
- 5 × battery powered registers
- 4 × interrupts: alarm A, periodic auto-wakeup, tamper detection and time stamp
- Multiplexed function output, calibration clock output, alarm event or wakeup event
- Multiplexed function input, reference clock input, one-channel tamper detection and time stamp

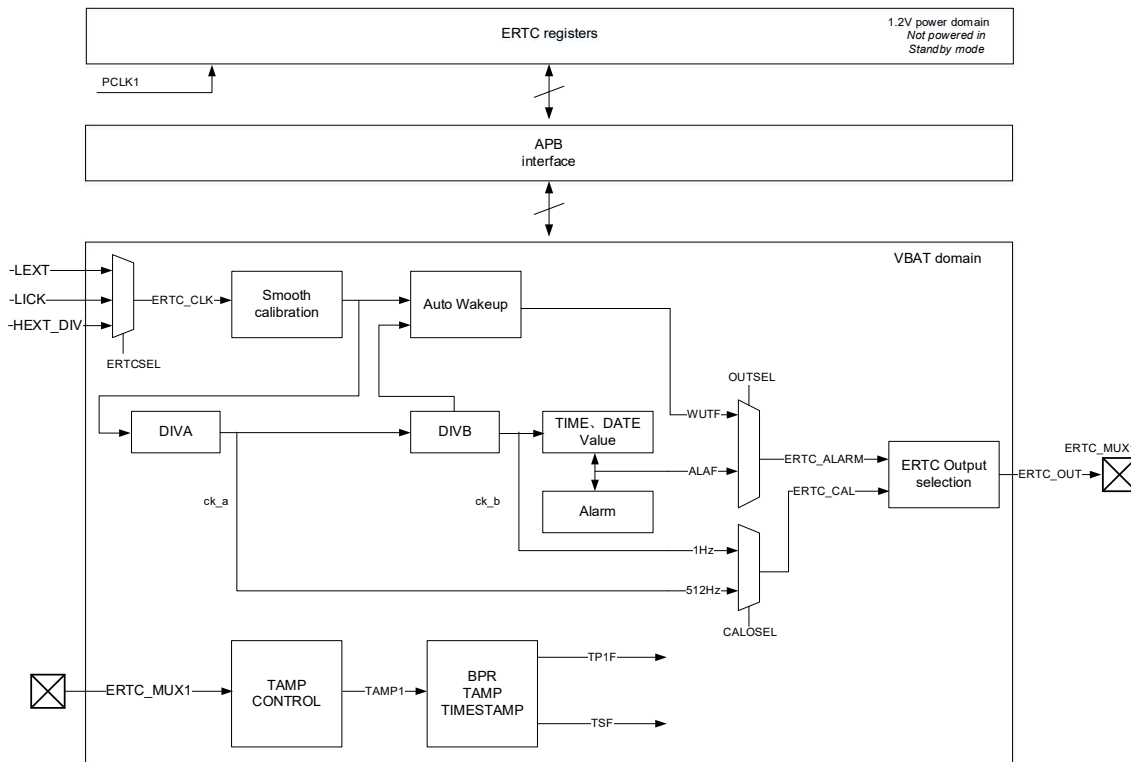


Figure 213. ERTC block diagram

## ERTC function overview

### ERTC clock

ERTC clock source (ERTC\_CLK) is selected via clock controller from a LEXT, LICK, and HEXT/32.

The ERTC embeds two dividers: A and B, programmed by the DIVA[6:0] and DIVB[14:0] respectively. It is recommended that the DIVA is configured to a higher value in order to minimum power consumption. After being divided by prescaler A and B, the ERTC\_CLK generates ck\_a and ck\_b clocks, respectively. The ck\_a is used for subsecond update, while the ck\_b is used for calendar update and periodic auto wakeup. The clock frequencies of ck\_a and ck\_b can be obtained from the following equation:

$$F_{ck\_a} = \frac{f_{ERTC\_CLK}}{DIVA + 1}$$

$$F_{ck\_b} = \frac{f_{ERTC\_CLK}}{(DIVB + 1) \times (DIVA + 1)}$$

To obtain ck\_b with frequency of 1 Hz, DIVA = 127, DIVB = 255, and 32.768 kHz LEXT should be used. This ck\_b is then used for calendar update.

## ERTC initialization

### ERTC register write protection

After a power-on reset, all ERTC registers are write protected. Such protection mechanism is not affected by the system reset. Write access to the ERTC registers (except the ERTC\_STS[14:8], ERTC\_TAMP and ERTC\_BPRx registers) can be enabled by unlocking it.

To unlock the write protection of ERTC registers, the steps below should be respected:

1. Enable power interface clock by setting PWCEN = 1 in the CRM\_APB1EN register.
2. Unlock write protection of the battery powered domain by setting BPWEN = 1 in the PWC\_CTRL register.
3. Write 0xCA and 0x53 to the ERTC\_WP register in sequence. Writing an incorrect key will activate the write protection again.

Table 67 lists the ERTC registers that can be configured only after the write protection is unlocked and when the initialization mode is entered.

**Table 67. RTC register map and reset values**

| Register    | ERTC_WP enabled  | Whether to enter initialization mode | Others                        |
|-------------|------------------|--------------------------------------|-------------------------------|
| ERTC_TIME   | Y                | Y                                    | —                             |
| ERTC_DATE   | Y                | Y                                    | —                             |
| ERTC_CTRL   | Y                | Bit 6 and 4 only                     | —                             |
| ERTC_STS    | Y, except [14:8] | —                                    | —                             |
| ERTC_DIV    | Y                | Y                                    | —                             |
| ERTC_WAT    | Y                | N                                    | Configurable when WATWF = 1   |
| ERTC_ALA    | Y                | N                                    | Configurable when ALAWF = 1   |
| ERTC_WP     | —                | —                                    | —                             |
| ERTC_SBS    | —                | —                                    | —                             |
| ERTC_TADJ   | Y                | N                                    | Configurable when TADJF = 0   |
| ERTC_TSTM   | —                | —                                    | —                             |
| ERTC_TSDT   | —                | —                                    | —                             |
| ERTC_TSSBS  | —                | —                                    | —                             |
| ERTC_SCAL   | Y                | N                                    | Configurable when CALUPDF = 0 |
| ERTC_TAMP   | N                | N                                    | —                             |
| ERTC_ALASBS | Y                | N                                    | Configurable when ALAWF = 1   |
| ERTC_BPRx   | N                | N                                    | —                             |

### Clock and calendar initialization

After the register write protection is unlocked, follow the procedure below for clock and calendar initialization:

1. Set the IMEN bit to enter initialization mode.
2. Wait until the initialization flag INITF bit is set.
3. Configure DIVB and DIVA.
4. Configure the clock and calendar values.
5. Leave the initialization mode by clearing the IMEN bit. Wait until the UPDF bit is set, indicating the completion of the calendar update. The calendar starts counting.

The ERTC also allows the fine-tuning for daylight saving time and clock.

Daylight saving time feature: It is used to increase (ADD1H = 1) or decrease (DEC1H = 1) one hour in the calendar, without completing the whole initialization process.

Clock calibration: It is used for the fine calibration of the current clock. If only DECSBS[14:0] is configured, the value will be added to the DIVB counter and a clock latency will be generated. If only ADD1S bit is set, the current clock will increase by one second. If both DECSBS[14:0] and ADDIS bit are configured, the clock will increase by a fraction of a second.

Time latency (ADD1S = 0):  $\text{DECSBS}/(\text{DIVB} + 1)$

Time advance (ADD1S = 1):  $1 - (\text{DECSBS}/(\text{DIVB} + 1))$

Note: To avoid subsecond overflow, SBS[15] = 0 must be asserted before setting the ERTC\_TADJ register. Reference clock detection and coarse digital calibration cannot be used at the same time. Thus when RCDEN = 1, coarse digital calibration is not supported.

### Reading the calendar

The ERTC offers two different ways to read the calendar, that is, synchronous read (DREN = 0) and asynchronous read (DREN = 1).

In the case of DREN = 0, the clock and calendar values can be obtained by reading a synchronous shadow register via the PCLK1. The UPDF bit is set each time the shadow register is synchronized with the ERTC calendar value located in the battery powered domain. The synchronization is performed every two ERTC\_CLK. The shadow register is reset by a system reset. To ensure consistency between the 3 values (ERTC\_SBS, ERTC\_TIME and ERTC\_DATE registers), reading lower-order registers will lock the values in the higher-order registers until the ERTC\_DATE register is read. For example, reading the ERTC\_SBS register will lock the values in the ERTC\_TIME and ERTC\_DATE registers.

In the case of DREN = 1, the ERTC will perform direct read access to the ERTC clock and calendar located in the battery powered domain with the PCLK1, avoiding the occurrence of errors caused by time synchronization. In this mode, the UPDF flag is cleared by hardware. To ensure the data is correct when reading clock and calendar, the software must read the clock and calendar registers twice, and compare the results of two read operations. If the result is not aligned, read again until that the results of two read accesses are consistent. Besides, it is also possible to compare the least significant bits of the two read operations to determine their consistency.

Note: In Standby and Deepsleep modes, the current calendar values are not copied into the shadow registers. When waking up from these two modes, UPDF = 0 must be asserted, and then wait until UPDF = 1, to ensure that the latest calendar value can be read. In synchronous read (DREN = 0) mode, the frequency of the PCLK1 must be at least seven times the ERTC\_CLK frequency. In asynchronous read (DREN = 1), an additional APB cycle is required to complete the read operations of the calendar register.

### Alarm clock initialization

The ERTC contains two programmable alarm clocks: alarm clock A and alarm clock B, and their respective interrupts.

The alarm clock value is programmed with the ERTC\_ALASBS/ERTC\_ALA (ERTC\_ALBSBS/ERTC\_ALB). When the programmed alarm value matches the calendar value, an alarm event is generated if an alarm clock is enabled. The MASKx bit can be used to selectively mask calendar fields. The calendar fields, which are masked, are not allocated with an alarm clock.

To configure the alarm clocks, the following steps should be respected:

1. Disable alarm clock A or alarm clock B (by setting ALAEN = 0 or ALBEN = 0)
2. Wait until the ALAWF or ALBWF bit is set to enable write access to the alarm clock A or B

3. Configure alarm clock A or B registers (ERTC\_ALA/ERTC\_ALASBS and ERTC\_ALB/ERTC\_ALBSBS)

4. Enable alarm clock A or B by setting ALAEN = 1 or ALBEN = 1

Note: If MASK1 = 0 in the ERTC\_ALA or ERTC\_ALB, the alarm clock can work normally only when the DIVB value is at least equal to 3.

## Periodic automatic wakeup

Periodic automatic wakeup unit is used to wake up ERTC from low power consumption modes automatically. The period is programmed with the VAL[15:0] bit (When WATCLK[2] = 1, it is extended to 17 bits, and the wakeup counter value is VAL + 216). When the wakeup counter value drops from the VAL to zero, the WATF bit is set, and a wakeup event is generated, with the wakeup counter being reloaded with the VAL value. An interrupt is also generated if a periodic wakeup interrupt is enabled.

The WATCLK[2:0] bit can be used to select a wakeup timer clock, including ERTC\_CLK/16, ERTC\_CLK/8, ERTC\_CLK/4, ERTC\_CLK/2 and ck\_b (usually 1 Hz). The cooperation between wakeup timer clocks and wakeup counter values enable users to adjust the wakeup period freely.

To enable a periodic automatic wakeup, the following steps should be respected:

1. Disable a periodic automatic wakeup by setting WATEN = 0.
2. Wait until WATWF = 1 to enable write access to the wakeup reload timer and WATCLK[2:0].
3. Configure the wakeup timer counter value and wakeup timer through VAL[15:0] and WATCLK[2:0] bits.
4. Enable a timer by setting WATEN = 1.

Note: 1. A wakeup timer is not affected by a system reset and low power consumption modes (Sleep, Deepsleep and Standby modes).

2. In debug mode, if the ERTC\_CLK is selected as wakeup clock, the counter which is used for periodic wakeup works normally.

## ERTC calibration

### Smooth digital calibration:

Smooth digital calibration has a higher and well-distributed performance than the coarse digital calibration. The calibration is performed by increasing or decreasing ERTC\_CLK in an evenly manner.

The smooth digital calibration period is around  $2^{20}$  ERTC\_CLK (32 seconds) when the ERTC\_CLK is 32.768 kHz. The DEC[8:0] bit specifies the number of pulses to be masked during the  $2^{20}$  ERTC\_CLK cycles. A maximum of 511 pulses can be removed. When the ADD bit is set, 512 pulses can be inserted during the  $2^{20}$  ERTC\_CLK cycles. When DEC[8:0] and ADD are sued together, a deviation ranging from -511 to +512 ERTC\_CLK cycles can be added during the  $2^{20}$  ERTC\_CLK cycles.

The effective calibrated frequency ( $F_{SCAL}$ ):

$$F_{SCAL} = F_{ERTC\_CLK} \times \left[ 1 + \frac{ADD \times 512 - DEC}{2^{20} + DEC - ADD \times 512} \right]$$

When the divider A is less than 3, the calibration operates as if ADD was equal to 0. The divider B value should be reduced so that each second is accelerated by 8 ERTC\_CLK cycles, which means that 256 ERTC\_CLK cycles are added every 32 seconds. When DEC[8:0] and ADD are sued together, a deviation ranging from -255 to +256 ERTC\_CLK cycles can be added during the  $2^{20}$  ERTC\_CLK cycles.



At this point, the effective calibrated frequency ( $F_{SCAL}$ ):

$$F_{SCAL} = F_{ERTC\_CLK} \times \left[ 1 + \frac{256 - DEC}{2^{20} + DEC - 256} \right]$$

It is also possible to select 8 or 16-second digital calibration period through the CAL8 and CAL16 bits. The 8-second period takes priority over 16-second. In other words, when both 8-second and 16-second are enabled, 8-second calibration period prevails.

The CALUPDF flag in the ERTC indicates the calibration status. During the configuration of ERTC\_SCAL registers, the CALUPDF bit is set, indicating that the calibration value is being updated; Once the calibration value is successfully applied, this bit is cleared automatically, indicating the completion of the calibration value update.

## Time stamp function

When time stamp event is detected on the tamper pin (valid edge is detected), the current calendar value will be stored to the time stamp register.

When a time stamp event occurs, the time stamp flag bit (TSF) in the ERTC\_STS register will be set. If a new time stamp event is detected when time stamp flag (TSF) is already set, then the time stamp overflow flag (TSOF) will be set, but the time stamp registers will remain the result of the last event. By setting the TSIEN bit, an interrupt can be generated when a time stamp event occurs.

### Usage of time stamp:

1. How to enable time stamp when a valid edge is detected on a tamper pin
  - Select a time stamp in by setting the TSPIN bit.
  - Select a rising edge or falling edge to trigger time stamp by setting the TSEDG bit.
  - Enable time stamp by setting TSEN = 1.
2. How to save time stamp on a tamper event
  - Configure tamper detection registers.
  - Enable tamper detection time stamp by setting TPTSEN = 1.

Note: The TSF bit will be set after two  $ck_a$  cycles following a time stamp event. It is suggested that users poll TSOF bit when the TSF is set.

## Tamper detection

The ERTC has one tamper detection mode: TAMP1. It can be configured as a level detection with filter or edge detection. TAMP1 can select either ERTC\_MUX.

The TPIF will be set when a valid tamper event is detected. An interrupt will also be generated if a tamper detection interrupt is enabled. If the TPTSEN bit is already set, a time stamp event will be generated accordingly. Once a tamper event occurs, the battery powered registers will be reset so as to ensure data security in the battery powered domain.

### How to configure edge detection

1. Select edge detection by setting TPFLT = 00, and select a valid edge (TP1EDG).
2. According to your needs, configure whether to activate a time stamp on a tamer event (TPTSEN = 1).
3. According to your needs, enable a tamper detection interrupt (TPIEN = 1).

### How to configure level detection with filter

1. Select level detection with filter, and valid level sampling times (TPFLT  $\neq$  00).
2. Select tamper detection valid level (TP1EDG).
3. Select tamper detection sampling frequency (through the TPFREQ bit )
4. According to your needs, enable tamper detection pull-up (setting TPPU = 1). When TPPU = 1 is asserted, tamper detection pre-charge time must be configured through the TPPR bit.
5. According to your needs, configure whether to activate a time stamp on a tamper event (TPTSEN = 1).
6. According to your needs, enable a tamper interrupt (TPIEN = 1).
7. Enable TMAP1 by setting TP1EN = 1.

In the case of edge detection mode, the following two points deserve our attention:

1. If a rising edge is configured to enable tamper detection, and the tamper detection pin turns to high level before tamper detection is enabled, then a tamper event will be detected right after the tamper detection is enabled;
2. If a falling edge is configured to enable tamper detection, and the tamper detection pin turns to low level before tamper detection is enabled, then a tamper event will be detected right after the tamper detection is enabled;

### Multiplexed function output

ERTC provides a set of multiplexed function output for the following events:

1. Clocks calibrated (OUTSEL = 0 and CALOEN = 1)
  - Output 512 Hz (CALOSEL = 0)
  - Output 1Hz (CALOSEL = 1)
2. Alarm clock A (OUTSEL = 1)
3. Wakeup events (OUTSEL = 3)

When alarm clock or wakeup events are selected (OUTSEL  $\neq$  0), it is possible to select output type (open-drain or push-pull) with the OUTTYPE bit, and output polarity with the OUTP bit.

### ERTC wakeup

ERTC can be woken up by alarm clocks, periodic auto wakeup, time stamps or tamper events. To enable an ERTC interrupt, configure as follows:

1. Configure the EXINT line corresponding to ERTC interrupts as an interrupt mode and enable it, and select a rising edge
2. Enable a NVIC channel corresponding to ERTC interrupts
3. Enable an ERTC interrupt

Table 68 lists the ERTC clock sources, events and interrupts that are able to wakeup low-power modes.

**Table 68. ERTC low-power mode wakeup**

| Clock sources | Events                    | Wake up Sleep | Wake up Deepsleep | Wakeup Standby |
|---------------|---------------------------|---------------|-------------------|----------------|
| HEXT          | Alarm clock A             | √             | ×                 | ×              |
|               | Periodic automatic wakeup | √             | ×                 | ×              |
|               | Time stamp                | √             | ×                 | ×              |
|               | Tamper event              | √             | ×                 | ×              |
| LICK          | Alarm clock A             | √             | √                 | √              |
|               | Periodic automatic wakeup | √             | √                 | √              |
|               | Time stamp                | √             | √                 | √              |
|               | Tamper event              | √             | √                 | √              |
| LEXT          | Alarm clock A             | √             | √                 | √              |
|               | Periodic automatic wakeup | √             | √                 | √              |
|               | Time stamp                | √             | √                 | √              |
|               | Tamper event              | √             | √                 | √              |

**Table 69. Interrupt control bits**

| Interrupt events          | Event flag | Interrupt enable bit | EXINT line |
|---------------------------|------------|----------------------|------------|
| Alarm clock A             | ALAF       | ALAIEN               | 17         |
| Periodic automatic wakeup | WATF       | WATIEN               | 20         |
| Time stamp                | TSF        | TSIEN                | 19         |
| Tamper event              | TP1F/TP2F  | TPIEN                | 19         |

## ERTC registers

These peripheral registers must be accessed by words (32 bits).

ERTC registers are 32-bit addressable registers.

**Table 70. ERTC register map and reset values**

| Register   | Offset | Reset value |
|------------|--------|-------------|
| ERTC_TIME  | 0x00   | 0x0000 0000 |
| ERTC_DATE  | 0x04   | 0x0000 2101 |
| ERTC_CTRL  | 0x08   | 0x0000 0000 |
| ERTC_STS   | 0x0C   | 0x0000 0007 |
| ERTC_DIV   | 0x10   | 0x007F 00FF |
| ERTC_WAT   | 0x14   | 0x0000 FFFF |
| ERTC_ALA   | 0x1C   | 0x0000 0000 |
| ERTC_WP    | 0x24   | 0x0000 0000 |
| ERTC_SBS   | 0x28   | 0x0000 0000 |
| ERTC_TADJ  | 0x2C   | 0x0000 0000 |
| ERTC_TSTM  | 0x30   | 0x0000 0000 |
| ERTC_TSDT  | 0x34   | 0x0000 000D |
| ERTC_TSSBS | 0x38   | 0x0000 0000 |
| ERTC_SCAL  | 0x3C   | 0x0000 0000 |
| ERTC_TAMP  | 0x40   | 0x0000 0000 |

| Register    | Offset    | Reset value |
|-------------|-----------|-------------|
| ERTC_ALASBS | 0x44      | 0x0000 0000 |
| ERTC_BPRx   | 0x50-0x60 | 0x0000 0000 |

### ERTC time register (ERTC\_TIME)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:23 | Reserved | 0x000       | resd | Kept at its default value.   |
| Bit 22    | AMPM     | 0x0         | rw   | AM/PM<br>0: AM<br>1: PM<br><br>Note: This bit is applicable for 12-hr format only. It is 0 for 24-hr format instead. |
| Bit 21:20 | HT       | 0x0         | rw   | Hour tens  |
| Bit 19:16 | HU       | 0x0         | rw   | Hour units   |
| Bit 15    | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 14:12 | MT       | 0x0         | rw   | Minute tens  |
| Bit 11:8  | MU       | 0x0         | rw   | Minute units   |
| Bit 7     | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 6:4   | ST       | 0x0         | rw   | Second tens  |
| Bit 3:0   | SU       | 0x0         | rw   | Second units   |

### ERTC date register (ERTC\_DATE)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:24 | Reserved | 0x00        | resd | Kept at its default value.  |
| Bit 23:20 | YT       | 0x0         | rw   | Year tens   |
| Bit 19:16 | YU       | 0x0         | rw   | Year units  |
| Bit 15:13 | WK       | 0x1         | rw   | Week day<br>0: Forbidden<br>1: Monday<br>2: Tuesday<br>3: Wednesday<br>4: Thursday<br>5: Friday<br>6: Saturday<br>7: Sunday |
| Bit 12    | MT       | 0x0         | rw   | Month tens  |
| Bit 11:8  | MU       | 0x1         | rw   | Month units   |
| Bit 7:6   | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 5:4   | DT       | 0x0         | rw   | Date tens   |
| Bit 3:0   | DU       | 0x1         | rw   | Date units  |

### ERTC control register (ERTC\_CTRL)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:24 | Reserved | 0x00        | resd | Kept at its default value.  |
| Bit 23    | CALOEN   | 0x0         | rw   | Calibration output enable<br>0: Calibration output disabled<br>1: Calibration output enabled  |
| Bit 22:21 | OUTSEL   | 0x0         | rw   | Output source selection<br>00: Output source disabled<br>01: Alarm clock A<br>10: Alarm clock B<br>11: Wakeup events  |
| Bit 20    | OUTP     | 0x0         | rw   | Output polarity<br>0: High<br>1: Low  |
| Bit 19    | CALOSEL  | 0x0         | rw   | Calibration output selection<br>0: 512 Hz<br>1: 1 Hz  |
| Bit 18    | BPR      | 0x0         | rw   | Battery powered domain data register<br>This bit in the battery powered domain is not affected by a system reset. It is used to store the daylight saving time change or others that need to be saved permanently.                    |
| Bit 17    | DEC1H    | 0x0         | wo   | Decrease 1 hour<br>0: No effect<br>1: Subtract 1 hour<br>Note: This bit is applicable only when the current hour is not 0. The next second takes effect when this bit is set (don't set this bit when the hour is being incremented). |
| Bit 16    | ADD1H    | 0x0         | wo   | Add 1 hour<br>0: No effect<br>1: Add 1 hour<br>Note: The next second takes effect when this bit is set (don't set this bit when the hour is being incremented).   |
| Bit 15    | TSIEN    | 0x0         | rw   | Timestamp interrupt enable<br>0: Timestamp interrupt disabled<br>1: Timestamp interrupt enabled   |
| Bit 14    | WATIEN   | 0x0         | rw   | Wakeup timer interrupt enable<br>0: Wakeup timer interrupt disable<br>1: Wakeup timer interrupt enabled   |
| Bit 13    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 12    | ALAIEN   | 0x0         | rw   | Alarm A interrupt enable<br>0: Alarm A interrupt disabled<br>1: Alarm A interrupt enabled   |
| Bit 11    | TSEN     | 0x0         | rw   | Timestamp enable<br>0: Timestamp disabled<br>1: Timestamp enabled   |
| Bit 10    | WATEN    | 0x0         | rw   | Wakeup timer enable<br>0: Wakeup timer disabled<br>1: Wakeup timer enabled  |
| Bit 9     | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 8     | ALAEN    | 0x0         | rw   | Alarm A enable<br>0: Alarm A disabled<br>1: Alarm A enabled   |

| Bit     | Name     | Reset value | Type | Description   |
|---------|----------|-------------|------|---|
| Bit 7   | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 6   | HM       | 0x0         | rw   | Hour mode<br>0: 24-hour format<br>1: 12-hour format   |
| Bit 5   | DREN     | 0x0         | rw   | Date/time register direct read enable<br>0: Date/time register direct read disabled. ERTC_TIME, ERTC_DATE and ERTC_SBS values are taken from the synchronized registers, which are updated once every two ERTC_CLK cycles<br>1: Date/time register direct read enabled. ERTC_TIME, ERTC_DATE and ERTC_SBS values are taken from the battery powered domain. |
| Bit 4   | RCDEN    | 0x0         | rw   | Reference clock detection enable<br>0: Reference clock detection disabled<br>1: Reference clock detection enabled   |
| Bit 3   | TSEDG    | 0x0         | rw   | Timestamp trigger edge<br>0: Rising edge<br>1: Falling edge   |
| Bit 2:0 | WATCLK   | 0x0         | rw   | Wakeup timer clock selection<br>000: ERTC_CLK/16<br>001: ERTC_CLK/8<br>010: ERTC_CLK/4<br>011: ERTC_CLK/2<br>10x: ck_a<br>11x: ck_a is selected. $2^{16}$ is added to the wakeup counter value, and wakeup time = ERTC_WAT + $2^{16}$ .<br>Note: The write access to this field is supported when WATEN = 0 and WATWF = 1.                                  |

### ERTC initialization and status register (ERTC\_STS)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:17 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 16    | CALUPDF  | 0x0         | ro   | Calibration value update complete flag<br>0: Calibration value update is complete<br>1: Calibration value update is in progress<br>This bit is automatically set when software writes to the ERTC_SCAL register. It is automatically cleared when a new calibration value is taking into account. When this bit is set, the write access to the ERTC_SCAL register is not allowed. |
| Bit 15:14 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 13    | TP1F     | 0x0         | rw0c | Tamper detection 1 flag<br>0: No tamper event<br>1: Tamper event occurs  |
| Bit 12    | TSOF     | 0x0         | rw0c | Timestamp overflow flag<br>0: No timestamp overflow<br>1: Timestamp overflow occurs<br>If a new time stamp event is detected when time stamp flag (TSF) is already set, this bit will be set by hardware.  |

| Bit    | Name     | Reset value | Type | Description   |
|--------|----------|-------------|------|---|
| Bit 11 | TSF      | 0x0         | rw0c | <p>Timestamp flag</p> <p>0: No timestamp event<br/>1: Timestamp event occurs</p> <p>It is recommended to double check the TSOF flag after reading a timestamp and clearing the TSF. Otherwise, a new timestamp event may be detected while clearing the TSF.</p> <p>Note: The clearing operation of this bit takes effect after two APB_CLK cycles.</p> |
| Bit 10 | WATF     | 0x0         | rw0c | <p>Wakeup timer flag</p> <p>0: No wakeup timer event<br/>1: Wakeup timer event occurs</p> <p>Note: The clearing operation of this bit takes effect after two APB_CLK cycles.</p>  |
| Bit 9  | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 8  | ALAF     | 0x0         | rw0c | <p>Alarm clock A flag</p> <p>0: No alarm clock event<br/>1: Alarm clock event occurs</p> <p>Note: The clearing operation of this bit takes effect after two APB_CLK cycles.</p>   |
| Bit 7  | IMEN     | 0x0         | rw   | <p>Initialization mode enable</p> <p>0: Initialization mode disabled<br/>1: Initialization mode enabled</p> <p>When an initialization mode is entered, the calendar stops running.</p>  |
| Bit 6  | IMF      | 0x0         | ro   | <p>Enter initialization mode flag</p> <p>0: Initialization mode is not entered<br/>1: Initialization mode is entered</p> <p>The ERTC_TIME, ERTC_DATE and ERTC_DIV registers can be modified only when an initialization mode is enabled (INITEN = 1) and entered (INITEF = 1).</p>  |
| Bit 5  | UPDF     | 0x0         | rw0c | <p>Calendar update flag</p> <p>0: Calendar update is in progress<br/>1: Calendar update is complete</p> <p>The UPDF bit is set each time the shadow register is synchronized with the ERTC calendar value located in the battery powered domain. The synchronization is performed every two ERTC_CLK.</p>   |
| Bit 4  | INITF    | 0x0         | ro   | <p>Calendar initialization flag</p> <p>0: Calendar has not been initialized<br/>1: Calendar has been initialized</p> <p>This bit is set when the calendar year filed (ERTC_DATE) is different from 0. It is cleared when the year is 0.</p>   |
| Bit 3  | TADJF    | 0x0         | ro   | <p>Time adjustment flag</p> <p>0: No time adjustment<br/>1: Time adjustment is in progress</p> <p>This bit is automatically set when a write access to the ERTC_TADJ register is performed. It is automatically cleared at the end of time adjustment.</p>  |
| Bit 2  | WATWF    | 0x1         | ro   | <p>Wakeup timer register allows write flag</p> <p>0: Wakeup timer register configuration not allowed<br/>1: Wakeup timer register configuration allowed</p>   |
| Bit 1  | Reserved | 0x0         | resd | Kept at its default value.  |

| Bit   | Name  | Reset value | Type | Description  |
|-------|-------|-------------|------|--|
| Bit 0 | ALAWF | 0x1         | ro   | Alarm A register allows write flag<br>0: Alarm A register write operation not allowed<br>1: Alarm A register write operation allowed |

### ERTC divider register (ERTC\_DIV)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:23 | Reserved | 0x000       | resd | Kept at its default value.   |
| Bit 22:16 | DIVA     | 0x7F        | rw   | Divider A  |
| Bit 15    | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 14:0  | DIVB     | 0x00FF      | rw   | Divider B<br>Calendar clock = $ERTC\_CLK / ((DIVA + 1) \times (DIVB + 1))$ |

### ERTC wakeup timer register (ERTC\_WAT)

| Bit       | Name     | Reset value | Type | Description                |
|-----------|----------|-------------|------|----------------------------|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value. |
| Bit 15:0  | VAL      | 0xFFFF      | rw   | Wakeup timer reload value  |

### ERTC alarm clock A register (ERTC\_ALA)

| Bit       | Name  | Reset value | Type | Description  |
|-----------|-------|-------------|------|--|
| Bit 31    | MASK4 | 0x0         | rw   | Date/week day mask<br>0: Date/week day is not masked<br>1: Alarm clock doesn't care about date/week day      |
| Bit 30    | WKSEL | 0x0         | rw   | Date/week day select<br>0: Date<br>1: Week day (DT[1:0] is not used)   |
| Bit 29:28 | DT    | 0x0         | rw   | Date tens  |
| Bit 27:24 | DU    | 0x0         | rw   | Date/week day units  |
| Bit 23    | MASK3 | 0x0         | rw   | Hour mask<br>0: No hour mask<br>1: Alarm clock doesn't care about hours                                      |
| Bit 22    | AMPM  | 0x0         | rw   | AM/PM<br>0: AM<br>1: PM<br>Note: This bit is applicable for 12-hour format only. It is 0 for 24-hour format. |
| Bit 21:20 | HT    | 0x0         | rw   | Hour tens  |
| Bit 19:16 | HU    | 0x0         | rw   | Hour units   |
| Bit 15    | MASK2 | 0x0         | rw   | Minute mask<br>0: No minute mask<br>1: Alarm clock doesn't care about minutes                                |
| Bit 14:12 | MT    | 0x0         | rw   | Minute tens  |
| Bit 11:8  | MU    | 0x0         | rw   | Minute units   |
| Bit 7     | MASK1 | 0x0         | rw   | Second mask<br>0: No second mask<br>1: Alarm clock doesn't care about seconds                                |
| Bit 6:4   | ST    | 0x0         | rw   | Second tens  |
| Bit 3:0   | SU    | 0x0         | rw   | Second units   |



## ERTC write protection register (ERTC\_WP)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:8 | Reserved | 0x000000    | resd | Kept at its default value   |
| Bit 7:0  | CMD      | 0x00        | wo   | Command register<br>All ERTC register write protection is unlocked by writing 0xCA and 0x53. Writing any other value will re-activate write protection. |

## ERTC subsecond register (ERTC\_SBS)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value  |
| Bit 15:0  | SBS      | 0x0000      | ro   | Sub-second value<br>Subsecond is the value in the DIVB counter. Clock frequency = $ERTC\_CLK / (DIVA + 1)$ . |

## ERTC time adjustment register (ERTC\_TADJ)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31    | ADD1S    | 0x0         | wo   | Add 1 second<br>0: No effect<br>1: Add one second<br>This bit can be written only when TADJF = 0. It is intended to be used with DECSBS in order to fine-tune the time. |
| Bit 30:15 | Reserved | 0x0000      | resd | Kept at its default value   |
| Bit 14:0  | DECSBS   | 0x0000      | wo   | DECSBS[14:0]: Decrease sub-second value<br>Delay (ADD1S = 0): Delay = $DECSBS / (DIVB + 1)$<br>Advance (ADD1S = 1): Advance = $1 - (DECSBS / (DIVB + 1))$               |

## ERTC time stamp time register (ERTC\_TSTM)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:23 | Reserved | 0x000       | resd | Kept at its default value  |
| Bit 22    | AMPM     | 0x0         | ro   | AM/PM<br>0: AM<br>1: PM<br>Note: This bit is applicable for 12-hour format only. It is 0 for 24-hour format. |
| Bit 21:20 | HT       | 0x0         | ro   | Hour tens  |
| Bit 19:16 | HU       | 0x0         | ro   | Hour units   |
| Bit 15    | Reserved | 0x0         | resd | Kept at its default value  |
| Bit 14:12 | MT       | 0x0         | ro   | Minute tens  |
| Bit 11:8  | MU       | 0x0         | ro   | Minute units   |
| Bit 7     | Reserved | 0x0         | resd | Kept at its default value  |
| Bit 6:4   | ST       | 0x0         | ro   | Second tens  |
| Bit 3:0   | SU       | 0x0         | ro   | Second units   |

Note: The content of this register is valid only when the TSF is set in the ERTC\_STS register. It is cleared when TSF bit is reset.

### ERTC time stamp date register (ERTC\_TSDT)

| Bit       | Name     | Reset value | Type | Description               |
|-----------|----------|-------------|------|---------------------------|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value |
| Bit 15:13 | WK       | 0x0         | ro   | Week day                  |
| Bit 12    | MT       | 0x0         | ro   | Month tens                |
| Bit 11:8  | MU       | 0x0         | ro   | Month units               |
| Bit 7:6   | Reserved | 0x0         | resd | Kept at its default value |
| Bit 5:4   | DT       | 0x0         | ro   | Date tens                 |
| Bit 3:0   | DU       | 0x0         | ro   | Date units                |

Note: The content of this register is valid only when the TSF is set in the ERTC\_STS register. It is cleared when TSF bit is reset.

### ERTC time stamp subsecond register (ERTC\_TSSBS)

| Bit       | Name     | Reset value | Type | Description               |
|-----------|----------|-------------|------|---------------------------|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value |
| Bit 15:0  | SBS      | 0x0000      | ro   | Sub-second value          |

Note: The content of this register is valid only when the TSF is set in the ERTC\_STS register. It is cleared when TSF bit is reset.

### ERTC smooth calibration register (ERTC\_SCAL)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value   |
| Bit 15    | ADD      | 0x0         | rw   | Add ERTC clock<br>0: No ERTC clock added<br>1: One ERTC_CLK is inserted every $2^{11}$ ERTC_CLK cycles  |
| Bit 14    | CAL8     | 0x0         | rw   | 8-second calibration period<br>0: No effect<br>1: 8-second calibration  |
| Bit 13    | CAL16    | 0x0         | rw   | 16 second calibration period<br>0: No effect<br>1: 16-second calibration  |
| Bit 12:9  | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 8:0   | DEC      | 0x000       | rw   | Decrease ERTC clock<br>DEC out of ERTC_CLK cycles are masked during the $2^{20}$ ERTC_CLK periods. This bit is usually used with ADD. When the ADD is set, the actual number of ERTC_CLK is equal to $2^{20} + 512 - DEC$ during the $2^{20}$ ERTC_CLK periods. |

### ERTC tamper configuration register (ERTC\_TAMP)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:19 | Reserved | 0x0000      | resd | Kept at its default value   |
| Bit 18    | OUTTYPE  | 0x0         | rw   | Output type<br>0: Open-drain output<br>1: Push-pull output  |
| Bit 17:16 | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 15    | TPPU     | 0x0         | rw   | Tamper detection pull-up<br>0: Tamper detection pull-up enabled<br>1: Tamper detection pull-up disabled |

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 14:13 | TPPR     | 0x0         | rw   | Tamper detection pre-charge time<br>0: 1 ERTC_CLK cycle<br>1: 2 ERTC_CLK cycles<br>2: 4 ERTC_CLK cycles<br>3: 8 ERTC_CLK cycles   |
| Bit 12:11 | TPFLT    | 0x0         | rw   | Tamper detection filter time<br>0: No filter<br>1: Tamper is detected after 2 consecutive samples<br>2: Tamper is detected after 4 consecutive samples<br>3: Tamper is detected after 8 consecutive samples |
| Bit 10:8  | TPFREQ   | 0x0         | rw   | Tamper detection frequency<br>0: ERTC_CLK/32768<br>1: ERTC_CLK/16384<br>2: ERTC_CLK/8192<br>3: ERTC_CLK/4096<br>4: ERTC_CLK/2048<br>5: ERTC_CLK/1024<br>6: ERTC_CLK/512<br>7: ERTC_CLK/256                  |
| Bit 7     | TPTSEN   | 0x0         | rw   | Tamper detection timestamp enable<br>0: Tamper detection timestamp disabled<br>1: Tamper detection timestamp enabled. Save timestamp on a tamper event.   |
| Bit 6:3   | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 2     | TPIEN    | 0x0         | rw   | Tamper detection interrupt enable<br>0: Tamper detection interrupt disabled<br>1: Tamper detection interrupt enabled  |
| Bit 1     | TP1EDG   | 0x0         | rw   | Tamper detection 1 valid edge<br>If TPFLT = 0:<br>0: Rising edge<br>1: Falling edge<br>If TPFLT > 0:<br>0: Low<br>1: High   |
| Bit 0     | TP1EN    | 0x0         | rw   | Tamper detection 1 enable<br>0: Tamper detection 1 disabled<br>1: Tamper detection 1 enabled  |

### ERTC alarm clock A subsecond register (ERTC\_ALASBS)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:28 | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 27:24 | SBSMSK   | 0x0         | rw   | Sub-second mask<br>0: No comparison. Alarm A doesn't care about subseconds.<br>1: SBS[0] is compared<br>2: SBS[1:0] are compared<br>3: SBS[2:0] are compared<br>...<br>14: SBS[13:0] are compared<br>15: SBS[14:0] are compared |
| Bit 23:15 | Reserved | 0x000       | rw   | Kept at its default value   |
| Bit 14:0  | SBS      | 0x0000      | rw   | Sub-second value  |

### ERTC battery powered domain data register (ERTC\_BPRx)

| Bit      | Name | Reset value | Type | Description  |
|----------|------|-------------|------|--|
| Bit 31:0 | DT   | 0x0000 0000 | rw   | Battery powered domain data<br>BPR_DTx registers are powered on by $V_{BAT}$ so that they are not reset by a system reset. They are reset on a tamper event or when a battery powered domain is reset. |

# 18 Analog-to-digital converter (ADC)

## ADC introduction

The ADC is a peripheral that converts an analog input signal into a 12-bit digital signal. Its sampling rate is as high as 2 MSPS. It has up to 18 channels for sampling and conversion, including 16 external input channels.

## ADC main features

### In terms of analog:

- 12-bit configurable resolution
- Self-calibration time: 154 ADC clock cycles
- ADC conversion time
  - ADC conversion time is 0.5  $\mu$ s at 28 MHz (in 12-bit resolution)
- ADC supply requirement: Refer to HT32F49041 data sheet for more information
- ADC input range:  $V_{REF-} \leq V_{IN} \leq V_{REF+}$

### In terms of digital control:

- Regular channels and preempted channels with different priority
- Regular channels and preempted channels both have their own trigger detection circuit
- Each channel can independently define its own sampling time
- Conversion sequence supports various conversion modes
- Oversampling: hardware oversampling up to 16-bit resolution
- Optional data alignment mode
- Programmable voltage monitor threshold
- Regular channels with DMA transfers
- Interrupt generation at one of the following events:
  - End of the conversion of preempted channels
  - End of the conversion of regular channels
  - Voltage outside the threshold programmed

## ADC structure

Figure 214 shows the block diagram of ADC.

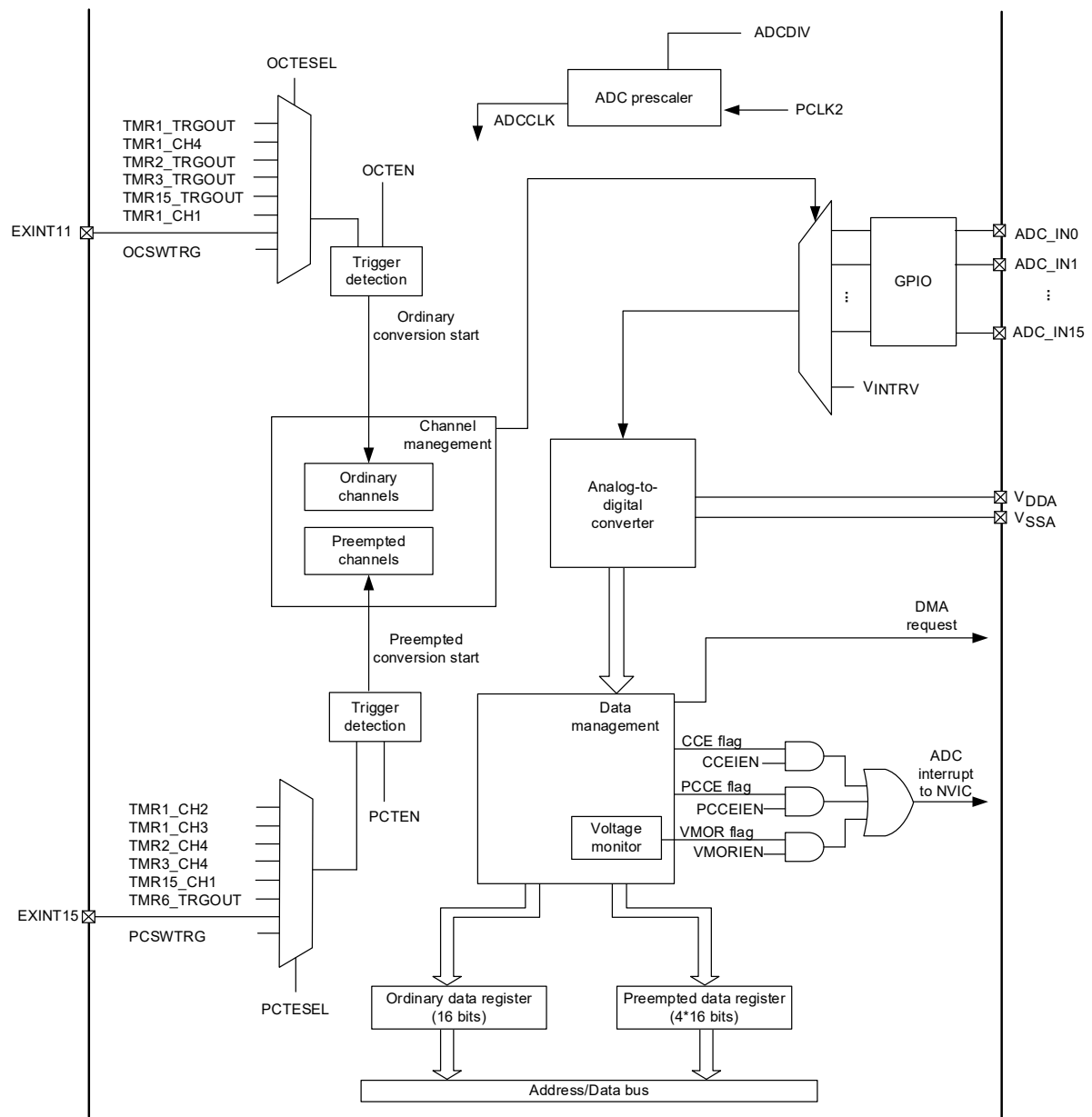


Figure 214. ADC1 block diagram

#### Input pin description:

- $V_{DDA}$ : Analog supply, ADC analog supply
- $V_{SSA}$ : Analog supply ground, ADC analog supply ground
- $ADC\_INx$ : Analog input signal channels

Refer to the HT32F49041 datasheets for more information about the input pin connections and voltage ranges.

## ADC function overview

### Channel management

#### Analog signal channel input:

There are 18 analog signal channel inputs for each of the ADCs, expressed by  $ADC\_INx$  ( $x=0$  to 17).

- $ADC1\_IN0$  to  $ADC1\_IN15$  are referred to as the external analog input,  $ADC1\_IN16$  as  $V_{SSA}$ ,  $ADC1\_IN17$  as an internal reference voltage.

#### Channel conversion

The conversions are divided into two groups: ordinary and preempted channels. The preempted group has priority over the ordinary group.

If the preempted channel trigger occurs during the ordinary channel conversion, then the ordinary channel conversion is interrupted, giving the priority to the preempted channel, and the ordinary channel continues its conversion at the end of the preempted channel conversion. If the ordinary channel trigger occurs during the preempted channel conversion, the ordinary channel conversion won't start until the end of the preempted channel conversion.

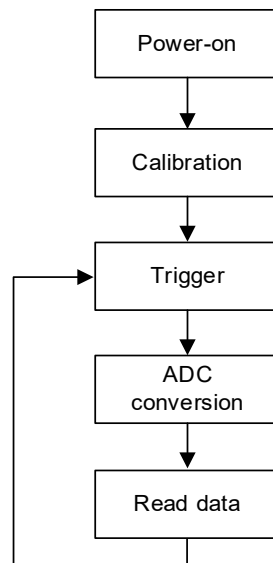
Program the  $ADC\_INx$  into the ordinary channel sequence ( $ADC\_OSQx$ ) and the preempted channel sequence ( $ADC\_PSQ$ ), and the same channel can be repeated, the total number of sequences is determined by  $OCLEN$  and  $PCLEN$ , then it is ready to enable the ordinary channel or preempted channel conversion.

### Internal reference voltage

The internal reference voltage of the typical value 1.2 V is connected to  $ADC1\_IN17$ . It is required to enable the  $ITSRVEN$  bit in the  $ADC\_CTRL2$  register before the internal reference channel conversion. The converted data of such channel can be used to calculate the external reference voltage.

### ADC operation process

Figure 215 shows the basic operation process of the ADC. It is recommended to do the calibration after the initial power-on in order to improve the accuracy of sampling and conversion. After the calibration, trigger is used to enable ADC sampling and conversion. Read data at the end of the conversion.



**Figure 215. ADC basic operation process**

### Power-on and calibration

#### Power-on

Set the ADCxEN bit in the CRM\_APB2EN register to enable ADC clocks: PCLK2 and ADCCLK.

Program the desired ADCCLK frequency by setting the ADCDIV bit in the CRM\_CFG register. The ADCCLK is derived from PCLK2 frequency division.

Note: ADCCLK must be less than 28 MHz.

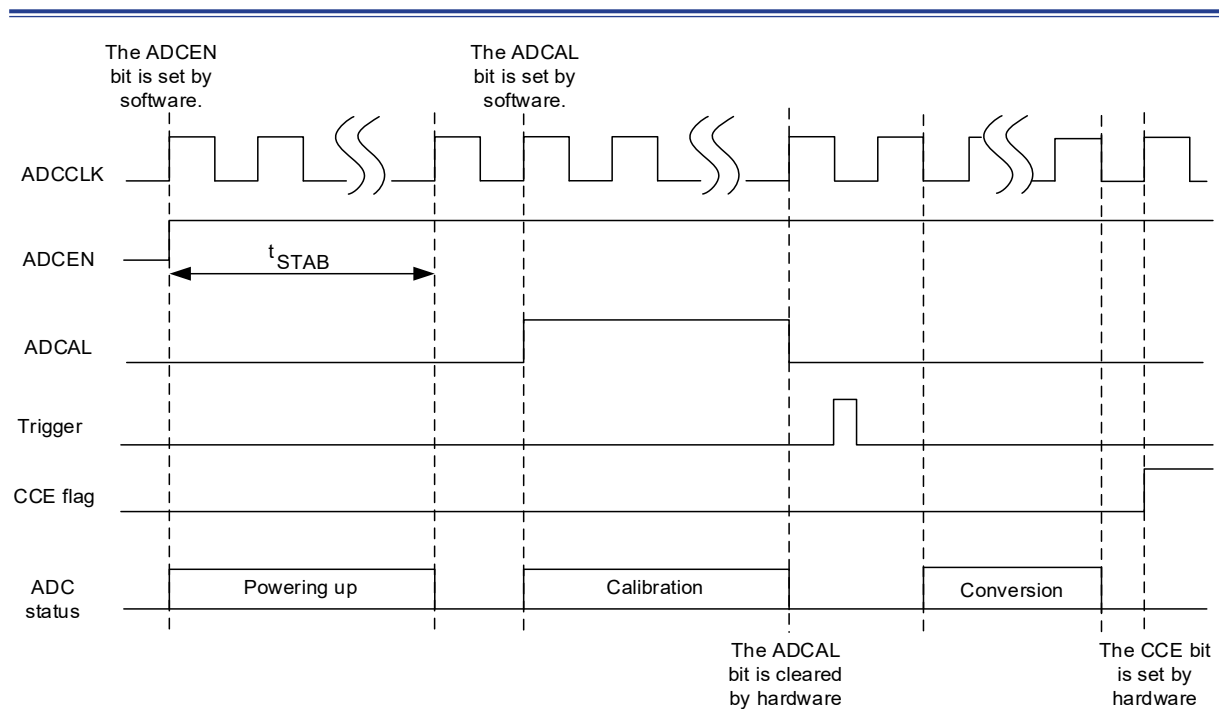
Then set the ADCEN bit in the ADC\_CTRL2 register to supply the ADC, and wait until the RDF flag is set before starting ADC conversion. Clear the ADCEN bit will stop the ADC conversion and result in a reset. In the meantime, the ADC is switched off to save power.

#### Calibration

After power-on, the calibration is enabled by setting the ADCAL bit in the ADC\_CTRL2 register. When the calibration is complete, the ADCAL bit is cleared by hardware and the conversion is performed by software trigger.

After each calibration, the calibration value is stored in ADC\_ODT register, and then value is automatically sent back to the ADC so as to eliminate capacitance errors. The storage of the calibration value will not set the OCCE flag, or generate interrupts or DMA requests.





**Figure 216. ADC power-on and calibration**

## Trigger

The ADC triggers contain ordinary channel trigger and preempted channel trigger. The ordinary channel conversion is triggered by ordinary channel triggers while the preempted channel conversion is triggered by preempted ones. The valid polarity for external trigger sources can be selected by the OCETE and PCETE bits in the ADC\_CTRL2 register. The ADC starts conversion after a trigger source is detected.

The conversion can be triggered by software write operation to the OCSWTRG and PCSWTRG bits in the ADC\_CTRL2 register, or by an external event. The external events include timer and pin triggers. The OCTESEL and PCTESEL bits in the ADC\_CTRL2 register are used to select specific trigger sources, as shown in Table 71.

**Table 71. Trigger sources for ADC**

| OCTESEL | Trigger source            | PCTESEL | Trigger source            |
|---------|---------------------------|---------|---------------------------|
| 000     | TMR1_TRGOUT event         | 000     | TMR1_CH2 event            |
| 001     | TMR1_CH4 event            | 001     | TMR1_CH3 event            |
| 010     | TMR2_TRGOUT event         | 010     | TMR2_CH4 event            |
| 011     | TMR3_TRGOUT event         | 011     | TMR3_CH4 event            |
| 100     | TMR15_TRGOUT event        | 100     | TMR15_CH1 event           |
| 101     | TMR1_CH1 event            | 101     | TMR6_TRGOUT event         |
| 110     | EXINT line11 external pin | 110     | EXINT line15 external pin |
| 111     | OCSWTRG bit               | 111     | PCSWTRG bit               |

## Sampling and conversion sequence

The sampling period can be configured by setting the CSPTx bit in the ADC\_SPT1 and ADC\_SPT2 registers. A single one conversion time is calculated with the following formula:

$$\text{A single one conversion time (ADCCLK period)} = \text{sampling time} + 12.5$$

Example:

If the CSPTx selects 1.5 period, then one conversion needs  $1.5 + 12.5 = 14$  ADCCLK periods

If the CSPTx selects 7.5 period, then one conversion needs  $7.5 + 12.5 = 20$  ADCCLK periods.

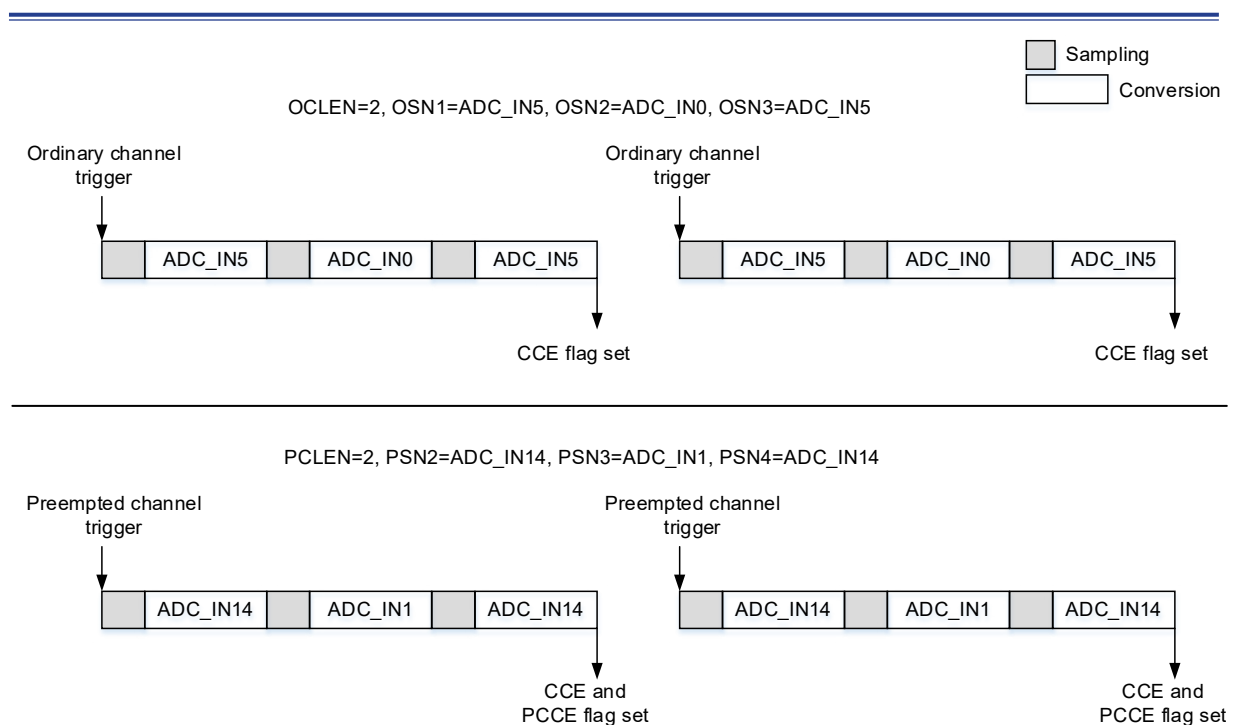
## Conversion sequence management

Only one channel is converted at each trigger event by default, that is, OSN1-defined channel or PSN4-defined channel.

The detailed conversion sequence modes are described in the following sections. With this, the channels can be converted in a specific order.

### Sequence mode

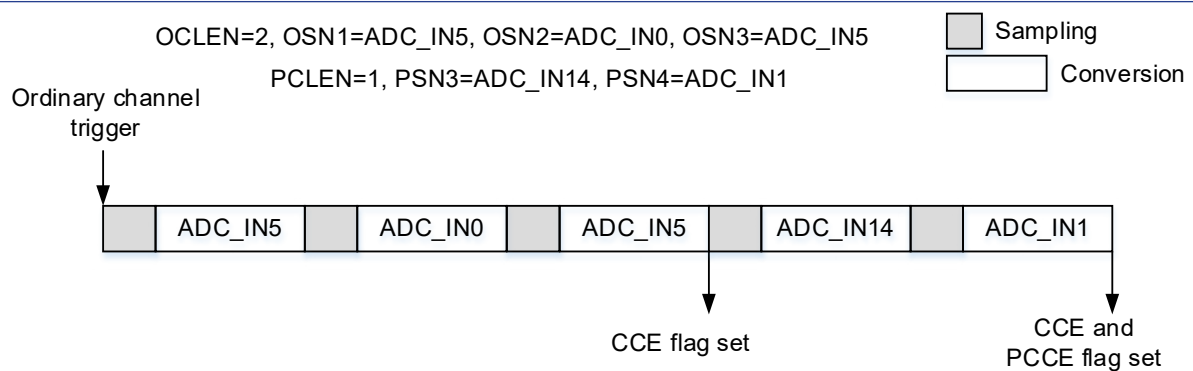
The sequence mode is enabled by setting the SQEN bit in the ADC\_CTRL1 register. The ADC\_OSQx registers are used to configure the sequence and total number of the ordinary channels while the ADC\_PSQ register is used to define the sequence and total number of the preempted channels. When the sequence mode is enabled, a single trigger event enables the conversion of a group of channels in order. The ordinary channels start converting from the QSN1 while the preempted channels starts from the PSNx, where  $x = 4 - \text{PCLEN}$ . Figure 217 shows an example of the behavior in sequence mode.



**Figure 217. Sequence mode**

### Automatic preempted group conversion mode

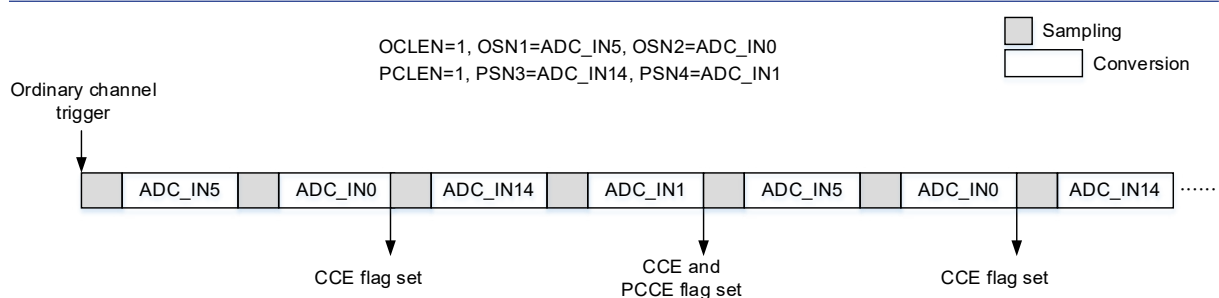
The automatic preempted group conversion mode is enabled by setting the PCAUTOEN bit in the ADC\_CTRL1 register. Once the ordinary channel conversion is over, the preempted group will automatically continue its conversion. This mode can work with the sequence mode. The preempted group conversion starts automatically at the end of the conversion of the ordinary group. Figure 218 shows an example of the behavior when the automatic preempted group conversion mode works with the ordinary group.



**Figure 218. Preempted group auto conversion mode**

### Repetition mode

The repetition mode is enabled by setting the RPEN bit in the ADC\_CTRL2 register. When a trigger signal is detected, the ordinary channels will be converted repeatedly. This mode can work with the ordinary channel conversion in the sequence mode to enable the repeated conversion of the ordinary group. Such mode can also work with the preempted group auto conversion mode to repeatedly convert the ordinary group and preempted group in sequence. Figure 219 shows an example of the behavior when the repetition mode works with the sequence mode and preempted group auto conversion mode.



**Figure 219. Repetition mode**

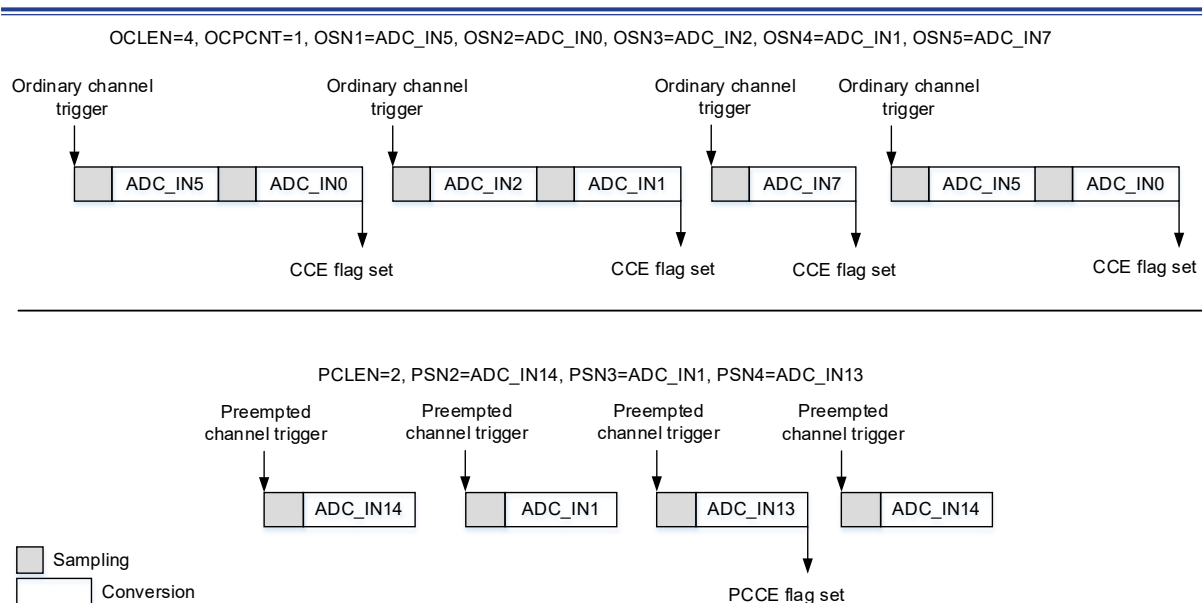
### Partition mode

The partition mode of the ordinary group can be enabled by setting the OCPEN bit in the ADC\_CTRL1 register. In this mode, the ordinary group conversion sequence length (OCLen bit in the ADC\_OSQ1 register) is divided into a smaller sub-group, in which the number of the channels is programmed with the OCPCNT bit in the ADC\_CTRL1 register. A single trigger event will enable

the conversion of all the channels in the sub-group. Each trigger event selects different sub-group in order.

Set the PCPEN bit in the ADC\_CTRL1 register will enable the partition mode of the preempted group. In this mode, the preempted group conversion sequence length (OCLN bit in the ADC\_OSQ1 register) is divided into a sub-group with only one channel. A single one trigger event will enable the conversion of all the channels in the sub-group. Each trigger event selects different sub-group in order.

The partition mode cannot be used with the repetition mode at the same time. Figure 220 shows an example of the behavior in partition mode for ordinary group and preempted group.



**Figure 220. Partition mod**

## Oversampling

A single oversampling converted data can be done through multiple conversions of the same channel in which the cumulative converted data is averaged.

- Oversampling ratio is selected through the OSRSEL bit in the ADC\_OVSP register. This bit is used to specify the oversampling multiple, which is performed by converting the same channel several times.
- Oversampling shift is selected through the OSSSEL bit in the ADC\_OVSP register, which is performed by right shift.

If the averaged data is greater than 16 bits, then only pick up the right-aligned 16-bit data and put them into a 16-bit data register, shown in Table 72.

Example:

If 4x oversampling is selected through the OSRSEL bit, then the same channel is converted by four times in a single oversampling conversion, and the converted data derived from 4 conversions is put together. If 6-bit resolution is selected through the OSSSE bit, then the cumulative data is divided by  $2^6$  and rounded up.

**Table 72. Correlation between maximum cumulative data, oversampling multiple and shift digits**

| Oversampling multiple | 2x     | 4x     | 8x     | 16x    | 32x     | 64x     | 128x    | 256x    |
|-----------------------|--------|--------|--------|--------|---------|---------|---------|---------|
| Max cumulative data   | 0x1FFE | 0x3FFC | 0x7FF8 | 0xFFF0 | 0x1FFE0 | 0x3FFC0 | 0x7FF80 | 0xFFF00 |
| No shift              | 0x1FFE | 0x3FFC | 0x7FF8 | 0xFFF0 | 0xFFE0  | 0xFFC0  | 0xFF80  | 0xFF00  |
| Shift 1 digit         | 0x0FFF | 0x1FFE | 0x3FFC | 0x7FF8 | 0xFFF0  | 0xFFE0  | 0xFFC0  | 0xFF80  |
| Shift 2 digits        | 0x0800 | 0x0FFF | 0x1FFE | 0x3FFC | 0x7FF8  | 0xFFF0  | 0xFFE0  | 0xFFC0  |
| Shift 3 digits        | 0x0400 | 0x0800 | 0x0FFF | 0x1FFE | 0x3FFC  | 0x7FF8  | 0xFFF0  | 0xFFE0  |
| Shift 4 digits        | 0x0200 | 0x0400 | 0x0800 | 0x0FFF | 0x1FFE  | 0x3FFC  | 0x7FF8  | 0xFFF0  |
| Shift 5 digits        | 0x0100 | 0x0200 | 0x0400 | 0x0800 | 0x0FFF  | 0x1FFE  | 0x3FFC  | 0x7FF8  |
| Shift 6 digits        | 0x0080 | 0x0100 | 0x0200 | 0x0400 | 0x0800  | 0x0FFF  | 0x1FFE  | 0x3FFC  |
| Shift 7 digits        | 0x0040 | 0x0080 | 0x0100 | 0x0200 | 0x0400  | 0x0800  | 0x0FFF  | 0x1FFE  |
| Shift 8 digits        | 0x0020 | 0x0040 | 0x0080 | 0x0100 | 0x0200  | 0x0400  | 0x0800  | 0x0FFF  |

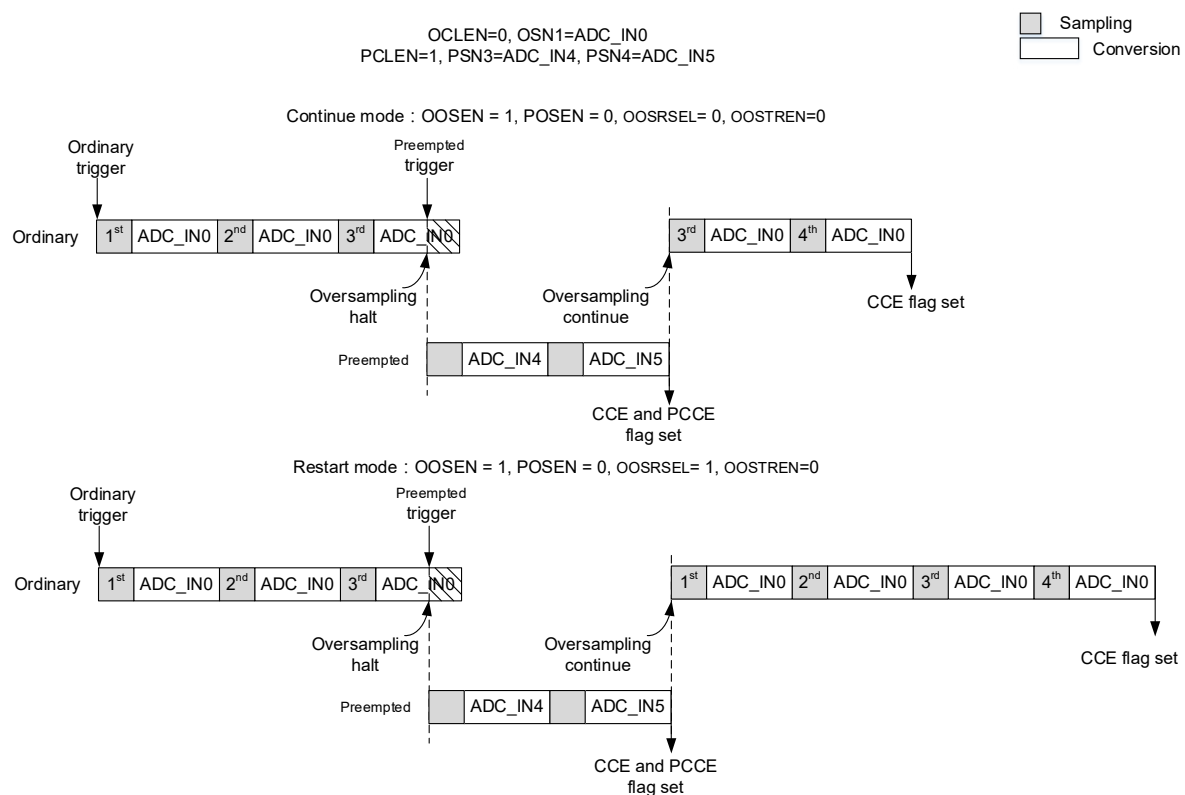
When using oversampling conversion mode, the DTALIGN and PCDTOx are ignored, and data must be right aligned.

#### Oversampling of ordinary group of channels

The OOSRSEL bit in the ADC\_OVSP register can be used to resume ordinary oversampling mode.

- OOSRSEL = 0: continuous conversion mode. Ordinary group of channels, after being interrupted by preempted group of channels during oversampling, will retain the converted data and resume from the last interrupted ordinary conversion.
- OOSRSEL = 1: restart mode. Ordinary group of channels, after being interrupted by preempted group of channels during oversampling, will be reset and restart the ordinary conversion.

Figure 221 shows the differences between ordinary continuous mode and restart mode in 4x oversampling rate and sequential mode.

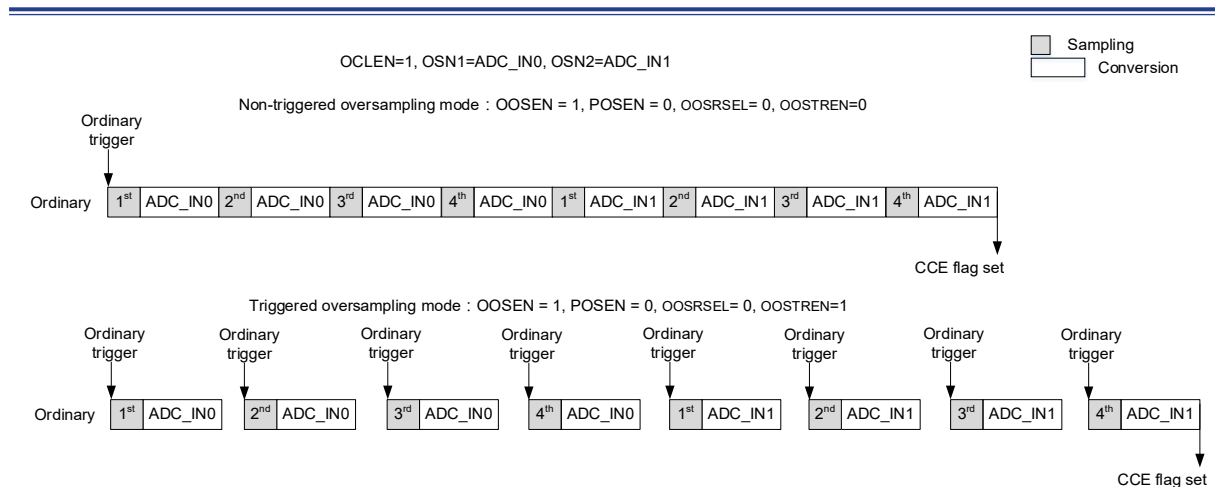


**Figure 221. Ordinary oversampling restart mode selection**

Trigger mode can be enabled by setting the OOSTREN bit in the ADC\_OVSP register. The user must trigger each of the ordinary conversions. In this mode, once the ordinary conversion is interrupted by preempted group of channels, it is necessary to re-trigger ordinary group of channels before resuming the ordinary channels.

When the trigger mode works together with conversion sequence management mode, trigger mode is applied, and the conversion complete flag follows the conversion sequence management mode. Figure 222 shows the behavior when the ordinary trigger mode works together with resume mode in 4x oversampling rate and sequential mode.

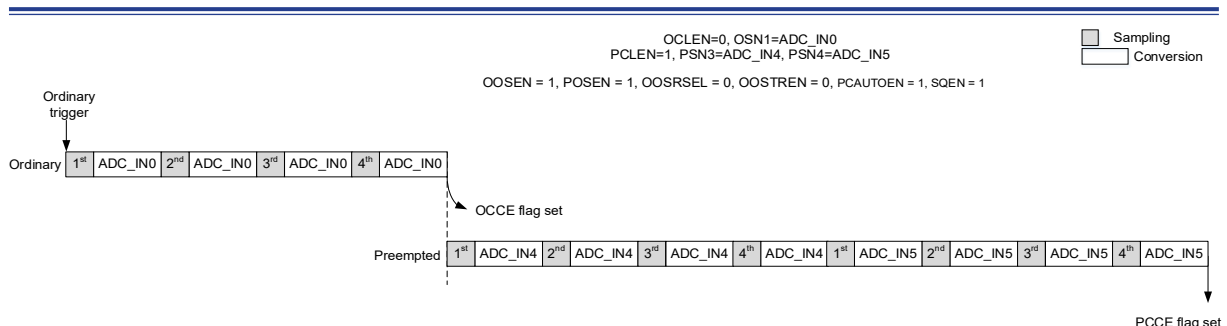
Note: It is not possible to use both the trigger mode and repetition mode simultaneously.



**Figure 222. Ordinary oversampling trigger mode**

### Oversampling of preempted group of channels

It is possible to use both the preempted oversampling and ordinary oversampling simultaneously or individually. The oversampling of the preempted group of channels does not affect the ordinary oversampling modes. Figure 223 shows the behavior when the preempted oversampling and ordinary oversampling trigger mode are used simultaneously in 4x oversampling rate and auto-conversion of preempted group.



**Figure 223. Oversampling of preempted group of channels**

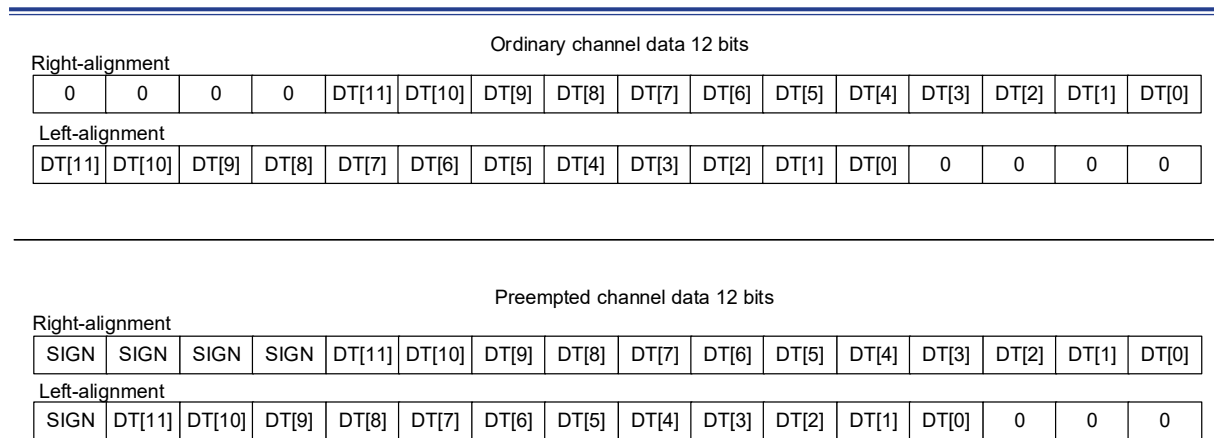
## Data management

At the end of the conversion of the ordinary group, the converted value is stored in the ADC\_ODT register. Once the preempted group conversion ends, the converted data of the preempted group is stored in the ADC\_PDTx register.

### Data alignment

DTALIGN bit in the ADC\_CTRL2 register selects the alignment of data (right-aligned or left-aligned). Apart from this, the converted data of the preempted group is decreased by the offset written in the ADC\_PCDTOx register. Thus the result may be a negative value, marked by SIGN.

The data are aligned on a half-word basis except when the resolution is set to 6-bit. In this case, the data are aligned on a byte basis, as shown in Figure 224.



**Figure 224. Data alignment**

### Data read

Read access to the ADC\_ODT register using CPU or DMA gets the converted data of the ordinary group. Read access to the ADC\_PDTx register using CPU gets the converted data of the preempted group.

When the OCDMAEN bit is set in the ADC\_CTRL2 register, the ADC will issue DMA requests each time the ADC\_OTD register is updated.

### Voltage monitoring

The OCVMEN bit or PCVMEN bit in the ADC\_CTRL1 register is used to enable voltage monitoring based on the converted data.

The VMOR bit will be set if the converted result is outside the high threshold (ADC\_VMHB register) or less than the low threshold (ADC\_VMLB register).

The VMSEGEN bit in the ADC\_CTRL1 register is used to enable voltage monitor on either a single channel or all the channels. The VMSEL bit is used to select a specific channel that requires voltage monitoring.

Voltage monitoring is based on the comparison result between the original converted data and the 12-bit voltage monitor boundary register, irrespective of the CRSEL, PCDTOx and DTALIGN bits.

When using an oversampler, voltage monitoring is based on the comparison result between the 16-bit registers (ADC\_VMHB[15:0] and ADC\_VMLB[15:0]) and the oversampled data.

### Status flag and interrupts

Each of the ADCs has its dedicated ADCx\_STS registers, that is, OCCS (ordinary channel conversion start flag), PCCS (preempted channel conversion start flag), PCCE (preempted channel conversion end flag), OCCE (ordinary channel conversion end flag) and VMOR (voltage monitor out of range).

PCCE, CCE and VMOR have their respective interrupt enable bits. Once the interrupt bits are enabled, the corresponding flag is set and an interrupt is sent to CPU.



## ADC registers

Table 73 lists ADC register map and their reset values.

These peripheral registers must be accessed by word (32 bits).

**Table 73. ADC register map and reset values**

| Register   | Offset | Reset value |
|------------|--------|-------------|
| ADC_STS    | 0x000  | 0x0000 0000 |
| ADC_CTRL1  | 0x004  | 0x0000 0000 |
| ADC_CTRL2  | 0x008  | 0x0000 0000 |
| ADC_SPT1   | 0x00C  | 0x0000 0000 |
| ADC_SPT2   | 0x010  | 0x0000 0000 |
| ADC_PCDTO1 | 0x014  | 0x0000 0000 |
| ADC_PCDTO2 | 0x018  | 0x0000 0000 |
| ADC_PCDTO3 | 0x01C  | 0x0000 0000 |
| ADC_PCDTO4 | 0x020  | 0x0000 0000 |
| ADC_VMHB   | 0x024  | 0x0000 0FFF |
| ADC_VMLB   | 0x028  | 0x0000 0000 |
| ADC_OSQ1   | 0x02C  | 0x0000 0000 |
| ADC_OSQ2   | 0x030  | 0x0000 0000 |
| ADC_OSQ3   | 0x034  | 0x0000 0000 |
| ADC_PSQ    | 0x038  | 0x0000 0000 |
| ADC_PDT1   | 0x03C  | 0x0000 0000 |
| ADC_PDT2   | 0x040  | 0x0000 0000 |
| ADC_PDT3   | 0x044  | 0x0000 0000 |
| ADC_PDT4   | 0x048  | 0x0000 0000 |
| ADC_ODT    | 0x04C  | 0x0000 0000 |
| ADC_OVSP   | 0x080  | 0x0000 0000 |

### ADC status register (ADC\_STS)

Accessed by words.

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31:5 | Reserved | 0x0000000   | resd | Kept at its default value.   |
| Bit 4    | OCCS     | 0x0         | rw0c | Ordinary channel conversion start flag<br>This bit is set by hardware and cleared by software (writing 0).<br>0: No ordinary channel conversion started<br>1: Ordinary channel conversion has started    |
| Bit 3    | PCCS     | 0x0         | rw0c | Preempted channel conversion start flag<br>This bit is set by hardware and cleared by software (writing 0).<br>0: No preempted channel conversion started<br>1: Preempted channel conversion has started |
| Bit 2    | PCCE     | 0x0         | rw0c | Preempted channel end of conversion flag<br>This bit is set by hardware and cleared by software (writing 0).<br>0: Conversion is not complete<br>1: Conversion is complete                               |

| Bit   | Name | Reset value | Type | Description  |
|-------|------|-------------|------|--|
| Bit 1 | OCCE | 0x0         | rw0c | End of conversion flag<br>This bit is set by hardware. It is cleared by software (writing 0) or by reading the ADC_ODT register.<br>0: Conversion is not complete<br>1: Conversion is complete<br>Note: This bit is set at the end of the ordinary or preempted group. |
| Bit 0 | VMOR | 0x0         | rw0c | Voltage monitoring out of range flag<br>This bit is set by hardware and cleared by software (writing 0).<br>0: Voltage is within the value programmed<br>1: Voltage is outside the value programmed  |

### ADC control register1 (ADC\_CTRL1)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:24 | Reserved | 0x00        | resd | Kept at its default value.   |
| Bit 23    | OCVMEN   | 0x0         | rw   | Voltage monitoring enable on ordinary channels<br>0: Voltage monitoring disabled on ordinary channels<br>1: Voltage monitoring enabled on ordinary channels  |
| Bit 22    | PCVMEN   | 0x0         | rw   | Voltage monitoring enable on preempted channels<br>0: Voltage monitoring disabled on preempted channels<br>1: Voltage monitoring enabled on preempted channels   |
| Bit 21:16 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 15:13 | OCPCNT   | 0x0         | rw   | Partitioned mode conversion count of ordinary channels<br>000: 1 channel<br>001: 2 channels<br>.....<br>111: 8 channels<br>Note: In this mode, the preempted group converts only one channel at each trigger.  |
| Bit 12    | PCPEN    | 0x0         | rw   | Partitioned mode enable on preempted channels<br>0: Partitioned mode disabled on preempted channels<br>1: Partitioned mode enabled on preempted channels   |
| Bit 11    | OCPEN    | 0x0         | rw   | Partitioned mode enable on ordinary channels<br>This is set and cleared by software to enable or disable partitioned mode on ordinary channels.<br>0: Partitioned mode disabled on ordinary channels<br>1: Partitioned mode enabled on ordinary channels |
| Bit 10    | PCAUTOEN | 0x0         | rw   | Preempted group automatic conversion enable after ordinary group<br>0: Preempted group automatic conversion disabled<br>1: Preempted group automatic conversion enabled  |
| Bit 9     | VMSGEN   | 0x0         | rw   | Voltage monitoring enable on a single channel<br>0: Disabled (Voltage monitoring enabled on all channels)<br>1: Enabled (Voltage monitoring enabled a single channel)  |

| Bit     | Name    | Reset value | Type | Description  |
|---------|---------|-------------|------|--|
| Bit 8   | SQEN    | 0x0         | rw   | Sequence mode enable<br>0: Sequence mode disabled (a single channel is converted)<br>1: Sequence mode enabled (the selected multiple channels are converted)<br>Note: If this mode is enabled and the CCEIEN/PCCEIEN is set, a CCE or PCCE interrupt is generated only at the end of conversion of the last channel. |
| Bit 7   | PCCEIEN | 0x0         | rw   | Conversion end interrupt enable on Preempted channels<br>0: Conversion end interrupt disabled on Preempted channels<br>1: Conversion end interrupt enabled on Preempted channels   |
| Bit 6   | VMORIEN | 0x0         | rw   | Voltage monitoring out of range interrupt enable<br>0: Voltage monitoring out of range interrupt disabled<br>1: Voltage monitoring out of range interrupt enabled  |
| Bit 5   | CCEIEN  | 0x0         | rw   | Channel conversion end interrupt enable<br>0: Channel conversion end interrupt disabled<br>1: Channel conversion end interrupt enabled   |
| Bit 4:0 | VMCSEL  | 0x00        | rw   | Voltage monitoring channel select<br>This field is valid only when the VMSGEN is enabled.<br>00000: ADC_IN0 channel<br>00001: ADC_IN1 channel<br>.....<br>01111: ADC_IN15 channel<br>10000: ADC_IN16 channel<br>10001: ADC_IN17 channel<br>10010 ~ 11111: Unused, configuration is not allowed.                      |

## ADC control register2 (ADC\_CTRL2)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 30:24 | Reserved | 0x00        | resd | Kept at its default value  |
| Bit 23    | ITSRVEN  | 0x0         | rw   | Internal V <sub>INTRV</sub> enable<br>0: Internal V <sub>INTRV</sub> disabled<br>1: Internal V <sub>INTRV</sub> enabled  |
| Bit 22    | OCSWTRG  | 0x0         | rw   | Conversion of ordinary channels triggered by software<br>0: Conversion of ordinary channels not triggered<br>1: Conversion of ordinary channels triggered (This bit is cleared by software or by hardware as soon as the conversion starts)    |
| Bit 21    | PCSWTRG  | 0x0         | rw   | Conversion of preempted channels triggered by software<br>0: Conversion of preempted channels not triggered<br>1: Conversion of preempted channels triggered (This bit is cleared by software or by hardware as soon as the conversion starts) |
| Bit 20    | OCTEN    | 0x0         | rw   | Trigger mode enable for ordinary channel conversion<br>0: Disabled<br>1: Enabled   |

| Bit       | Name      | Reset value | Type | Description  |
|-----------|-----------|-------------|------|--|
| Bit 19:17 | OCTESEL   | 0x0         | rw   | Trigger event select for ordinary channel conversion<br>000: TMR1 TRGOUT event<br>001: TMR1 CH4 event<br>010: TMR 2 TRGOUT event<br>011: TMR 3 TRGOUT event<br>100: TMR 15 TRGOUT event<br>101: TMR 1 CH1 event<br>110: EXINT11<br>111: OCSWTRG  |
| Bit 16    | Reserved  | 0x0         | resd | Kept at its default value  |
| Bit 15    | PCTEN     | 0x0         | rw   | Trigger mode enable for preempted channels conversion<br>0: Disabled<br>1: Enabled   |
| Bit 14:12 | PCTESEL   | 0x0         | rw   | Trigger event select for preempted channel conversion<br>000: TMR1 CH2 event<br>001: TMR1 CH3 event<br>010: TMR 2 CH4 event<br>011: TMR 3 CH4 event<br>100: TMR 15 CH1 event<br>101: TMR 6 TRGOUT event<br>110: EXINT15  |
| Bit 11    | DTALIGN   | 0x0         | rw   | Data alignment<br>0: Right alignment<br>1: Left alignment  |
| Bit 10:9  | Reserved  | 0x0         | resd | Kept at its default value  |
| Bit 8     | OCDMAEN   | 0x0         | rw   | DMA transfer enable of ordinary channels<br>0: Disabled<br>1: Enabled  |
| Bit 7:4   | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 3     | ADCALINIT | 0x0         | rw   | Initialize A/D calibration<br>This bit is set by software and cleared by hardware. It is cleared after the calibration registers are initialized.<br>0: No initialization occurred or initialization completed<br>1: Enable initialization or initialization is ongoing  |
| Bit 2     | ADCAL     | 0x0         | rw   | A/D Calibration<br>0: No calibration occurred or calibration completed<br>1: Enable calibration or calibration is in process   |
| Bit 1     | RPEN      | 0x0         | rw   | Repetition mode enable<br>0: Repetition mode disabled<br>When SQEN = 0, a single conversion is done each time when a trigger event arrives; when SQEN = 1, a group of conversion is done each time when a trigger event arrives.<br>1: Repetition mode enabled<br>When SQEN = 0, continuous conversion mode on a single channel is enabled at each trigger event; when SQEN = 1, continuous conversion mode on a group of channels is enabled at each trigger event. |

| Bit   | Name  | Reset value | Type | Description   |
|-------|-------|-------------|------|---|
| Bit 0 | ADCEN | 0x0         | rw   | <p>A/D converter enable</p> <p>0: A/D converter disabled (ADC goes to power-down mode)</p> <p>1: A/D converter enabled</p> <p>Note:</p> <p>When this bit is in OFF state, write an ON command can wake up The ADC from power-down mode.</p> <p>When this bit in ON state, write an ON command repeatedly while other bits of the register remain unchanged can start a regular group conversion.</p> <p>The application should pay attention to the fact that there is a delay of <math>t_{STAB}</math> between power on and start of conversion.</p> |

### ADC sampling time register 1 (ADC\_SPT1)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:24 | Reserved | 0x00        | resd | Kept at its default value.  |
| Bit 23:21 | CSPT17   | 0x0         | rw   | <p>Sample time selection of channel ADC_IN17</p> <p>000: 1.5 cycles</p> <p>001: 7.5 cycles</p> <p>010: 13.5 cycles</p> <p>011: 28.5 cycles</p> <p>100: 41.5 cycles</p> <p>101: 55.5 cycles</p> <p>110: 71.5 cycles</p> <p>111: 239.5 cycles</p> |
| Bit 20:18 | CSPT16   | 0x0         | rw   | <p>Sample time selection of channel ADC_IN16</p> <p>000: 1.5 cycles</p> <p>001: 7.5 cycles</p> <p>010: 13.5 cycles</p> <p>011: 28.5 cycles</p> <p>100: 41.5 cycles</p> <p>101: 55.5 cycles</p> <p>110: 71.5 cycles</p> <p>111: 239.5 cycles</p> |
| Bit 17:15 | CSPT15   | 0x0         | rw   | <p>Sample time selection of channel ADC_IN15</p> <p>000: 1.5 cycles</p> <p>001: 7.5 cycles</p> <p>010: 13.5 cycles</p> <p>011: 28.5 cycles</p> <p>100: 41.5 cycles</p> <p>101: 55.5 cycles</p> <p>110: 71.5 cycles</p> <p>111: 239.5 cycles</p> |
| Bit 14:12 | CSPT14   | 0x0         | rw   | <p>Sample time selection of channel ADC_IN14</p> <p>000: 1.5 cycles</p> <p>001: 7.5 cycles</p> <p>010: 13.5 cycles</p> <p>011: 28.5 cycles</p> <p>100: 41.5 cycles</p> <p>101: 55.5 cycles</p> <p>110: 71.5 cycles</p> <p>111: 239.5 cycles</p> |

| Bit      | Name   | Reset value | Type | Description  |
|----------|--------|-------------|------|--|
| Bit 11:9 | CSPT13 | 0x0         | rw   | Sample time selection of channel ADC_IN13<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 8:6  | CSPT12 | 0x0         | rw   | Sample time selection of channel ADC_IN12<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 5:3  | CSPT11 | 0x0         | rw   | Sample time selection of channel ADC_IN11<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 2:0  | CSPT10 | 0x0         | rw   | Sample time selection of channel ADC_IN10<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |

### ADC sampling time register 2 (ADC\_SPT2)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:30 | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 29:27 | CSPT9    | 0x0         | rw   | Sample time selection of channel ADC_IN9<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |

| Bit       | Name  | Reset value | Type | Description   |
|-----------|-------|-------------|------|---|
| Bit 26:24 | CSPT8 | 0x0         | rw   | Sample time selection of channel ADC_IN8<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 23:21 | CSPT7 | 0x0         | rw   | Sample time selection of channel ADC_IN7<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 20:18 | CSPT6 | 0x0         | rw   | Sample time selection of channel ADC_IN6<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 17:15 | CSPT5 | 0x0         | rw   | Sample time selection of channel ADC_IN5<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 14:12 | CSPT4 | 0x0         | rw   | Sample time selection of channel ADC_IN4<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 11:9  | CSPT3 | 0x0         | rw   | Sample time selection of channel ADC_IN3<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |

| Bit     | Name  | Reset value | Type | Description   |
|---------|-------|-------------|------|---|
| Bit 8:6 | CSPT2 | 0x0         | rw   | Sample time selection of channel ADC_IN2<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 5:3 | CSPT1 | 0x0         | rw   | Sample time selection of channel ADC_IN1<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |
| Bit 2:0 | CSPT0 | 0x0         | rw   | Sample time selection of channel ADC_IN0<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |

### ADC preempted channel data offset register x (ADC\_PCDTOx) (x = 1..4)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:12 | Reserved | 0x00000     | resd | Kept at its default value  |
| Bit 11:0  | PCDTOx   | 0x000       | rw   | Data offset for Preempted channel x<br>Converted data stored in the ADC_PDTx = Raw converted data – ADC_PCDTOx |

### ADC voltage monitor high threshold register (ADC\_VWHB)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description                      |
|-----------|----------|-------------|------|----------------------------------|
| Bit 31:16 | Reserved | 0x00000     | resd | Kept at its default value        |
| Bit 15:0  | VMHB     | 0xFFFF      | rw   | Voltage monitoring high boundary |

### ADC voltage monitor low threshold register (ADC\_VWLB)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description                     |
|-----------|----------|-------------|------|---------------------------------|
| Bit 31:12 | Reserved | 0x00000     | resd | Kept at its default value       |
| Bit 11:0  | VMLB     | 0x000       | rw   | Voltage monitoring low boundary |



## ADC ordinary sequence register 1 (ADC\_OSQ1)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:24 | Reserved | 0x00        | resd | Kept at its default value   |
| Bit 23:20 | OCLEN    | 0x0         | rw   | Ordinary conversion sequence length<br>0000: 1 conversion<br>0001: 2 conversions<br>.....<br>1111: 16 conversions |
| Bit 19:15 | OSN16    | 0x00        | rw   | Number of 16th conversion in ordinary sequence  |
| Bit 14:10 | OSN15    | 0x00        | rw   | Number of 15th conversion in ordinary sequence  |
| Bit 9:5   | OSN14    | 0x00        | rw   | Number of 14th conversion in ordinary sequence  |
| Bit 4:0   | OSN13    | 0x00        | rw   | Number of 13th conversion in ordinary sequence  |

Note: The number can be from 0 to 17. For example, if the number is set to 3, it means that the 13<sup>th</sup> conversion is ADC\_IN3 channel.

## ADC ordinary sequence register 2 (ADC\_OSQ2)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description                                    |
|-----------|----------|-------------|------|--|
| Bit 31:30 | Reserved | 0x0         | resd | Kept at its default value                      |
| Bit 29:25 | OSN12    | 0x00        | rw   | Number of 12th conversion in ordinary sequence |
| Bit 24:20 | OSN11    | 0x00        | rw   | Number of 11th conversion in ordinary sequence |
| Bit 19:15 | OSN10    | 0x00        | rw   | Number of 10th conversion in ordinary sequence |
| Bit 14:10 | OSN9     | 0x00        | rw   | Number of 9th conversion in ordinary sequence  |
| Bit 9:5   | OSN8     | 0x00        | rw   | Number of 8th conversion in ordinary sequence  |
| Bit 4:0   | OSN7     | 0x00        | rw   | Number of 7th conversion in ordinary sequence  |

Note: The number can be from 0 to 17. For example, if the number is set to 8, it means that the 7<sup>th</sup> conversion is ADC\_IN8 channel.

## ADC ordinary sequence register 3 (ADC\_OSQ3)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description                                   |
|-----------|----------|-------------|------|---|
| Bit 31:30 | Reserved | 0x0         | resd | Kept at its default value                     |
| Bit 29:25 | OSN6     | 0x00        | rw   | Number of 6th conversion in ordinary sequence |
| Bit 24:20 | OSN5     | 0x00        | rw   | Number of 5th conversion in ordinary sequence |
| Bit 19:15 | OSN4     | 0x00        | rw   | Number of 4th conversion in ordinary sequence |
| Bit 14:10 | OSN3     | 0x00        | rw   | Number of 3rd conversion in ordinary sequence |
| Bit 9:5   | OSN2     | 0x00        | rw   | Number of 2nd conversion in ordinary sequence |
| Bit 4:0   | OSN1     | 0x00        | rw   | Number of 1st conversion in ordinary sequence |

Note: The number can be from 0 to 17. For example, if the number is set to 8, it means that the 1st conversion is ADC\_IN17 channel.

## ADC preempted sequence register (ADC\_PSQ)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:30 | Reserved | 0x0         | resd | Kept at its default value   |
| Bit 21:20 | PCLEN    | 0x0         | rw   | Preempted conversion sequence length<br>00: 1 conversion<br>01: 2 conversions<br>10: 3 conversions<br>11: 4 conversions |
| Bit 19:15 | PSN4     | 0x00        | rw   | Number of 4th conversion in preempted sequence  |
| Bit 14:10 | PSN3     | 0x00        | rw   | Number of 3rd conversion in preempted sequence  |
| Bit 9:5   | PSN2     | 0x00        | rw   | Number of 2nd conversion in preempted sequence  |
| Bit 4:0   | PSN1     | 0x00        | rw   | Number of 1st conversion in preempted sequence  |

Note: The number can be from 0 to 17. For example, if the number is set to 3, it refers to the ADC\_IN3 channel.

If PCLEN is less than 4, the conversion sequence starts from 4-PCLEN. For example, when ADC\_PSQ ([21:0]) = 10 00110 00101 00100 00011, it indicates that the scan conversion follows the sequence: 4, 5, 6, not 3, 4, 5.

## ADC preempted data register x (ADC\_PDTx) (x = 1..4)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description                            |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value              |
| Bit 15:0  | PDTx     | 0x0000      | rw   | Conversion data from preempted channel |

## ADC ordinary data register (ADC\_ODT)

Accessed by words.

| Bit       | Name    | Reset value | Type | Description  |
|-----------|---------|-------------|------|--|
| Bit 31:16 | ADC2ODT | 0x0000      | ro   | ADC2 conversion data of ordinary channel<br>Note:<br>These bits are reserved in ADC2 and ADC3.<br>In ADC1, these bits are valid only in master/slave mode, and they contain the conversion result from the ADC2 ordinary channels. |
| Bit 15:0  | ODT     | 0x0000      | ro   | Conversion data of ordinary channel  |

## ADC oversampling register (ADC\_OVSP)

Accessed by words.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:11 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 10    | OOSRSEL  | 0x0         | rw   | Ordinary oversampling restart mode select<br>When the ordinary oversampling is interrupted by preempted conversions, this bit can be used to select where to resume ordinary conversions.<br>0: Continuous mode (ordinary oversampling buffer will be reserved)<br>1: Restart mode (ordinary oversampling buffer will be cleared, that is, the previously oversampled times are reset) |
| Bit 9     | OOSTREN  | 0x0         | rw   | Ordinary oversampling trigger mode enable<br>0: Disabled (only one trigger is needed for all oversampling conversions)<br>1: Enabled (Each oversampling conversion needs a trigger)  |
| Bit 8:5   | OSSSEL   | 0x0         | rw   | Oversampling shift select<br>This field is used to define the number of right-shift used in the oversampling results.<br>0000: No shift<br>0001: 1 bit<br>0010: 2 bits<br>0011: 3 bits<br>0100: 4 bits<br>0101: 5 bits<br>0110: 6 bits<br>0111: 7 bits<br>1000: 8 bits<br>1001 ~ 1111: Unused. Do not configure.   |
| Bit 4:2   | OSRSEL   | 0x0         | rw   | Oversampling ratio select<br>000: 2x<br>001: 4x<br>010: 8x<br>011: 16x<br>100: 32x<br>101: 64x<br>110: 128x<br>111: 256x   |
| Bit 1     | POSEN    | 0x0         | rw   | Preempted oversampling enable<br>0: Preempted oversampling disabled<br>1: Preempted oversampling enabled   |
| Bit 0     | OOKEN    | 0x0         | rw   | Ordinary oversampling enable<br>0: Ordinary oversampling disabled<br>1: Ordinary oversampling enabled  |

# 19 Controller area network (CAN)

## CAN introduction

CAN (Controller Area Network) is a distributed serial communication protocol for real-time and reliable data communication among various nodes. It supports the CAN protocol version 2.0A and 2.0B.

## CAN main features

- Baud rates up to 1 Mbit/s
- Supports the time triggered communication
- Interrupt enable and mask
- Configurable automatic retransmission mode

### Transmission

- Three transmit mailboxes
- Configurable transmit priority
- Supports the time stamp on transmission

### Reception

- Two FIFOs with three-level depth
- 14 filter banks
- Supports the identifier list mode
- Supports the identifier mask mode
- FIFO overrun management

### Time triggered communication mode

- 16-bit timers
- Time stamp on transmission

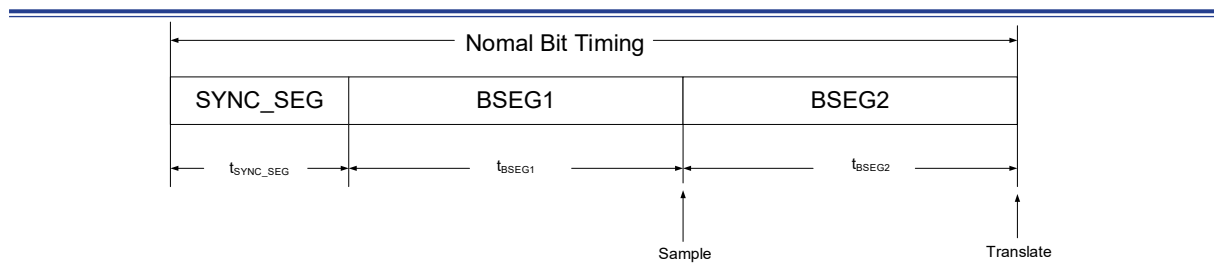
## Baud rate

The nominal bit time of the CAN bus consists of three parts as follows:

Synchronization segment (SYNC\_SEG): This segment has one time unit, and its time duration is defined by the BRDIV[11:0] bit in the CAN\_BTMTG register.

Bit segment 1 (BIT SEGMENT 1): It is referred to as BSEG1 including the PROP\_SEG and PHASE\_SEG1 of the CAN standard. Its duration is between 1 and 16 time units, defined by the BTS1[3:0] bit.

Big segment 2 (BIT SEGMENT 2): It is referred to as BSEG2 including the PHASE\_SEG2 of the CAN standard. Its duration is between 1 and 8 time units, defined by the BTS2[2:0] bit.



**Figure 225. Bit timing**

**Baud rate formula:**

$$\text{BaudRate} = \frac{1}{\text{Nomal Bit Timing}}$$

$$\text{Nomal Bit Timing} = t_{\text{SYNC\_SEG}} + t_{\text{BSEG1}} + t_{\text{BSEG2}}$$

where

$$t_{\text{SYNC\_SEG}} = 1 \times t_q$$

$$t_{\text{BSEG1}} = (1 + \text{BTS1}[3:0]) \times t_q$$

$$t_{\text{BSEG2}} = (1 + \text{BTS2}[2:0]) \times t_q$$

$$t_q = (1 + \text{BRDIV}[11:0]) \times t_{\text{pclk}}$$

**Hard synchronization and resynchronization**

The start location of each bit in CAN nodes is always in synchronization segment by default, and the sampling is performed at the edge location of bit segment 1 and big segment 2 simultaneously.

During the actual transmission, each bit of the CAN nodes has certain phase error due to the oscillator drift, transmission delay among the network nodes and noise interference. To avoid the impact on the communication, the start-bit edge and its subsequent falling edge can be synchronized or resynchronized. The time length of the synchronization compensation cannot be greater than the resynchronization width (1 to 4 time units, defined by the RSAW[1:0] bit).

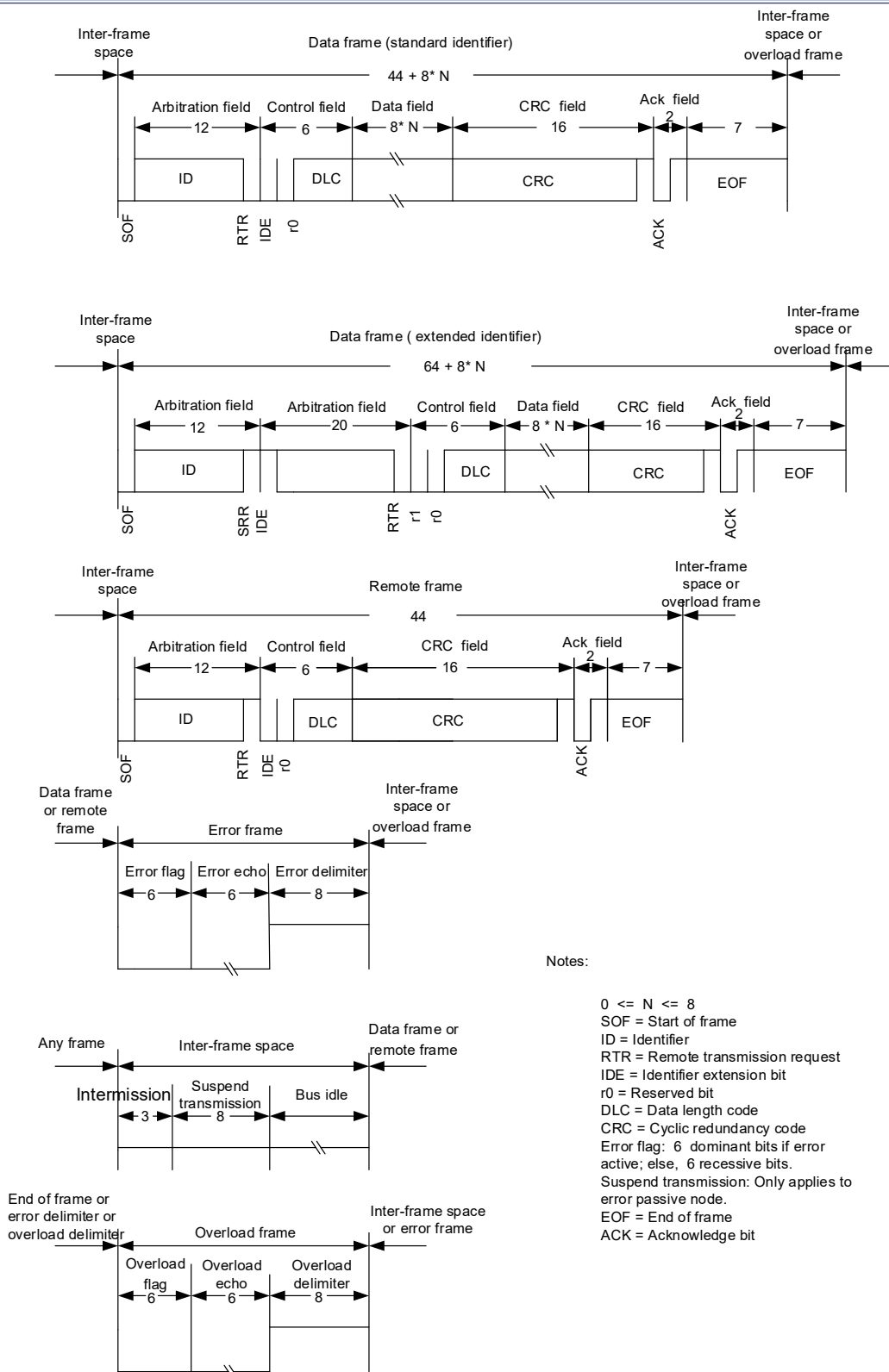
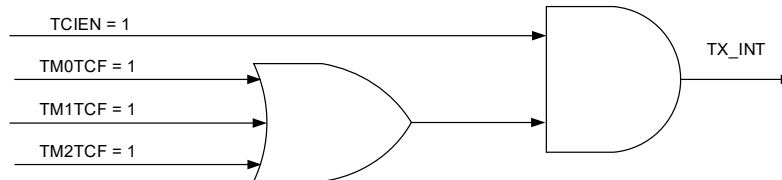


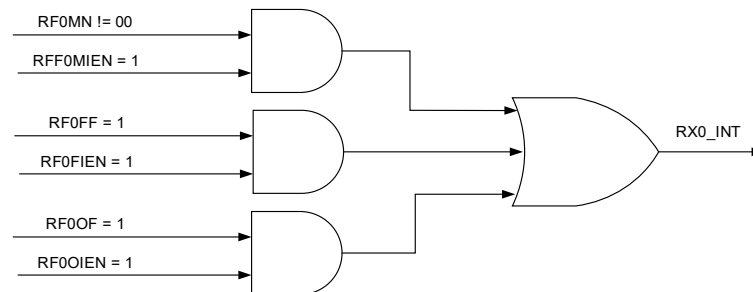
Figure 226. Frame type

## Interrupt management

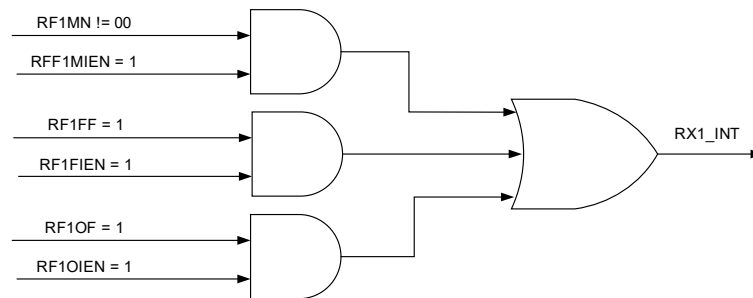
The CAN controller contains four interrupt vectors that can be used to enable or disable interrupts by setting the CAN\_INTEN register.



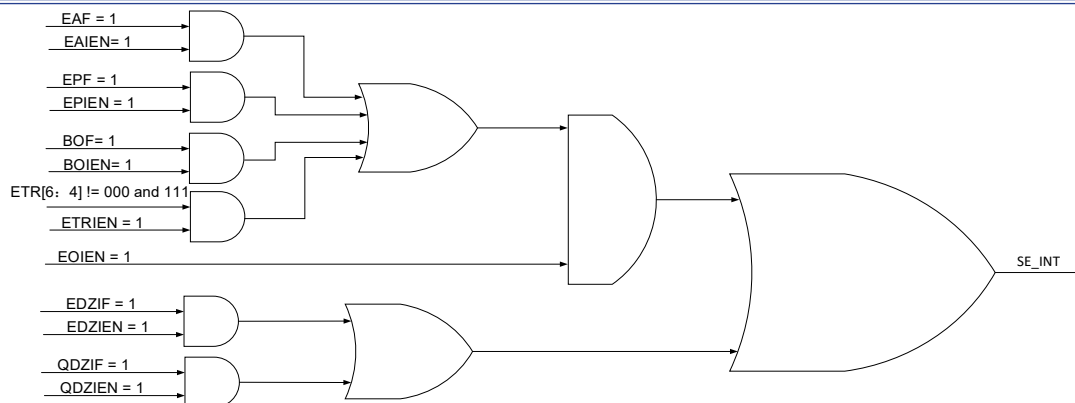
**Figure 227. Transmit interrupt generation**



**Figure 228. Receive interrupt 0 generation**



**Figure 229. Receive interrupt 1 generation**



**Figure 230. Status error interrupt generation**

## Design tips

The following information can be used as reference for CAN application development:

### ■ Debug control

When the system enters the debug mode, the CAN controller stops or continues to work normally, depending on the CANx\_PAUSE bit in the DEBUG\_CTRL register or the PTD bit in the CAN\_MCTRL register.

### ■ Time triggered communication

The timer triggered communication is used to improve the real-time performance so as to avoid bus competition. It is activated by setting TTCEN = 1 in the CAN\_MCTRL register. The internal 16-bit timer is incremented each CAN bit time, and is sampled on the Start Of Frame bit to generate the time stamp value, which is stored in the CAN\_RFCx and CAN\_TMCx register.

### ■ Register access protection

The CAN\_BTMG register can be modified only when the CAN is in frozen mode.

Although the transmission of incorrect data will not cause problems at the network level, it can have severe impact on the application. Thus a transmit mailbox can be modified only when it is in empty state. The filter configuration in the CAN\_FMCFG, CAN\_FBWCFG and CAN\_FRF registers can be modified only when FCS = 1. The CAN\_FiFBx register can be modified only when FCS = 1 or FAENx = 0.

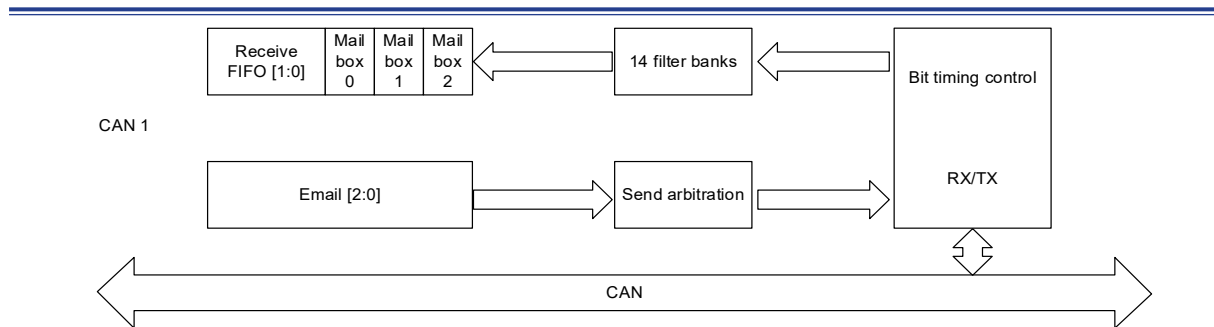
## Function overview

### General description

As the number of nodes in the CAN network and the number of messages grows, an enhanced filtering mechanism is required to handle all types of messages in order to reduce the processing time of message reception. One FIFO scheme is used to ensure that the CPU can concentrate on application tasks for a long period of time without the loss of messages. In the meantime, the priority order of the messages to be transmitted is configured by hardware. Standard identifiers (11-bit) and extended identifiers (29-bit) are fully supported by hardware.



Based on the above mentioned conditions, the CAN controller provides 14 scalable/configurable identifier filter banks, 2 receive FIFOs with storing 3 complete messages each and being totally managed by hardware, and 3 transmit mailboxes with their transmit priority order defined by the transmit scheduler.



**Figure 231. CAN block diagram**

## Operating modes

The CAN controller has three operating modes:

### Sleep mode

After a system reset, the CAN controller is in Sleep mode. In this mode, the CAN clock is stopped to reduce power consumption and an internal pull-up resistance is disabled. However, the software can still access to the mailbox registers.

The software request the CAN controller to enter Sleep mode by setting the DZEN bit in the CAN\_MCTRL register. The hardware confirms the request by setting the DZC bit in the CAN\_MSTS register.

Exit Sleep mode in two ways: The CAN controller can be woke up by hardware clearing the DZEN bit when the AEDEN bit in the CAN\_MCTRL register and the CAN bus activity is detected. It can also be woke up by software clearing the DZEN bit.

Switch to Frozen mode: The CAN controller switches from Sleep mode to Frozen mode when the FZEN bit is set in the CAN\_MCTRL register and the DZEN bit is cleared. Such switch operation is confirmed by hardware setting the FZC bit in the CAN\_MSTS register.

Switch to Communication mode: The CAN controller enters Communication mode when the FZEN and DZEN bits are both cleared and the CAN controller has synchronized with the bus. In other words, it must wait for 11 consecutive recessive bits to be detected on the CANRX pin.

### Frozen mode

The software initialization can be done only in Frozen mode, including the CAN\_BTMG and CAN\_MCTRL registers. But the initialization of the 14 CAN filter banks (mode, scale, FIFO association, activation and filter values) can be done in non-Frozen mode. When the CAN controller is in Frozen mode, message reception and transmission are both disabled.

Switch to Communication mode: The CAN controller leaves Frozen mode when the FZEN bit is cleared in the CAN\_MCTRL register. This switch operation is confirmed by hardware clearing the FZC bit in the CAN\_MSTS register. The CAN controller must be synchronized with the bus.

Switch to Sleep mode: The CAN controller enters Sleep mode if DZEN = 1 and FZEN = 0 in the CAN\_MCTRL register. This switch operation is confirmed by hardware setting the DZC bit in the CAN\_MSTS register.

### Communication mode

After the CAN\_BTMG and CAN\_MCTRL registers are configured in Frozen mode, the CAN controller enters Communication mode and is ready for message reception and transmission.

Switch to Sleep mode: The CAN controller switches to Sleep mode when the DZEN bit is set in the CAN\_MCTRL register and the current CAN bus transmission is complete.

Switch to Frozen mode: The CAN controller enters Frozen mode when the FZEN bit is set in the CAN\_MCTRL register and the current CAN bus transmission is complete.

## Test modes

The CAN controller defines three test modes, including Listen-only mode, Loop back mode and combined Listen-only and Loop back mode. Test mode can be selected by setting the LOEN and LBEN bits in the CAN\_BTMG register.

- Listen-only mode is selected when the LOEN bit is set in the CAN\_BTMG register. In this mode, the CAN is able to receive data, but only recessive bits are output on the CANTX pin. In the meantime, the dominant bits output on the CANTX can be monitored by the receive side but without affecting the CAN bus.
- Loop back mode is selected by setting the LBEN bit in the CAN\_BTMG register. In this mode, The CAN only receives the level signal on its CANTX pin. Meanwhile, the CAN can also send data to the external bus. The Loop back mode is mainly used for self-test functions.
- It is possible to combine the Listen-only and Loop back mode by setting the LOEN and LBEN bits in the CAN\_BTMG register. In this case, the CAN is disconnected from the bus network, the CANTX pin remains in recessive state, and the transmit side is connected to the receive side.

## Message filtering

The received message has to go through filtering by its identifier. If passed, the message will be stored in the corresponding FIFOs. If not, the message will be discarded. The whole operation is done by hardware without using CPU resources.

### Filter bit width

The CAN controller provides 28 configurable and scalable filter banks (0 ~ 27). Each filter bank has two 32-bit registers, CAN\_FiFB1 and CAN\_FiFB2. The filter bit width can be configured as two 16 bits or one 32 bits, depending on the corresponding bits in the CAN\_FBWCFG register.

32-bit filter register CAN\_FiFBx includes the SID[10:0], EID[17:0], IDT and RTR bits.

|                       |                 |                |     |   |
|-----------------------|-----------------|----------------|-----|---|
| CAN_FiFB1[31:21]      | CAN_FiFB1[20:3] | CAN_FiFB1[2:0] |     |   |
| CAN_FiFB2[31:21]      | CAN_FiFB2[20:3] | CAN_FiFB2[2:0] |     |   |
| SID[10: 0]/EID[28:18] | EID[17:0]       | IDT            | RTR | 0 |

Two 16-bit filter register CAN\_FiFBx includes SID[10:0], IDT, RTR and EID[17:15] bits

|                  |                   |                   |                 |                 |                    |
|------------------|-------------------|-------------------|-----------------|-----------------|--------------------|
| CAN_FiFB1[31:21] | CAN_FiFB1 [20:19] | CAN_FiFB1 [18:16] | CAN_FiFB1[15:5] | CAN_FiFB1 [4:3] | CAN_FiFB1 [2:0]    |
| CAN_FiFB2[31:21] | CAN_FiFB2 [20:19] | CAN_FiFB2 [18:16] | CAN_FiFB2[15:5] | AN_FiFB2 [4:3]  | CAN_FiFB2 [2:0]    |
| SID[10:0]        | IDT               | RTR               | EID[17:15]      | SID[10:0]       | IDT RTR EID[17:15] |

### Filtering mode

The filter can be configured in identifier mask mode or in identifier list mode by setting the FMSELx bit in the CAN\_FMCFG register. The mask mode is used to specify which bits must match the pre-programmed identifiers, and which bits do not need. In identifier list mode, the identifier must match the pre-programmed identifier. The two modes can be used in conjunction with filter width to deliver four filtering modes below:

|         |                  |                 |                |
|---------|------------------|-----------------|----------------|
| ID      | CAN_FiFB1[31:21] | CAN_FiFB1[20:3] | CAN_FiFB1[2:0] |
| Mask    | CAN_FiFB2[31:21] | CAN_FiFB2[20:3] | CAN_FiFB2[2:0] |
| Mapping | SID[10:0]        | EID[17:0]       | IDT RTR 0      |

**Figure 232. 32-bit identifier mask mode**

|         |                  |                 |                |
|---------|------------------|-----------------|----------------|
| ID      | CAN_FiFB1[31:21] | CAN_FiFB1[20:3] | CAN_FiFB1[2:0] |
| ID      | CAN_FiFB2[31:21] | CAN_FiFB2[20:3] | CAN_FiFB2[2:0] |
| Mapping | SID[10:0]        | EID[17:0]       | IDT RTR 0      |

**Figure 233. 32-bit identifier list mode**

|         |                  |                    |
|---------|------------------|--------------------|
| ID      | CAN_FiFB1[15:5]  | CAN_FiFB1[4:0]     |
| Mask    | CAN_FiFB1[31:21] | CAN_FiFB1[20:16]   |
| ID      | CAN_FiFB2[15:5]  | CAN_FiFB2[4:0]     |
| Mask    | CAN_FiFB2[31:21] | CAN_FiFB2[20:16]   |
| Mapping | SID[10:0]        | RTR IDT EID[17:15] |

**Figure 234. 16-bit identifier mask mode**

|         |                  |                    |
|---------|------------------|--------------------|
| ID      | CAN_FiFB1[15:8]  | CAN_FiFB1[7:0]     |
| ID      | CAN_FiFB1[31:24] | CAN_FiFB1[23:16]   |
| ID      | CAN_FiFB2[15:8]  | CAN_FiFB2[7:0]     |
| ID      | CAN_FiFB2[31:24] | CAN_FiFB2[23:16]   |
| Mapping | SID[10:0]        | RTR IDT EID[17:15] |

**Figure 235. 16-bit identifier list mode**

## Filter match number

14 filter banks have different filtering effects dependent on the bit width mode. For example, 32-bit identifier mask mode contains the filters numbered  $n$  while 16-bit identifier list mode contains the filters numbered  $n$ ,  $n + 1$ ,  $n + 2$  and  $n + 3$ . When a frame of message passes through the filter number  $N$ , the number  $N$  is stored in the RFFMN[7:0] bit in the CAN\_RFCx register. The distribution of the filter number does not take into account the activation state of the filter banks.

| Filter bank | FIFO0                  | Active | Filter number | Filter bank         | FIFO1                 | Active              | Filter number |
|-------------|------------------------|--------|---------------|---------------------|-----------------------|---------------------|---------------|
| 0           | CAN_F0FB1[31:0]-ID     | Yes    | 0             | 3                   | CAN_F3FB1[15:0]-ID    | Yes                 | 0             |
|             | CAN_F0FB2[31:0]-ID     |        | 1             |                     | CAN_F3FB1[31:16]-ID   |                     | 1             |
| 1           | CAN_F1FB1[15:0]-ID     | Yes    | 2             |                     | CAN_F3FB2[15:0]-ID    |                     | 2             |
|             | CAN_F1FB1[31:16]-ID    |        | 3             |                     | CAN_F3FB2[31:16]-ID   |                     | 3             |
|             | CAN_F1FB2[15:0]-ID     |        | 4             | 4                   | CAN_F4FB1[31:0]-ID    | Yes                 | 4             |
|             | CAN_F1FB2[31:16]-ID    |        | 5             |                     | CAN_F4FB2[31:0]-Mask  |                     |               |
| 2           | CAN_F2FB1[31:0]-ID     | Yes    | 6             | 5                   | CAN_F5FB1[15:0]-ID    | No                  | 5             |
|             | CAN_F2FB2[31:0]-Mask   |        |               |                     | CAN_F5FB1[31:16]-Mask |                     | 6             |
| 6           | CAN_F6FB1[15:0]-ID     | No     | 7             |                     | CAN_F5FB2[15:0]-ID    |                     |               |
|             | CAN_F6FB1[31:16]-Mask  |        | 8             |                     | CAN_F5FB2[31:16]-Mask |                     |               |
|             | CAN_F6FB2[15:0]-ID     |        | 7             | CAN_F7FB1[15:0]-ID  | No                    | 7                   |               |
|             | CAN_F6FB2[31:16]-Mask  |        |               | CAN_F7FB1[31:16]-ID |                       | 8                   |               |
| 9           | CAN_F9FB1[31:0]-ID     | No     |               | 9                   |                       | CAN_F7FB2[15:0]-ID  | 9             |
|             | CAN_F9FB2[31:0]-ID     |        |               | 10                  |                       | CAN_F7FB2[31:16]-ID | 10            |
| 10          | CAN_F10FB1[15:0]-ID    | Yes    | 11            | 8                   | CAN_F8FB1[31:0]-ID    | Yes                 | 11            |
|             | CAN_F10FB1[31:16]-Mask |        |               |                     | CAN_F8FB2[31:0]-Mask  |                     |               |
|             | CAN_F10FB2[15:0]-ID    |        | 12            | 11                  | CAN_F11FB1[31:0]-ID   | Yes                 | 12            |
|             | CAN_F10FB2[31:16]-Mask |        |               |                     | CAN_F11FB2[31:0]-ID   |                     | 13            |
| 12          | CAN_F12FB1[15:0]-ID    | No     | 13            | 13                  | CAN_F13FB1[15:0]-ID   | Yes                 | 14            |
|             | CAN_F12FB1[31:16]-ID   |        | 14            |                     | CAN_F13FB1[31:16]-ID  |                     | 15            |
|             | CAN_F12FB2[15:0]-ID    |        | 15            |                     | CAN_F13FB2[15:0]-ID   |                     | 16            |
|             | CAN_F12FB2[31:16]-ID   |        | 16            |                     | CAN_F13FB2[31:16]-ID  |                     | 17            |

## Priority rules

When the CAN controller receives a frame of message, the message may pass through several filters. In this case, the filter match number stored in the receive mailbox is determined according to the following priority rules:

- A 32-bit filter has priority over a 16-bit filter
- For filters with identical bit width, the identifier list mode has priority over the identifier mask mode
- For filter with identical bit width and identifier mode, the lower number has priority over the higher number.

## Filter configuration

- The CAN filters are configured by setting the FCS bit in the CAN\_FCTRL register.
- Identifier mask mode or identifier list mode can be selected by setting the FMSELx bit in the CAN\_FMCFG register.

- The filter bit width can be configured as two 16 bits or one 32 bits by setting the FBWSELx bit in the CAN\_FBWCFG register.
  - The filter x is associated with FIFO0 or FIFO1 by setting the FRFSELx bit in the CAN\_FRF register.
  - The filter banks x are activated by setting FAENx = 1 in the CAN\_FACFG register.
  - Configure 0 ~ 27 filter banks by writing to the CAN\_FiFBx register (i = 0...27; x = 1, 2).
- Complete the CAN filter configuration by setting FCS = 0 in the CAN\_FCTRL register.

## Message transmission

### Register configuration

To transmit a message, it is necessary to select one transmit mailbox and configure it through the CAN\_TMIx, CAN\_TMCx, CAN\_TMDTLx and CAN\_TMDTHx registers. Once the mailbox configuration is complete, setting the TMSR bit in the CAN\_TMIx register can initiate CAN transmission.

### Message transmission

The mailbox enters pending state immediately after the mailbox is configured and the CAN controller receives the transmit request. At this point, the CAN controller will confirm whether the mailbox is given the highest priority or not. If yes, it will enter SCHEDULED STATE; otherwise, it will wait to get the highest priority. The mailbox in SCHEDULED state will monitor the CAN bus state so that the messages in SCHEDULED mailbox can be transmitted as soon as the CAN bus becomes idle. The mailbox will enter EMPTY state at the end of the message transmission.

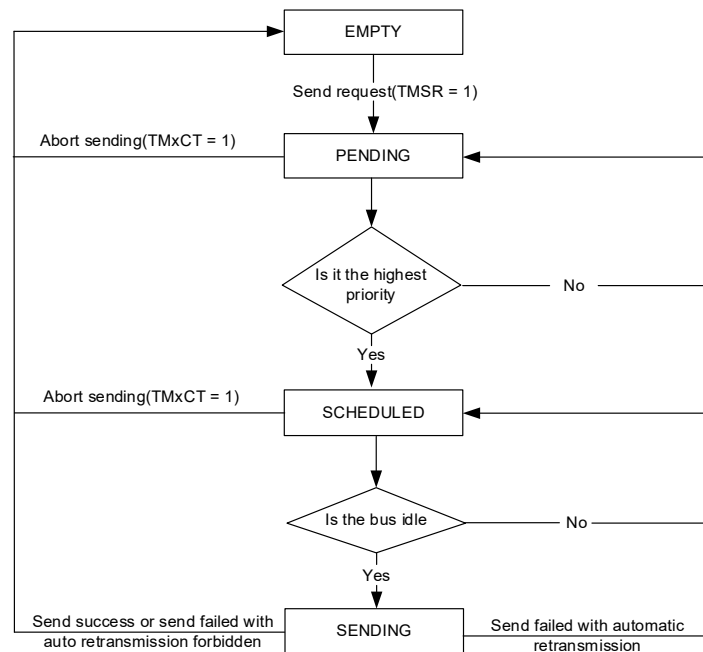


Figure 236. Transmit mailbox status

### Transmit priority configuration

When two or more transmit boxes are in PENDING state, their transmit priority must be given.

By identifier: When MMSSR = 0 in the CAN\_MCTRL register, the transmit order is defined by the identifier of the message in the mailbox. The message with lower identifier value has the highest priority. If the identifier values are the same, the message with lower mailbox number will be transmitted first.

By transmit request order: When MMSSR = 1 in the CAN\_MCTRL register, the transmit priority is given by the transmit request order of mailboxes.

### Transmit status and error status

The TMxTCF, TMxTSF, TMxALF, TMxTEF and TMxEF bits in the CAN\_TSTS register are used to indicate transmit status and error status.

TMxTCF bit: Transmission complete flag, indicating that the data transmission is complete when TMxTCF = 1.

TMxTSF bit: Transmission success flag, indicating that the data has been transmitted successfully when TMxTSF = 1.

TMxALF bit: Transmission arbitration lost flag, indicating that the data transmission arbitration is lost when TMxALF = 1.

TMxTEF bit: Transmission error flag, indicating that the data transmission failed due to bus error, and an error frame is sent when TMxTEF = 1.

TMxEF bit: Mailbox empty flag, indicating that the data transmission is complete and the mailbox becomes empty when TMxEF = 1.

### Transmit abort

The TMxCT bit is set in the CAN\_TSTS register to abort the transmission of the current mailbox, detailed as follows:

When the current transmission fails or arbitration is lost, if the automatic retransmission mode is disabled, the transmit mailbox become EMPTY; if the automatic retransmission mode is enabled, the transmit mailbox becomes SCHEDULED, the mailbox transmission then is aborted and becomes EMPTY.

When the current transmission is complete successfully, the mailbox becomes EMPTY.

## Message reception

### Register configuration

The CAN\_RFIx, CAN\_RFCx, CAN\_RFDTLx and CAN\_RFDTHx registers can be used by user applications to obtain valid messages.

### Message reception

The CAN controller boasts two FIFO with three levels to receive messages. FIFO rule is adopted. When the message is received correctly and has passed the identifier filtering, it is regarded as a valid message and is stored in the corresponding FIFO. The number of the received messages RFXMN[1:0] will be incremented by one whenever the receive FIFO receives a valid message. If a valid message is received when RFXMN[1:0] = 3, the controller will select either to overwrite the previous messages or discard the new incoming message through the MDRSEL bit in the CAN\_MCTRL register.

In the meantime, when the user reads a frame of message and the RfR is set in the CAN\_RfR register, one FIFO mailbox is released, and RfMN[1:0] bit is decremented by one in the CAN\_RfR register.

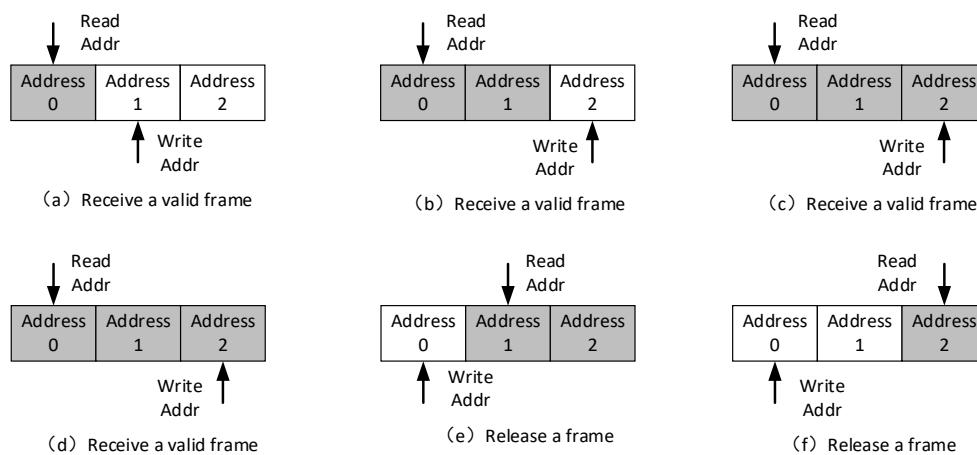
### Receive FIFO status

RfMN[1:0], RfFF and RfOF bits in the RfR register are used to indicate receive FIFO status.

RfMN[1:0]: indicates the number of valid messages stored in the FIFOx.

RfFF: indicates that three valid messages are stored in the FIFOx (i.e. the three mailboxes are full), as shown in (c) of Figure 237.

RfOF: indicates that a new valid message has been received while the FIFOx is full, as shown in (d) of Figure 237.



**Figure 237. Receive FIFO status**

## Error management

The status of CAN nodes is indicated by the receive error counter (TEC) and transmit error counter (REC) bits in the CAN\_ESTS register. In the meantime, the ETR[6:4] bit in the CAN\_ESTS register is used to record the last error source, and the corresponding interrupts will be generated when the CAN\_INTEN register is enabled.

- Error active flag: When both TEC and REC are lower than 128, the system is in the error active state. An error active flag is set when an error is detected.
- Error passive flag: When either TEC or REC is greater than 127, the system is in the error passive state. An error passive flag is set when an error is detected.
- Bus-off state: The bus-off state is entered when TEC is greater than 255. In this state, it is impossible to transmit and receive messages. The CAN recovers from bus-off state in two ways:
  - Option 1: When AEBOEN = 0 in the CAN\_MCTRL register, in communication mode, the software requests to enter Frozen mode and exit Frozen mode, and CAN will then resume from bus-off state after 128 occurrences of 11 consecutive recessive bits have been detected on the CAN RX pin.
  - Option 2: When AEBOEN = 1 in the CAN\_MCTRL register, the CAN will resume from bus-off state automatically after 128 occurrences of 11 consecutive recessive bits have been detected on the CAN RX pin.

## CAN registers

These peripheral registers must be accessed by words (32 bits).

**Table 74. CAN register map and reset values**

| Register | Offset      | Reset value |
|----------|-------------|-------------|
| MCTRL    | 000h        | 0x0001 0002 |
| MSTS     | 004h        | 0x0000 0C02 |
| TSTS     | 008h        | 0x1C00 0000 |
| RF0      | 00Ch        | 0x0000 0000 |
| FR1      | 010h        | 0x0000 0000 |
| INTEN    | 014h        | 0x0000 0000 |
| ESTS     | 018h        | 0x0000 0000 |
| BTMG     | 01Ch        | 0x0123 0000 |
| Reserved | 020h ~ 17Fh | xx          |
| TMI0     | 180h        | 0xFFFF XXXX |
| TMC0     | 184h        | 0xFFFF XXXX |
| TMDTL0   | 188h        | 0xFFFF XXXX |
| TMDTH0   | 18Ch        | 0xFFFF XXXX |
| TMI1     | 190h        | 0xFFFF XXXX |
| TMC1     | 194h        | 0xFFFF XXXX |
| TMDTL1   | 198h        | 0xFFFF XXXX |
| TMDTH1   | 19Ch        | 0xFFFF XXXX |
| TMI2     | 1A0h        | 0xFFFF XXXX |
| TMC2     | 1A4h        | 0xFFFF XXXX |
| TMDTL2   | 1A8h        | 0xFFFF XXXX |
| TMDTH2   | 1ACh        | 0xFFFF XXXX |
| RFI0     | 1B0h        | 0xFFFF XXXX |
| RFC0     | 1B4h        | 0xFFFF XXXX |
| RFDTL0   | 1B8h        | 0xFFFF XXXX |
| RFDTH0   | 1BCh        | 0xFFFF XXXX |
| RFI1     | 1C0h        | 0xFFFF XXXX |
| RFC1     | 1C4h        | 0xFFFF XXXX |
| RFDTL1   | 1C8h        | 0xFFFF XXXX |
| RFDTH1   | 1CCh        | 0xFFFF XXXX |
| Reserved | 1D0h ~ 1FFh | xx          |
| FCTRL    | 200h        | 0x2A1C 0E01 |
| FBWCFG   | 204h        | 0x0000 0000 |
| Reserved | 208h        | xx          |
| FSCFG    | 20Ch        | 0x0000 0000 |
| Reserved | 210h        | xx          |
| FRF      | 214h        | 0x0000 0000 |
| Reserved | 218h        | xx          |
| FACFG    | 21Ch        | 0x0000 0000 |
| Reserved | 220h ~ 23Fh | xx          |
| F0FB1    | 240h        | 0xFFFF XXXX |
| F0FB2    | 244h        | 0xFFFF XXXX |



| Register | Offset | Reset value |
|----------|--------|-------------|
| F1FB1    | 248h   | 0XXXXX XXXX |
| F1FB2    | 24Ch   | 0XXXXX XXXX |
| ...      | ...    | ...         |
| F13FB1   | 2A8h   | 0XXXXX XXXX |
| F13FB2   | 2ACh   | 0XXXXX XXXX |

## CAN control and status registers

### CAN master control register (CAN\_MCTRL)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:17 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 16    | PTD      | 0x1         | rw   | Prohibit trans when debug<br>0: Transmission works during debug<br>1: Transmission is prohibited during debug. Receive FIFO can be still accessible normally.<br>Note: Transmission can be disabled only when PTD and CANx_PAUSE bits in the DEBUG_CTRL register are set simultaneously.   |
| Bit 15    | SPRST    | 0x0         | rw1s | Software partial reset<br>0: Normal<br>1: Software partial reset<br>Note:<br>SPRST only reset receive FIFO and MCTRL register.<br>The CAN enters Sleep mode after reset. Then this bit is automatically cleared by hardware.   |
| Bit 14:8  | Reserved | 0x00        | resd | Kept at its default value.   |
| Bit 7     | TTCEN    | 0x0         | rw   | Time triggered communication mode enable<br>0: Time triggered communication mode disabled<br>1: Time triggered communication mode enabled  |
| Bit 6     | AEBOEN   | 0x0         | rw   | Automatic exit bus-off enable<br>0: Automatic exit bus-off disabled<br>1: Automatic exit bus-off enabled<br>Note:<br>When Automatic exit bus-off mode is enabled, the hardware will automatically leave bus-off mode as soon as an exit timing is detected on the CAN bus.<br>When Automatic exit bus-off mode is disabled, the software must enter/leave the freeze mode once more, and then the bus-off state is left only when an exit timing is detected on the CAN bus. |
| Bit 5     | AEDEN    | 0x0         | rw   | Automatic exit doze mode enable<br>0: Automatic exit sleep mode disabled<br>1: Automatic exit sleep mode enabled<br>Note:<br>When Automatic exit sleep mode is disabled, the sleep mode is left by software clearing the sleep request command.<br>When Automatic exit sleep mode is enabled, the sleep mode is left without the need of software intervention as soon as a message is monitored on the CAN bus.   |

| Bit   | Name   | Reset value | Type | Description   |
|-------|--------|-------------|------|---|
| Bit 4 | PRSFEN | 0x0         | rw   | Prohibit retransmission enable when sending fails enable<br>0: Retransmission is enabled.<br>1: Retransmission is disabled.   |
| Bit 3 | MDRSEL | 0x0         | rw   | Message discard rule select when overflow<br>0: The previous message is discarded.<br>1: The new incoming message is discarded.   |
| Bit 2 | MMSSR  | 0x0         | rw   | Multiple message transmit sequence rule<br>0: The message with the smallest identifier is first transmitted.<br>1: The message with the first request order is first transmitted.   |
| Bit 1 | DZEN   | 0x1         | rw   | Doze mode enable<br>0: Sleep mode is disabled.<br>1: Sleep mode is enabled.<br><br>Note:<br>The hardware will automatically leave sleep mode when the AEDEN is set and a message is monitored on the CAN bus.<br>After CAN reset or partial software reset, this bit is forced to be set by hardware, that is, the CAN will keep in sleep mode, by default.   |
| Bit 0 | FZEN   | 0x0         | rw   | Freeze mode enable<br>0: Freeze mode disabled<br>1: Freeze mode enabled<br><br>Note:<br>The CAN leaves Freeze mode once 11 consecutive recessive bits have been detected on the RX pin. For this reason, the software acknowledges the entry of Freeze mode after the FZC bit is cleared by hardware.<br><br>The Freeze mode is entered only when the current CAN activity (transmission or reception) is completed. Thus the software acknowledges the exit of Freeze mode after the FZC bit is cleared by hardware. |

#### CAN master status register (CAN\_MSTS)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:12 | Reserved | 0x00000     | resd | Kept at its default value.  |
| Bit 11    | REALRX   | 0x1         | ro   | Real time level on RX pin<br>0: Low<br>1: High  |
| Bit 10    | LSAMPRX  | 0x1         | ro   | Last sample level on RX pin<br>0: Low<br>1: High<br><br>Note: This value keeps updating with the REALRX.  |
| Bit 9     | CURS     | 0x0         | ro   | Current receive status<br>0: No reception occurs<br>1: Reception is in progress<br><br>Note: This bit is set by hardware when the CAN reception starts, and it is cleared by hardware at the end of reception.      |
| Bit 8     | CUSS     | 0x0         | ro   | Current transmit status<br>0: No transmit occurs<br>1: Transmit is in progress<br><br>Note: This bit is set by hardware when the CAN transmission starts, and it is cleared by hardware at the end of transmission. |

| Bit     | Name     | Reset value | Type | Description   |
|---------|----------|-------------|------|---|
| Bit 7:5 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 4   | EDZIF    | 0x0         | rw1c | <p>Enter doze mode interrupt flag</p> <p>0: Sleep mode is not entered or no condition for flag set.</p> <p>1: Sleep mode is entered.</p> <p>Note:</p> <p>This bit is set by hardware only when EDZIEN = 1 and the CAN enters Sleep mode. When set, this bit will generate a status change interrupt. This bit is cleared by software (writing 1 to itself) or by hardware when DZC is cleared.</p>  |
| Bit 3   | QDZIF    | 0x0         | rw1c | <p>Exit doze mode interrupt flag</p> <p>0: Sleep mode is not left or no condition for exit.</p> <p>1: Sleep mode has been left or exit condition has generated.</p> <p>Note:</p> <p>This bit is cleared by software (writing 1 to itself)</p> <p>Sleep mode is left when a SOF is detected on the bus.</p> <p>When QDZIEN = 1, this bit will generate a status change interrupt.</p>  |
| Bit 2   | EOIF     | 0x0         | rw1c | <p>Error occur interrupt flag</p> <p>0: No error interrupt or no condition for error interrupt flag</p> <p>1: Error interrupt is generated.</p> <p>Note:</p> <p>This bit is cleared by software (writing 1 to itself).</p> <p>This bit is set by hardware only when the corresponding bit is set in the CAN_ESTS register and the corresponding interrupt enable bit in the CAN_INTEN register is enabled. When set, this bit will generate a status change interrupt.</p>  |
| Bit 1   | DZC      | 0x1         | ro   | <p>Doze mode acknowledge</p> <p>0: The CAN is not in Sleep mode.</p> <p>1: CAN is in Sleep mode.</p> <p>Note:</p> <p>This bit is used to decide whether the CAN is in Sleep mode or not. This bit acknowledges the Sleep mode request generated by software.</p> <p>The Sleep mode can be entered only when the current CAN activity (transmission or reception) is completed. For this reason, the software acknowledges the entry of Sleep mode after this bit is set by hardware.</p> <p>The Sleep mode is left only once 11 consecutive recessive bits have been detect on the CAN RX pin. For this reason, the software acknowledges the exit of Sleep mode after this bit is cleared by hardware.</p> |

| Bit   | Name | Reset value | Type | Description   |
|-------|------|-------------|------|---|
| Bit 0 | FZC  | 0x0         | ro   | <p>Freeze mode confirm</p> <p>0: The CAN is not in Freeze mode.</p> <p>1: The CAN is in Freeze mode.</p> <p>Note:</p> <p>This bit is used to decide whether the CAN is in Freeze mode or not. This bit acknowledges the Freeze mode request generated by software.</p> <p>The Freeze mode can be entered only when the current CAN activity (transmission or reception) is completed. For this reason, the software acknowledges the entry of Freeze mode after this bit is set by hardware.</p> <p>The Freeze mode is left only once 11 consecutive recessive bits have been detect on the CAN RX pin. For this reason, the software acknowledges the exit of Freeze mode after this bit is cleared by hardware.</p> |

#### CAN transmit status register (CAN\_TSTS)

| Bit    | Name   | Reset value | Type | Description   |
|--------|--------|-------------|------|---|
| Bit 31 | TM2LPF | 0x0         | ro   | <p>Transmit mailbox 2 lowest priority flag</p> <p>0: Mailbox 2 is not given the lowest priority.</p> <p>1: Lowest priority (This indicates that more than one mailboxes are pending for transmission, the mailbox 2 has the lowest priority.)</p> |
| Bit 30 | TM1LPF | 0x0         | ro   | <p>Transmit mailbox 1 lowest priority flag</p> <p>0: Mailbox 1 is not given the lowest priority.</p> <p>1: Lowest priority (This indicates that more than one mailboxes are pending for transmission, the mailbox 1 has the lowest priority.)</p> |
| Bit 29 | TM0LPF | 0x0         | ro   | <p>Transmit mailbox 0 lowest priority flag</p> <p>0: Mailbox 0 is not given the lowest priority.</p> <p>1: Lowest priority (This indicates that more than one mailboxes are pending for transmission, the mailbox 0 has the lowest priority.)</p> |
| Bit 28 | TM2EF  | 0x1         | ro   | <p>Transmit mailbox 2 empty flag</p> <p>This bit is set by hardware when no transmission is pending in the mailbox 2.</p>   |
| Bit 27 | TM1EF  | 0x1         | ro   | <p>Transmit mailbox 1 empty flag</p> <p>This bit is set by hardware when no transmission is pending in the mailbox 1.</p>   |
| Bit 26 | TM0EF  | 0x1         | ro   | <p>Transmit mailbox 0 empty flag</p> <p>This bit is set by hardware when no transmission is pending in the mailbox 0.</p>   |

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 25:24 | TMNR     | 0x0         | ro   | <p>Transmit Mailbox number record</p> <p>Note:</p> <p>If the transmit mailbox is free, these two bits refer to the number of the next transmit mailbox free.</p> <p>For example, in case of free CAN, the value of these two bit becomes 01 after a message transmit request is written.</p> <p>If the transmit box is full, these two bits refer to the number of the transmit mailbox with the lowest priority.</p> <p>For example, when there are three messages are pending for transmission, the identifiers of mailbox 0, mailbox 1 and mailbox 2 are 0x400, 0x433 and 0x411 respectively, and the value of these two bits becomes 01.</p> |
| Bit 23    | TM2CT    | 0x0         | ro   | <p>Transmit mailbox 2 cancel transmit</p> <p>0: No effect</p> <p>1: Transmission is cancelled.</p> <p>Note: Software sets this bit to abort the transmission of mailbox 2. This bit is cleared by hardware when the transmit message in the mailbox 2 is cleared. Setting this bit has no effect if the mailbox 2 is free.</p>   |
| Bit 22:20 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 19    | TM2TEF   | 0x0         | rw1c | <p>Transmit mailbox 2 transmission error flag</p> <p>0: No error</p> <p>1: Mailbox 2 transmission error</p> <p>Note:</p> <p>This bit is set when the mailbox 2 transmission error occurred. It is cleared by software writing 1 or by hardware at the start of the next transmission</p>   |
| Bit 18    | TM2ALF   | 0x0         | rw1c | <p>Transmit mailbox 2 arbitration lost flag</p> <p>0: No arbitration lost</p> <p>1: Transmit mailbox 2 arbitration lost</p> <p>Note:</p> <p>This bit is set when the mailbox 2 transmission failed due to an arbitration lost. It is cleared by software writing 1 or by hardware at the start of the next transmission</p>  |
| Bit 17    | TM2TSF   | 0x0         | rw1c | <p>Transmit mailbox 2 transmission success flag</p> <p>0: Transmission failed</p> <p>1: Transmission was successful.</p> <p>Note:</p> <p>This bit indicates whether the mailbox 2 transmission is successful or not. It is cleared by software writing 1.</p>  |
| Bit 16    | TM2TCF   | 0x0         | rw1c | <p>Transmit mailbox 2 transmission completed flag</p> <p>0: Transmission is in progress</p> <p>1: Transmission is completed</p> <p>Note:</p> <p>This bit is set by hardware when the transmission/abort request on mailbox 2 has been completed. It is cleared by software writing 1 or by hardware when a new transmission request is received. Clearing this bit will clear the TM2TSF, TM2ALF and TM2TEF bits of mailbox 2.</p>   |

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 15    | TM1CT    | 0x0         | rw1s | Transmit mailbox 1 cancel transmit<br>0: No effect<br>1: Mailbox 1 cancel transmit<br>Note: This bit is set by software to abort the transmission request on mailbox 1. Clearing the message transmission on mailbox 1 will clear this bit. Setting by this software has no effect when the mailbox 1 is free.  |
| Bit 14:12 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 11    | TM1TEF   | 0x0         | rw1c | Transmit mailbox 1 transmission error flag<br>0: No error<br>1: Mailbox 1 transmission error<br>Note:<br>This bit is set when the mailbox 1 transmission error occurred. It is cleared by software writing 1 or by hardware at the start of the next transmission   |
| Bit 10    | TM1ALF   | 0x0         | rw1c | Transmit mailbox 1 arbitration lost flag<br>0: No arbitration lost<br>1: Transmit mailbox 1 arbitration lost<br>Note:<br>This bit is set when the mailbox 1 transmission failed due to an arbitration lost. It is cleared by software writing 1 or by hardware at the start of the next transmission  |
| Bit 9     | TM1TSF   | 0x0         | rw1c | Transmit mailbox 1 transmission success flag<br>0: Transmission failed<br>1: Transmission was successful.<br>Note:<br>This bit indicates whether the mailbox 1 transmission is successful or not. It is cleared by software writing 1.  |
| Bit 8     | TM1TCF   | 0x0         | rw1c | Transmit mailbox 1 transmission completed flag<br>0: Transmission is in progress<br>1: Transmission is completed<br>Note:<br>This bit is set by hardware when the transmission/abort request on mailbox 1 has been completed. It is cleared by software writing 1 or by hardware when a new transmission request is received. Clearing this bit will clear the TM1TSF, TM1ALF and TM1TEF bits of mailbox 1. |
| Bit 7     | TM0CT    | 0x0         | rw1s | Transmit mailbox 0 cancel transmit<br>0: No effect<br>1: Mailbox 0 cancel transmit<br>Note: This bit is set by software to abort the transmission request on mailbox 0. Clearing the message transmission on mailbox 0 will clear this bit. Setting by this software has no effect when the mailbox 0 is free.  |
| Bit 6:4   | Reserved | 0x0         | resd | Kept at its default value.  |

| Bit   | Name   | Reset value | Type | Description   |
|-------|--------|-------------|------|---|
| Bit 3 | TM0TEF | 0x0         | rw1c | Transmit mailbox 0 transmission error flag<br>0: No error<br>1: Mailbox 0 transmission error<br>Note:<br>This bit is set when the mailbox 0 transmission error occurred.<br>It is cleared by software writing 0 or by hardware at the start of the next transmission  |
| Bit 2 | TM0ALF | 0x0         | rw1c | Transmit mailbox 0 arbitration lost flag<br>0: No arbitration lost<br>1: Transmit mailbox 0 arbitration lost<br>Note:<br>This bit is set when the mailbox 0 transmission failed due to an arbitration lost.<br>It is cleared by software writing 1 or by hardware at the start of the next transmission   |
| Bit 1 | TM0TSF | 0x0         | rw1c | Transmit mailbox 0 transmission success flag<br>0: Transmission failed<br>1: Transmission was successful.<br>Note:<br>This bit indicates whether the mailbox 0 transmission is successful or not. It is cleared by software writing 1.  |
| Bit 0 | TM0TCF | 0x0         | rw1c | Transmit mailbox 0 transmission completed flag<br>0: Transmission is in progress<br>1: Transmission is completed<br>Note:<br>This bit is set by hardware when the transmission/abort request on mailbox 0 has been completed.<br>It is cleared by software writing 1 or by hardware when a new transmission request is received.<br>Clearing this bit will clear the TM0TSF, TM0ALF and TM0TEF bits of mailbox 0. |

#### CAN receive FIFO 0 register (CAN\_RF0)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31:6 | Reserved | 0x0000000   | resd | Kept at its default value.   |
| Bit 5    | RF0R     | 0x0         | rw1s | Receive FIFO 0 release<br>0: No effect<br>1: Release FIFO<br>Note:<br>This bit is set by software to release FIFO 0. It is cleared by hardware when the FIFO 0 is released.<br>Setting this bit by software has no effect when the FIFO 0 is empty.<br>If there are more than two messages pending in the FIFO 0, the software has to release the FIFO 0 to access the second message. |

| Bit     | Name     | Reset value | Type | Description  |
|---------|----------|-------------|------|--|
| Bit 4   | RF0OF    | 0x0         | rw1c | Receive FIFO 0 overflow flag<br>0: No overflow<br>1: Receive FIFO 0 overflow<br>Note:<br>This bit is set by hardware when a new message has been received and passed the filter while the FIFO 0 is full.<br>It is cleared by software by writing 1.   |
| Bit 3   | RF0FF    | 0x0         | rw1c | Receive FIFO 0 full flag<br>0: Receive FIFO 0 is not full<br>1: Receive FIFO 0 is full<br>Note:<br>This bit is set by hardware when three messages are pending in the FIFO 0.<br>It is cleared by software by writing 1.   |
| Bit 2   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 1:0 | RF0MN    | 0x0         | ro   | Receive FIFO 0 message num<br>Note:<br>These two bits indicate how many messages are pending in the FIFO 0.<br>RF0ML bit is incremented by one each time a new message has been received and passed the filter while the FIFO 0 is not full.<br>RF0ML bit is decremented by one each time the software releases the receive FIFO 0 by writing 1 to the RF0R bit. |

#### CAN receive FIFO 1 register (CAN\_RF1)

| Bit      | Name     | Reset value | Type | Description  |
|----------|----------|-------------|------|--|
| Bit 31:6 | Reserved | 0x0000000   | resd | Kept at its default value.   |
| Bit 5    | RF1R     | 0x0         | rw1s | Receive FIFO 1 release<br>0: No effect<br>1: Release FIFO<br>Note:<br>This bit is set by software to release FIFO 1. It is cleared by hardware when the FIFO 1 is released.<br>Setting this bit by software has no effect when the FIFO 1 is empty.<br>If there are more than two messages pending in the FIFO 0, the software has to release the FIFO 1 to access the second message. |
| Bit 4    | RF1OF    | 0x0         | rw1c | Receive FIFO 1 overflow flag<br>0: No overflow<br>1: Receive FIFO 1 overflow<br>Note:<br>This bit is set by hardware when a new message has been received and passed the filter while the FIFO 1 is full.<br>It is cleared by software by writing 1.   |



| Bit     | Name     | Reset value | Type | Description  |
|---------|----------|-------------|------|--|
| Bit 3   | RF1FF    | 0x0         | rw1c | Receive FIFO 1 full flag<br>0: Receive FIFO 1 is not full<br>1: Receive FIFO 1 is full<br>Note:<br>This bit is set by hardware when three messages are pending in the FIFO 1.<br>It is cleared by software by writing 1.   |
| Bit 2   | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 1:0 | RF1MN    | 0x0         | ro   | Receive FIFO 1 message num<br>Note:<br>These two bits indicate how many messages are pending in the FIFO 1.<br>RF1ML bit is incremented by one each time a new message has been received and passed the filter while the FIFO 1 is not full.<br>RF1ML bit is decremented by one each time the software releases the receive FIFO 1 by writing 1 to the RF1R bit. |

#### CAN interrupt enable register (CAN\_INTEN)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:18 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 17    | EDZIEN   | 0x0         | rw   | Enter doze mode interrupt enable<br>0: Enter sleep mode interrupt disabled<br>1: Enter sleep mode interrupt enabled<br>Note: EDZIF flag bit corresponds to this interrupt. An interrupt is generated when both this bit and EDZIF bit are set. |
| Bit 16    | QDZIEN   | 0x0         | rw   | Quit doze mode interrupt enable<br>0: Quit sleep mode interrupt disabled<br>1: Quit sleep mode interrupt enabled<br>Note: The flag bit of this interrupt is the QDZIF bit. An interrupt is generated when both this bit and QDZIF bit are set. |
| Bit 15    | EOIEN    | 0x0         | rw   | Error occur interrupt enable<br>0: Error interrupt disabled<br>1: Error interrupt enabled<br>Note: The flag bit of this interrupt is the EOIF bit. An interrupt is generated when both this bit and EOIF bit are set.                          |
| Bit 14:12 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 11    | ETRIEN   | 0x0         | rw   | Error type record interrupt enable<br>0: Error type record interrupt disabled<br>1: Error type record interrupt enabled<br>Note: EOIF is set only when this interrupt is enabled and the ETR[2:0] is set by hardware.                          |
| Bit 10    | BOIEN    | 0x0         | rw   | Bus-off interrupt enable<br>0: Bus-off interrupt disabled<br>1: Bus-off interrupt enabled<br>Note: EOIF is set only when this interrupt is enabled and the BOF is set by hardware.   |

| Bit   | Name     | Reset value | Type | Description  |
|-------|----------|-------------|------|--|
| Bit 9 | EPIEN    | 0x0         | rw   | Error passive interrupt enable<br>0: Error passive interrupt disabled<br>1: Error passive interrupt enabled<br>Note: EOIF is set only when this interrupt is enabled and the EPF is set by hardware.   |
| Bit 8 | EAIEN    | 0x0         | rw   | Error active interrupt enable<br>0: Error warning interrupt disabled<br>1: Error warning interrupt enabled<br>Note: EOIF is set only when this interrupt is enabled and the EAF is set by hardware.  |
| Bit 7 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 6 | RF1OIEEN | 0x0         | rw   | Receive FIFO 1 overflow interrupt enable<br>0: Receive FIFO 1 overflow interrupt disabled<br>1: Receive FIFO 1 overflow interrupt enabled<br>Note: The flag bit of this interrupt is the RF1OF bit. An interrupt is generated when this bit and RF1OF bit are set. |
| Bit 5 | RF1FIEEN | 0x0         | rw   | Receive FIFO 1 full interrupt enable<br>0: Receive FIFO 1 full interrupt disabled<br>1: Receive FIFO 1 full interrupt enabled<br>Note: The flag bit of this interrupt is the RF1FF bit. An interrupt is generated when this bit and RF1FF bit are set.             |
| Bit 4 | RF1MIEEN | 0x0         | rw   | FIFO 1 receive message interrupt enable<br>0: FIFO 1 receive message interrupt disabled<br>1: FIFO 1 receive message interrupt enabled<br>Note: The flag bit of this interrupt is RF1MN bit, so an interrupt is generated when this bit and RF1MN bit are set.     |
| Bit 3 | RF0OIEEN | 0x0         | rw   | Receive FIFO 0 overflow interrupt enable<br>0: Receive FIFO 0 overflow interrupt disabled<br>1: Receive FIFO 0 overflow interrupt enabled<br>Note: The flag bit of this interrupt is RF0OF bit, so an interrupt is generated when this bit and RF0OF bit are set.  |
| Bit 2 | RF0FIEEN | 0x0         | rw   | Receive FIFO 0 full interrupt enable<br>0: Receive FIFO 0 full interrupt disabled<br>1: Receive FIFO 0 full interrupt enabled<br>Note: The flag bit of this interrupt is the RF0FF bit. An interrupt is generated when this bit and RF0FF bit are set              |
| Bit 1 | RF0MIEEN | 0x0         | rw   | FIFO 0 receive message interrupt enable<br>0: FIFO 0 receive message interrupt disabled<br>1: FIFO 0 receive message interrupt enabled<br>Note: The flag bit of this interrupt is the RF0MN bit. An interrupt is generated when this bit and RF0MN bit are set     |
| Bit 0 | TCIEN    | 0x0         | rw   | Transmit mailbox empty interrupt enable<br>0: Transmit mailbox empty interrupt disabled<br>1: Transmit mailbox empty interrupt enabled<br>Note: The flag bit of this interrupt is the TMxTCF bit. An interrupt is generated when this bit and TMxTCF bit are set   |

### CAN error status register (CAN\_ESTS)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:24 | REC      | 0x00        | ro   | Receive error counter<br>This counter is implemented in accordance with the receive part of the fault confinement mechanism of the CAN protocol.   |
| Bit 23:16 | TEC      | 0x00        | ro   | Transmit error counter<br>This counter is implemented in accordance with the transmit part of the fault confinement mechanism of the CAN protocol.   |
| Bit 15:7  | Reserved | 0x00        | resd | Kept at its default value.   |
| Bit 6:4   | ETR      | 0x0         | rw   | Error type record<br>000: No error<br>001: Bit stuffing error<br>010: Format error<br>011: Acknowledgement error<br>100: Recessive bit error<br>101: Dominant bit error<br>110: CRC error<br>111: Set by software<br>Note:<br>This field is used to indicate the current error type. It is set by hardware according to the error condition detected on the CAN bus. It is cleared by hardware when a message has been transmitted or received successfully.<br>If the error code 7 is not used by hardware, this field can be set by software to monitor the code update. |
| Bit 3     | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 2     | BOF      | 0x0         | ro   | Bus-off flag<br>0: Bus-off state is not entered.<br>1: Bus-off state is entered.<br>Note: When the TEC is greater than 255, the bus-off state is entered, and this bit is set by hardware.   |
| Bit 1     | EPF      | 0x0         | ro   | Error passive flag<br>0: Error passive state is not entered<br>1: Error passive state is entered<br>Note: This bit is set by hardware when the current error times has reached the Error passive state limit (Receive Error Counter or Transmit Error Counter > 127)   |
| Bit 0     | EAF      | 0x0         | ro   | Error active flag<br>0: Error active state is not entered<br>1: Error active state is entered<br>Note: This bit is set by hardware when the current error times has reached the Error active state limit (Receive Error Counter or Transmit Error Counter ≥ 96)  |

### CAN bit timing register (CAN\_BTMG)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31    | LOEN     | 0x0         | rw   | Listen-Only mode<br>0: Listen-Only mode disabled<br>1: Listen-Only mode enabled   |
| Bit 30    | LBEN     | 0x0         | rw   | Loop back mode<br>0: Loop back mode disabled<br>1: Loop back mode enabled   |
| Bit 29:26 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 25:24 | RS AW    | 0x1         | rw   | Resynchronization width<br>$t_{RSAW} = t_{CAN} \times (RSAW[1:0] + 1)$<br>Note: This field defines the maximum of time unit that the CAN hardware is allowed to lengthen or shorten in a bit. |
| Bit 23    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 22:20 | BTS2     | 0x2         | rw   | Bit time segment 2<br>$t_{BTS2} = t_{CAN} \times (BTS2[2:0] + 1)$<br>Note: This field defines the number of time unit in Bit time segment 2.  |
| Bit 19:16 | BTS1     | 0x3         | rw   | Bit time segment 1<br>$t_{BTS1} = t_{CAN} \times (BTS1[3:0] + 1)$<br>Note: This field defines the number of time unit in Bit time segment 1.  |
| Bit 15:12 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 11:0  | BRDIV    | 0x000       | rw   | Baud rate division<br>$t_q = (BRDIV[11:0] + 1) \times t_{PCLK}$<br>Note: This field defines the length of a time unit ( $t_q$ ).  |

### CAN mailbox registers

This section describes the registers of the transmit and receive mailboxes. Refer to Section 19.6.5 for more information on register map.

Transmit and receive mailboxes are the same except:

- RFFMN field in the CAN\_RFCx register.
- A receive mailbox is read-only.
- A transmit mailbox can be written only when empty. TM2S = 1 in the CAN\_TSTS register indicates that the mailbox is empty.

There are three transmit mailboxes and two receive mailboxes. Each receive mailbox has 3-level depth of FIFO, and can only access to the first received message in the FIFO.

Each mailbox contains four registers.

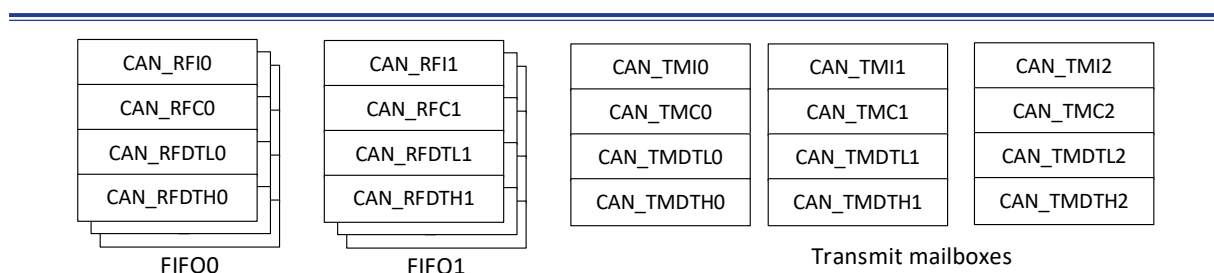


Figure 238. Transmit and receive mailboxes

### Transmit mailbox identifier register (CAN\_TMIx) (x = 0..2)

Note: 1. This register is write-protected when its mailboxes are pending for transmission.

2. This register implements the Transmit Request control (bit 0) – reset value 0.

| Bit       | Name        | Reset value | Type | Description  |
|-----------|-------------|-------------|------|--|
| Bit 31:21 | TMSID/TMEID | 0xXXX       | rw   | Transmit mailbox standard identifier or extended identifier high bytes<br>Note: This field defines the 11-bit high bytes of the standard identifier or extended identifier.        |
| Bit 20:3  | TMEID       | 0xxxxxx     | rw   | Transmit mailbox extended identifier<br>Note: This field defines the 18-bit low bytes of the extended identifier.  |
| Bit 2     | TMIDSEL     | 0xX         | rw   | Transmit mailbox identifier type select<br>0: Standard identifier<br>1: Extended identifier  |
| Bit 1     | TMFRSEL     | 0xX         | rw   | Transmit mailbox frame type select<br>0: Data frame<br>1: Remote frame   |
| Bit 0     | TMSR        | 0x0         | rw   | Transmit mailbox send request<br>0: No effect<br>1: Transmit request<br>Note: This bit is cleared by hardware when the transmission has been completed (The mailbox becomes empty) |

### Transmit mailbox data length and time stamp register (CAN\_TMCx) (x = 0..2)

All the bits in the register are write-protected when the mailbox is not in empty state.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | TMTS     | 0xxxxxx     | rw   | Transmit mailbox time stamp<br>Note: This field contains the value of the CAN timer sampled at the SOF transmission.   |
| Bit 15:9  | Reserved | 0xxx        | resd | Kept at its default value  |
| Bit 8     | TMTSTEN  | 0xX         | rw   | Transmit mailbox time stamp transmit enable<br>0: Time stamp is not sent<br>1: Time stamp is sent<br>Note:<br>This bit is valid only when the time-triggered communication mode is enabled.<br>In the time stamp MTS[15:0], the MTS[7:0] is stored in the TMDT7, and MTS[15:8] in the TMDT6. The data length must be programmed as 8 to send time stamp. |
| Bit 7:4   | Reserved | 0xX         | resd | Kept at its default value  |
| Bit 3:0   | TMDTBL   | 0xX         | rw   | Transmit mailbox data byte length<br>Note: This field defines the data length of a transmit message.<br>A transmit message can contain from 0 to 8 data bytes.   |

### Transmit mailbox data low register (CAN\_TMDTLx) (x = 0..2)

All the bits in the register are write-protected when the mailbox is not in empty state.

| Bit       | Name  | Reset value | Type | Description                  |
|-----------|-------|-------------|------|------------------------------|
| Bit 31:24 | TMDT3 | 0xXX        | rw   | Transmit mailbox data byte 3 |
| Bit 23:16 | TMDT2 | 0xXX        | rw   | Transmit mailbox data byte 2 |
| Bit 15:8  | TMDT1 | 0xXX        | rw   | Transmit mailbox data byte 1 |
| Bit 7:0   | TMDT0 | 0xXX        | rw   | Transmit mailbox data byte 0 |

### Transmit mailbox data high register (CAN\_TMDTHx) (x = 0..2)

All the bits in the register are write-protected when the mailbox is not in empty state.

| Bit       | Name  | Reset value | Type | Description  |
|-----------|-------|-------------|------|--|
| Bit 31:24 | TMDT7 | 0xXX        | rw   | Transmit mailbox data byte 7   |
| Bit 23:16 | TMDT6 | 0xXX        | rw   | Transmit mailbox data byte 6<br>Note: This field will be replaced with MTS[15:8] when the time-triggered communication mode is enabled and the corresponding time stamp transmit is enabled. |
| Bit 15:8  | TMDT5 | 0xXX        | rw   | Transmit mailbox data byte 5   |
| Bit 7:0   | TMDT4 | 0xXX        | rw   | Transmit mailbox data byte 4   |

### Receive FIFO mailbox identifier register (CAN\_RFIx) (x = 0..1)

Note: All the receive mailbox registers are read-only.

| Bit       | Name        | Reset value | Type | Description   |
|-----------|-------------|-------------|------|---|
| Bit 31:21 | RFSID/RFEID | 0XXXX       | ro   | Receive FIFO standard identifier or receive FIFO extended identifier<br>Note: This field defines the 11-bit high bytes of the standard identifier or extended identifier. |
| Bit 20:3  | RFEID       | 0XXXXXX     | ro   | Receive FIFO extended identifier<br>Note: This field defines the 18-bit low bytes of the extended identifier.   |
| Bit 2     | RFIDI       | 0xX         | ro   | Receive FIFO identifier type indication<br>0: Standard identifier<br>1: Extended identifier   |
| Bit 1     | RFFRI       | 0xX         | Ro   | Receive FIFO frame type indication<br>0: Data frame<br>1: Remote frame  |
| Bit 0     | Reserved    | 0x0         | resd | Kept at its default value   |

### Receive FIFO mailbox data length and time stamp register (CAN\_RFCx) (x = 0..1)

Note: All the receive mailbox registers are read-only.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | RFTS     | 0XXXXX      | ro   | Receive FIFO time stamp<br>Note: This field contains the value of the CAN timer sampled at the start of a receive frame. |
| Bit 15:8  | RFFMN    | 0xXX        | ro   | Receive FIFO filter match number<br>Note: This field contains the filter number that a message has passed through.       |
| Bit 7:4   | Reserved | 0xX         | resd | Kept at its default value  |

| Bit     | Name  | Reset value | Type | Description   |
|---------|-------|-------------|------|---|
| Bit 3:0 | RFDTL | 0xX         | ro   | Receive FIFO data length<br>Note: This field defines the data length of a receive message. A transmit message can contain from 0 to 8 data bytes. For a remote frame, its data length RFDTI is fixed 0. |

#### Receive FIFO mailbox data low register (CAN\_RFDTLx) (x = 0..1)

Note: All the receive mailbox registers are read-only.

| Bit       | Name  | Reset value | Type | Description              |
|-----------|-------|-------------|------|--------------------------|
| Bit 31:24 | RFDT3 | 0xXX        | ro   | Receive FIFO data byte 3 |
| Bit 23:16 | RFDT2 | 0xXX        | ro   | Receive FIFO data byte 2 |
| Bit 15:8  | RFDT1 | 0xXX        | ro   | Receive FIFO data byte 1 |
| Bit 7:0   | RFDT0 | 0xXX        | ro   | Receive FIFO data byte 0 |

#### Receive FIFO mailbox data high register (CAN\_RFDTHx) (x = 0..1)

Note: All the receive mailbox registers are read-only.

| Bit       | Name  | Reset value | Type | Description              |
|-----------|-------|-------------|------|--------------------------|
| Bit 31:24 | RFDT7 | 0xXX        | ro   | Receive FIFO data byte 7 |
| Bit 23:16 | RFDT6 | 0xXX        | ro   | Receive FIFO data byte 6 |
| Bit 15:8  | RFDT5 | 0xXX        | ro   | Receive FIFO data byte 5 |
| Bit 7:0   | RFDT4 | 0xXX        | ro   | Receive FIFO data byte 4 |

### CAN filter registers

#### CAN filter control register (CAN\_FCTRL)

Note: All the non-reserved bits of this register are controlled by software completely.

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:1 | Reserved | 0x150E0700  | resd | Kept at its default value   |
| Bit 0    | FCS      | 0x1         | rw   | Filter configuration switch<br>0: Disabled (Filter bank is active)<br>1: Enabled (Filter bank is in configuration mode)<br>Note: The initialization of the filter bank can be configured only when it is in configuration mode. |

#### CAN filter mode configuration register (CAN\_FCFG)

Note: This register can be written only when FCS = 1 in the CAN\_FCTRL register (The filter is in configuration mode)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:14 | Reserved | 0x00000     | resd | Kept at its default value  |
| Bit 13:0  | FMSELx   | 0x0000      | rw   | Filter mode select<br>Each bit corresponds to a filter bank.<br>0: Identifier mask mode<br>1: Identifier list mode |

### CAN filter bit width configuration register (CAN\_FBWCFG)

Note: This register can be written only when FCS = 1 in the CAN\_FCTRL register (The filter is in configuration mode)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:14 | Reserved | 0x00000     | resd | Kept at its default value   |
| Bit 13:0  | FBWSELx  | 0x0000      | rw   | Filter bit width select<br>Each bit corresponds to a filter bank.<br>0: Dual 16-bit<br>1: Single 32-bit |

### CAN filter FIFO association register (CAN\_FRF)

Note: This register can be written only when FCS = 1 in the CAN\_FCTRL register (The filter is in configuration mode)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:14 | Reserved | 0x00000     | resd | Kept at its default value   |
| Bit 13:0  | FRFSELx  | 0x0000      | rw   | Filter relation FIFO select<br>Each bit corresponds to a filter bank.<br>0: Associated with FIFO0<br>1: Associated with FIFO1 |

### CAN filter activation control register (CAN\_FACFG)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:14 | Reserved | 0x00000     | resd | Kept at its default value   |
| Bit 13:0  | FAENx    | 0x0000      | rw   | Filter active enable<br>Each bit corresponds to a filter bank.<br>0: Disabled<br>1: Enabled |

### CAN filter bank i filter bit register (CAN\_FiFBx) (i = 0..13; x = 1..2)

Note: There are 14 filter banks (i = 0..13). Each filter bank consists of two 32-bit registers, CAN\_FiFB[2:1]. This register can be modified only when the FAENx bit of the CAN\_FACFG register is cleared or the FCS bit of the CAN\_FCTRL register is set.

| Bit      | Name | Reset value | Type | Description   |
|----------|------|-------------|------|---|
| Bit 31:0 | FFDB | 0xFFFF XXXX | rw   | Filters filter data bit<br>Identifier list mode:<br>The configuration value of the register matches with the level of the corresponding bit of the data received on the bus (If it is a standard frame, the value of the corresponding bit of the extended frame is neglected.)<br>Identifier mark mode:<br>Only the bit with its register configuration value 1 can match with the level of the corresponding bit of the data received on the bus. It don't care when the register value is 0. |

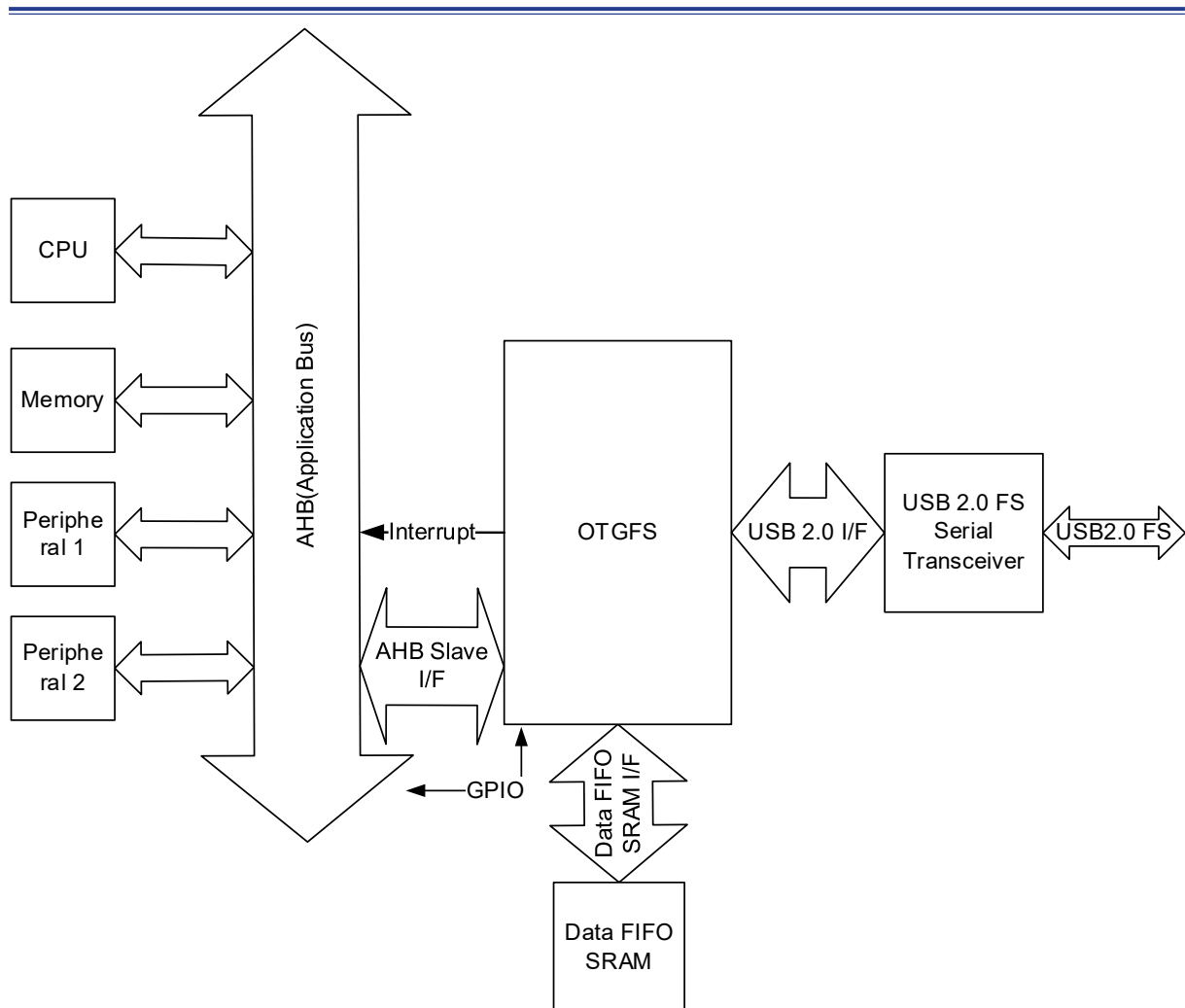


## 20 Universal serial bus full-speed device interface (OTGFS)

As a full-speed dual-role device, the OTGFS is fully compliant with the Universal Serial Bus Specification Revision 2.0.

### OTGFS structure

Figure 239 shows the block diagram of the OTGFS structure. The OTGFS module is connected to the AHB and has a dedicated SRAM of 1280 bytes.



**Figure 239. Block diagram of OTGFS structure**

## OTGFS functional description

The OTGFS module consists of an OTGFS controller, PHY and 1280-byte SRAM.

The OTGFS supports control transfer, bulk transfer, interrupt transfer and synchronous transfer.

The OTGFS is a USB full-speed dual role device controller. The status of the ID line determines whether the OTGFS acts as a host or device. When the ID line is floating, the OTGFS is used as a device. It is used as a host while the ID line is grounded. The internal 1.5 kΩ pull-up resistor and 15 kΩ pull-down resistor are available in the OTG PHY for the sake of dual role device.

In device mode, the OTGFS supports one bidirectional control endpoints, 7 IN endpoints, and 7 OUT endpoints; in host mode, the OTGFS supports 16 host channels.

The OTGFS supports SOF and OE pulse features: a SOF pulse generates at a SOF packet, the pulse can output to pins and the timer 2; an OE pulse generates when the OTGFS outputs data, the pulse can output to pins.

Suspend mode is supported. The OTGFS goes into power-saving mode after Suspend mode is entered.

As a device, a unified FIFO buffer is allocated for all OUT endpoints, and a separate FIFO buffer is provided to each of IN endpoints. As a host, a unified receive FIFO is allocated for all receive channels, a unified transmit FIFO for all non-periodic transmit channels, and a unified transmit FIFO for all periodic transmit channels.

OTGFS supports suspend mode. It enters this mode if a bus signal is not received within three minutes after the STOPPCLK bit is set in the OTGFS\_PCGCTL register. Besides, the PHY reception can be disabled by setting the LP\_MODE bit in the OTGS\_GCCFG register in order to reduce power consumption.

## OTGFS clock and pin configuration

### OTGFS clock configuration

The OTGFS interface has two clocks: USB control clock and APB bus clock. The USB full-speed device bus speed standard is 12 Mb/s±0.25 %, so it is necessary to supply 48 MHz±0.25 % for the OTGFS to perform USB bus sampling.

#### USBFS 48M clock has two sources:

##### ■ HICK 48M

When the HICK 48M clock is used as a USB control clock, it is recommended to enable ACC feature.

##### ■ Divided by PLL

The PLL output frequency must ensure that the USBDIV (see CRM\_CFG register) can be divided to 48 MHz.

Note: The APB clock frequency must be greater than 30 MHz when OTGFS is enabled.

## OTGFS pin configuration

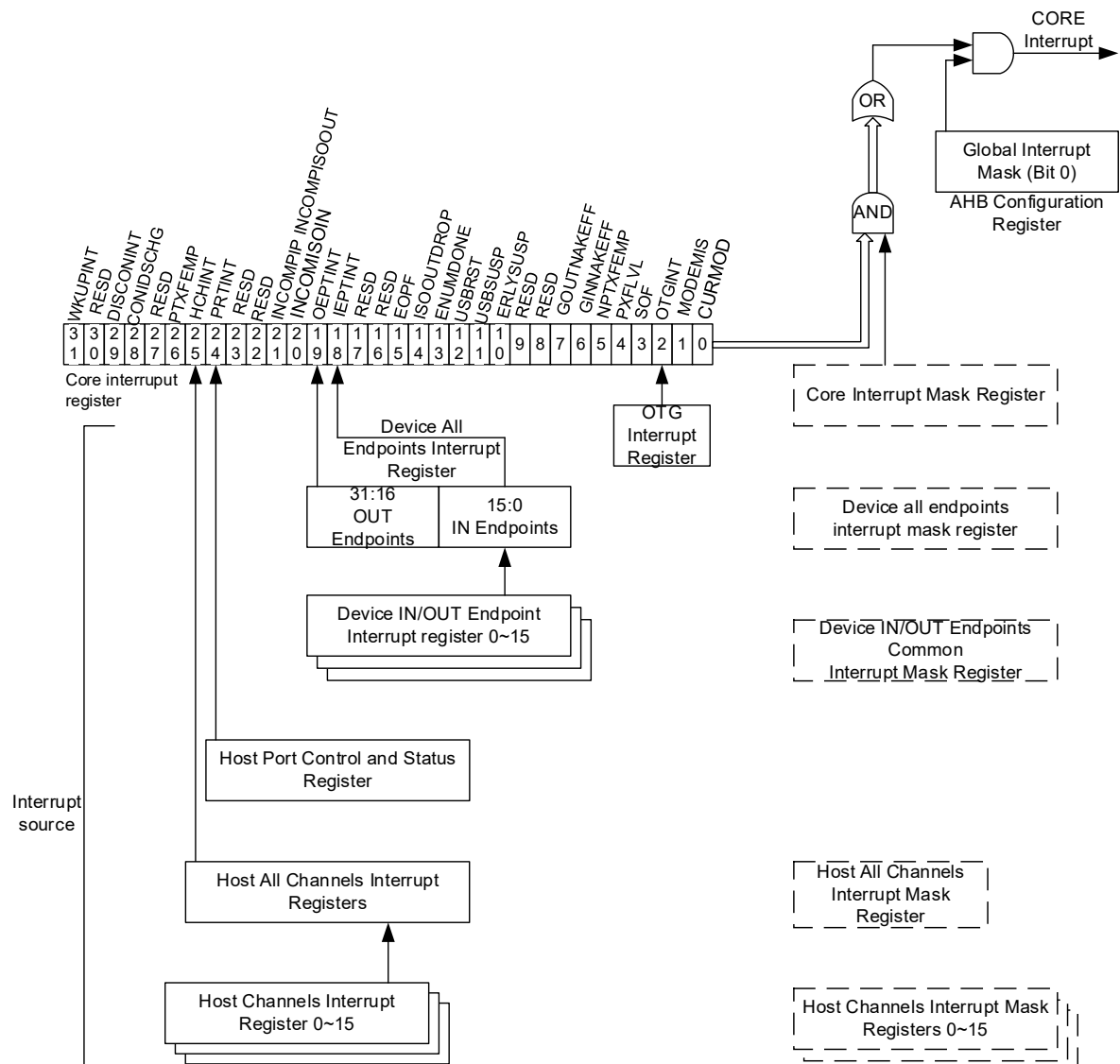
The OTGFS input/output pins are multiplexed with GPIOs. The GPIOs are used as OTGFS in one of the following conditions:

**Table 75. OTGFS input/output pins**

| Pin        | GPIO | Description  |
|------------|------|--|
| OTGFS_SOF  | PA8  | Enable OTG in CRM, and configure PA8 multiplexed function register as 0x3                                    |
| OTGFS_VBUS | PA9  | Configure PA9 as multiplexed function mode and PA9 multiplexed function register as 0x3                      |
| OTGFS_ID   | PA10 | Enable OTG in CRM, configure PA10 as multiplexed function mode and PA10 multiplexed function register as 0x3 |
| OTGFS_D-   | PA11 | Enable OTG in CRM, and PWRDOWN = 1   |
| OTGFS_D+   | PA12 | Enable OTG in CRM, and PWRDOWN = 1   |
| OTGFS1_OE  | PA4  | Enable OTG in CRM, and configure PA4 multiplexed function register as 0x2                                    |
|            | PA13 | Enable OTG in CRM, and configure PA13 multiplexed function register as 0x2                                   |
|            | PA15 | Enable OTG in CRM, and configure PA15 multiplexed function register as 0x5                                   |
|            | PC9  | Enable OTG in CRM, and configure PC9 multiplexed function register as 0x5                                    |

## OTGFS interrupts

Figure 240 shows the OTGFS interrupt hierarchy. Refer to the OTGFS interrupt register (OTGFS\_GINTSTS) and OTGFS interrupt mask register (OTGFS\_GINTMSK).



Note Because an interrupt mask only masks an interrupt, software must clear an interrupt before unmasking it, to avoid servicing an old interrupt.

**Figure 240. OTGFS interrupt hierarchy**

## OTGFS functional description

### OTGFS initialization

If the cable is connected during power-on, the current operation mode bit (CURMOD bit) in the controller interrupt register indicates the mode. The OTGFS controller enters host mode when A-type plug is connected or device mode when a B-type plug is connected.

This section explains the initialization of the OTGFS controller after power-on. The application must follow the initialization sequence, however in host or device mode. All controller global registers are initialized according to the controller configuration.

1. Program the following fields in the global AHB configuration register:
  - Global interrupt mask bit = 0x1
  - Non-periodic transmit FIFO empty level
  - Periodic transmit FIFO empty level
2. Program the following fields in the global AHB configuration register:
  - OTGFS\_GINTMSK.RXFLVLMSK = 0x0
3. Program the following fields in the OTGFS\_GUSBCFG register:
  - Full-speed timeout standard bit
  - USB turnaround time bit
4. The software must unmask the following bits in the OTGFS\_GINTMSK register:
  - OTG interrupt mask
  - Mode mismatch interrupt mask
5. The software can read the CURMOD bit in the OTGFS\_GINTSTS register to determine whether the OTGFS controller is operating in host or device mode.

## OTGFS FIFO configuration

### Device mode

A dynamic FIFO allocation is required during power-on or USB reset. In device mode, the application must meet the following conditions before modifying FIFO SRAM allocation.

- OTGFS\_DIEPCTLx / OTGFS\_DOEPCTLx.EPENA = 0x0
- OTGFS\_DIEPCTLx / OTGFS\_DOEPCTLx.NAKSTS = 0x1

The TXFNUM bit in the OTGFS\_GRSTCTL register is used to refresh the controller transmit FIFO. Refer to Section Refresh controller transmit FIFO for more information.

Attention should be paid to the following information during FIFO SRAM allocation:

#### (1) Receive FIFO SRAM allocation

- SRAM for SETUP Packets: 13 DWORDs must be reserved in the receive FIFO to receive one SETUP Packet on control endpoint. The controller does not use these locations, which are reserved for SETUP packets.
- One DWORD is to be reserved for global OUT NAK
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size/4) + 1 must be allocated to receive data packets. In most cases, two (largest packet size/4) + 1 spaces are recommended so that the USB can receive the subsequent packet while the previous packet is being transferred to the AHB. If there is a longer latency on AHB, sufficient spaces must be reserved to receive multiple packets in order to prevent synchronous data packet loss.
- Transfer complete status information, along with the last packet for each endpoint, is also pushed to the FIFO

- One location must be reserved for the disable status bit of each endpoint
- Typically, two DWORDs for each OUT endpoint are recommended.

## (2) Transmit FIFO SRAM allocation

The minimum SRAM space required for each IN endpoint transmit FIFO is the maximum data packet size for that particular IN endpoint. The more the space allocated to the transmit IN endpoint FIFO, the better the USB performance, and this helps to avoid latency on the AHB line.

**Table 76. OTGFS transmit FIFO SRAM allocation**

| FIFO name       | SRAM size   |
|-----------------|---|
| Receive FIFO    | rx_fifo_size, including setup packets, OUT endpoint control information and OUT data packets. |
| Transmit FIFO 0 | tx_fifo_size[0]   |
| Transmit FIFO 1 | tx_fifo_size[1]   |
| Transmit FIFO 2 | tx_fifo_size[2]   |
| .....           | .....   |
| Transmit FIFO i | tx_fifo_size[i]   |

Configure the following registers according to the above mentioned:

1. OTGFS receive FIFO size register (OTGFS\_GRXFSIZ)
  - OTGFS\_GRXFSIZ.RXFDEP = rx\_fifo\_size
2. Endpoint 0 TX FIFO size register (OTGFS\_DIEPTXF0)
  - OTGFS\_DIEPTXF0.INEPT0TXDEP = tx\_fifo\_size[0];
  - OTGFS\_DIEPTXF0.INEPT0TXSTADDR = rx\_fifo\_size;
3. Device IN endpoint transmit FIFO#1 size register (OTGFS\_DIEPTXF1)
  - OTGFS\_DIEPTXF1.INEPTXFSTADDR = OTGFS\_DIEPTXF0.INEPT0TXSTADDR + tx\_fifo\_size[0];
4. Device IN endpoint transmit FIFO#2 size register (OTGFS\_DIEPTXF2)
  - OTGFS\_DIEPTXF2.INEPTXFSTADDR = OTGFS\_DIEPTXF1.INEPTXFSTADDR + tx\_fifo\_size[1]
5. Device IN endpoint transmit FIFO#i size register (OTGFS\_DIEPTXF<sub>i</sub>)
  - OTGFS\_DIEPTXF<sub>i</sub>.INEPTXFSTADDR = OTGFS\_DIEPTXF<sub>i-1</sub>.INEPTXFSTADDR + tx\_fifo\_size[i-1]
6. After SRAM allocation, refresh transmit FIFO and receive FIFO to ensure normal FIFO running.
  - OTGFS\_GRSTCTL.TXFNUM = 0x10
  - OTGFS\_GRSTCTL.TXFFLSH = 0x1
  - OTGFS\_GRSTCTL.RXFFLSH = 0x1

The application cannot perform other operations on the controller until the TXFFLSH and RXFFLSH bits are cleared.

## Host mode

In host mode, the application must confirm the following status before changing FIFO SRAM allocation:

- All channels have been disabled
- All FIFOs are empty

After FIFO SRAM allocation is complete, the application must refresh all FIFOs in the controller through the TXFNUM bit in the OTGFS\_GRSTCTL register.

After allocation, the FIFO pointers must be reset by refreshing operation to ensure normal FIFO running. Refer to Section Refresh controller transmit FIFO for more information.

### (1) Receive FIFO SRAM allocation

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of  $(\text{largest packet size}/4) + 2$  must be allocated to receive data packets. If more synchronous endpoints are enabled, then at least two  $(\text{largest packet size}/4) + 2$  spaces must be allocated to receive back-to-back packets. In most cases, two  $(\text{largest packet size}/4) + 2$  spaces are recommended so that the USB can receive the subsequent packet while the previous packet is being transferred to the AHB. If there is a longer latency on AHB, sufficient spaces must be reserved to receive multiple packets in order to prevent synchronous data packet loss.

Transfer complete status information and channel abort information, along with the last packet in the host channel is also pushed to the FIFO. Thus, two DWORDs must be allocated for this.

### (2) Transmit FIFO SRAM allocation

The minimum SRAM space required for the host non-periodic transmit FIFO is the largest packet size of all non-periodic OUT channels. The more the space allocated to the non-periodic FIFO, the better the USB performance, and this helps to avoid latency on the AHB line. Typically, two largest packet sizes of space is recommended so that the AHB can get the next data packet while the current packet is being transferred to the USB. If there is a longer latency on AHB, sufficient spaces must be reserved to receive multiple packets in order to prevent synchronous data packet loss.

### (3) Internal storage space allocation

**Table 77. OTGFS internal storage space allocation**

| FIFO Name                  | Data SRAM Size  |
|----------------------------|-----------------|
| Receive FIFO               | rx_fifo_size    |
| Non-periodic transmit FIFO | tx_fifo_size[0] |
| Periodic transmit FIFO     | tx_fifo_size[1] |

Configure the following registers according to the above mentioned:

1. OTGFS receive FIFO size register (OTGFS\_GRXFSIZ)
  - OTGFS\_GRXFSIZ.RXFDEP = rx\_fifo\_size
2. OTGFS Non-periodic TX FIFO size register (OTGFS\_GNPTXFSIZ)
  - OTGFS\_GNPTXFSIZ.NPTXFDEP = tx\_fifo\_size[0]
  - OTGFS\_GNPTXFSIZ.NPTXFSTADDR = rx\_fifo\_size
3. OTGFS host periodic transmit FIFO size register (OTGFS\_HPTXFSIZ)
  - OTGFS\_HPTXFSIZ.PTXFSIZE = tx\_fifo\_size[1]
  - OTGFS\_HPTXFSIZ.PTXFSTADDR = OTGFS\_GNPTXFSIZ.NPTXFSTADDR + tx\_fifo\_size[0]

4. After SRAM allocation, refresh transmit FIFO and receive FIFO to ensure normal FIFO running.

- OTGFS\_GRSTCTL.TXFNUM = 0x10
- OTGFS\_GRSTCTL.TXFFLSH = 0x1
- OTGFS\_GRSTCTL.RXFFLSH = 0x1
- The application cannot perform other operations on the controller until the TXFFLSH and RXFFLSH bits are cleared.

#### Refresh controller transmit FIFO

The application refreshes all transmit FIFOs through the TXFFLSH bit in the OTGFS\_GRSTCTL register:

- Check whether GINNAKEFF = 0 or not in the OTGFS\_GINTSTS register. If this bit has been cleared, write 0x1 to the OTGFS\_DCTL.SGPNINNAK register. When the NACK valid interrupt is set, it means that the controller does not read FIFO.
- Wait until GINNAKEFF = 0x1 in the OTGFS\_GINTSTS register, indicating that the NAK configuration has taken effect for all IN endpoints.
- Poll the OTGFS\_GRSTCTL register and wait until AHBIDLE = 1. AHBIDLE = H indicates that the controller does not write the FIFO.
- Confirm whether TXFFLSH = 0x0 or not in the OTGFS\_GRSTCTL register. If TXFFLSH is cleared, write the transmit FIFO number to be refreshed into the OTGFS\_GRSTCTL.TXFNUM register.
- Set TXFFLSH = 0x1 in the OTGFS\_GRSTCTL register, and wait until it is cleared.
- Set the CGNPINNAK bit in the OTGFS\_DCTL register.

### OTGFS host mode

#### Host initialization

The following steps must be respected to initialize the controller:

1. Unmask interrupt through the PRTINTMSK bit in the OTGFS\_GINTMSK register.
2. Program the OTGFS\_HCFG register.
3. Set PRTPWR = 0x1 in the OTGFS\_HPRT register to drive VBUS supply on the USB.
4. Wait until that the PRTCONDET bit is set in the OTGFS\_HPRT0 register, indicating that the device is connected to the port.
5. Set PRTRST = 0x1 in the OTGFS\_HPRT register to issue a reset operation.
6. Wait for at least 10 ms to ensure the completion of the reset.
7. Set PRTRST = 0x0 in the OTGFS\_HPRT register.
8. Wait for the interrupt (PRTENCHNG bit in the OTGFS\_HPRT register).
9. Read the PRTSPD bit in the OTGFS\_HPRT register to get the enumeration speed.
10. Configure the HFIR register according to the selected PHY clock value.
11. Select the size of the receive FIFO by setting the OTGFS\_GRXFSIZ register.
12. Select the start address and size of the non-periodic transmit FIFO by setting the OTGFS\_GNPTXFSIZ register.



13. Select the start address and size of the periodic transmit FIFO by setting the OTGFS\_HPTXFSIZ register.

To communicate with the device, the application must enable and initialize at least one channel according to OTGFS channel initialization requirements.

### OTGFS channel initialization

To communicate with the device, the application must enable and initialize at least one channel according to the following steps:

1. Unmask the following interrupts by setting the OTGFS\_GINTMSK register:
  - Non-periodic transmit FIFO empty for OUT transfers
  - Non-periodic transmit FIFO half empty for OUT transfers
2. Unmask the interrupts of the selected channels by setting the OTGFS\_HAINTMSK register.
3. Unmask the transfer-related interrupts in the host channel interrupt register by setting the OTGFS\_HCINTMSKx register.
4. Configure the total transfer size (in bytes), and the expected number of the packets (including short packets) for the OTGFS\_HCTSIZx register of the selected channel. The application must configure the PID bit according to the initial data PID (it is the PID on the first OUT transfer, or to be received from the first IN transfer).
5. Configure the transfer size to ensure that the transfer size of the channel is a multiple of the largest packet size.
6. Configure the OTGFS\_HCCHARx register of the selected channel according to the device endpoint characteristics such as type, speed and direction (the channel cannot be enabled by setting the enable bit until the application is ready for packet transfer or reception).

### Halting a channel

The application can disable a channel by writing 0x1 to the CHDIS and CHENA bits in the OTGFS\_HCCHARx register. This enables the host to refresh the summited requests (if any) and generates a channel halted interrupt. The application cannot re-allocate channels for other transactions until an interrupt is generated in the OTGFS\_HCINTx register (CHHLTD bit). Those transactions that have already been started on the USB line are not interrupted by the host.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can refresh the submitted requests when the request queue is full (before disabling the channel) by setting CHDIS = 0x1, and CHENA = 0 in the OTGFS\_HCCHARx register.

When there is a transaction input in the request queue, the controller will trigger a RXFLVL interrupt. The application must generate a channel halted interrupt through the OTGFS\_GRXSTSP register.

The application is expected to abort a channel on any of the following conditions:

- When an interrupt (XFERC bit) is received in the OTGFS\_HCINTx register during a non-periodic IN transfer.
- When an STALL, XACTERR, BBLERR or DTGLERR interrupt in the OTGFS\_HCINTx register is received for an IN or OUT channel.

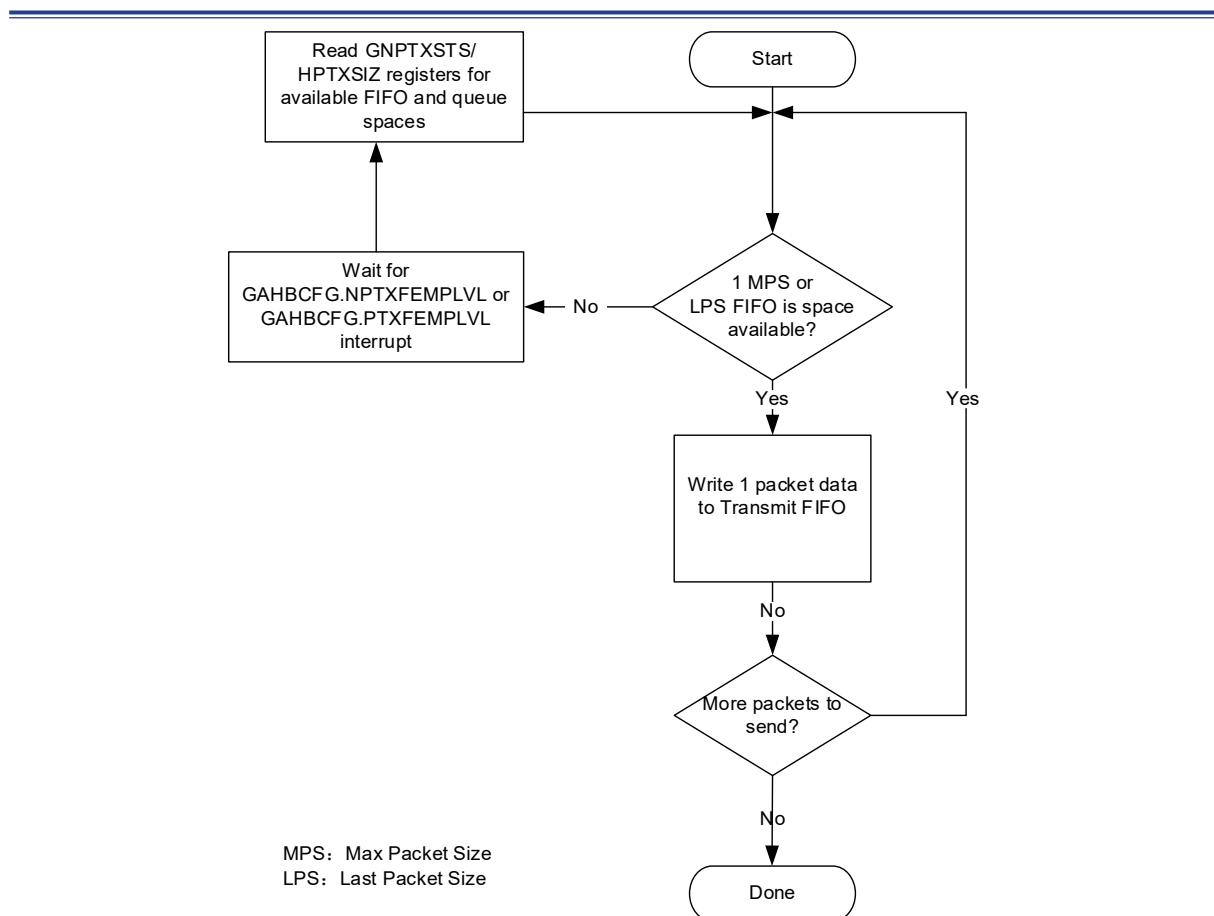
- When a DISCONINT (device disconnected) interrupt event is received in the OTGFS\_GINTSTS register, the application must check the PRTCONSTS bit in the OTGFS\_HPRT register. This is because when the device is disconnected with the host, the PRTCONSTS bit will be reset in the OTGFS\_HPRT register. The application must initiate a software reset to ensure that all channels have been cleared. Once the device is reconnected, the host must start a USB reset.
- When the application needs to abort a transfer before normal completion

### Queue depth

Up to 8 interrupt and synchronous transfer requests are supported in the periodic hardware transfer request queue; while up to 8 control and bulk transfer requests are allowed in the non-periodic hardware transfer request queue.

### Writing the transmit FIFO

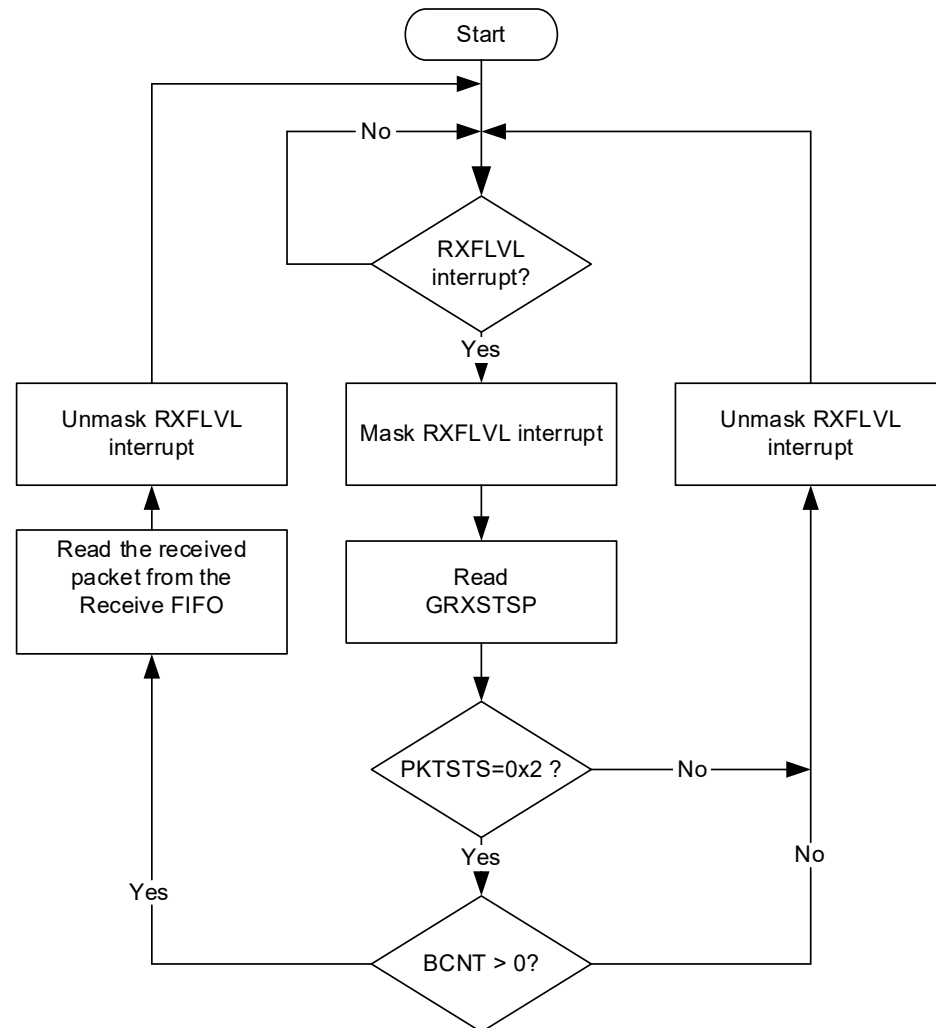
Figure 241 shows the flow chart of writing the transmit FIFO. The OTGFS host automatically writes a request (OUT request) to the periodic/non-periodic request queue when writing the last one WORD packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in WORDs. If the packet size is not aligned with WORD, the application must use padding. The OTGFS host determines the actual packet size according to the programmed maximum packet size and transfer size.



**Figure 241. Writing the transmit FIFO**

## Reading the receive FIFO

Figure 242 shows the flow chart of reading the receive FIFO. The application must ignore all packet statuses other than IN data packet (0x0010).



**Figure 242. Reading the receive FIFO**

## Special cases

### (1) Handling babble conditions

The OTGFS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more than the largest packet size for the channel. Port babble occurs if the controller continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

When the OTGFS controller detects a packet babble, it stops writing data to the receiver buffer and waits for the completion of packet. When it detects the end of packet, the OTGFS flushes the data already written in the receiver buffer and generates a babble interrupt.

When the OTGFS controller detects a port babble, it flushes the receive FIFO and disables the port. Then the controller generates a Port disable interrupt. Once receiving the interrupt, the application must determine that this is not caused by an overcurrent condition (another cause of the port disable interrupt) by checking the PRTOVRACT bit in the OTGFS\_HPRT register, then perform a software reset. The controller does not send any more tokens if a port babble signal is detected.

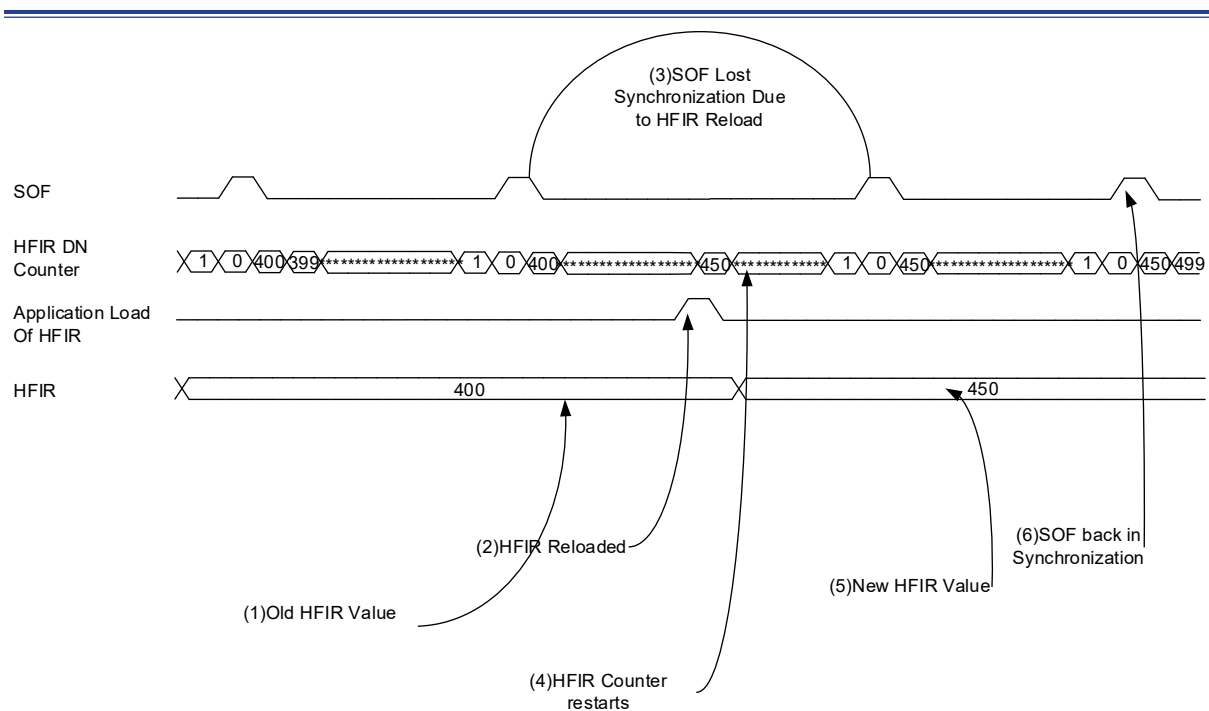
## (2) Handling device disconnected conditions

If the device is suddenly disconnected, an interrupt is generated on a disconnect event (DISCONINT bit in the OTGFS\_GINTSTS register). Upon receiving this interrupt, the application must start a software reset through the CSFTRST in the OTGFS\_GRSTCTL register.

### Host HFIR feature

The host frame interval register (HFIR) defines the interval between two consecutive SOFs (full-speed) or Keep-Alive tokens. This field contains the number of PHY clock for the required frame interval. This is mainly used to adjust the SOF duration based on PHY clock frequencies.

Figure 243 shows the HFIR behavior when the HFIRRLDCTRL is set to 0x0 in the OTGFS\_HFIR register.

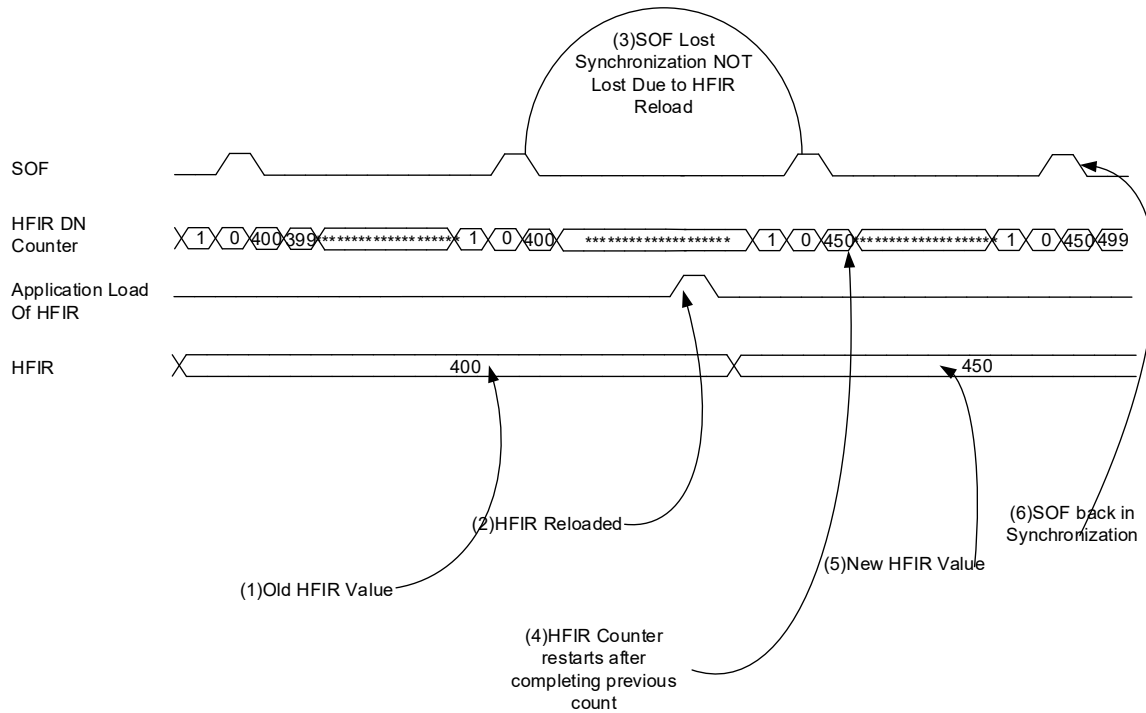


**Figure 243. HFIR behavior when HFIRRLDCTRL = 0x0**

The sequence of operation is as follows:

1. After power-on reset, the current HFIR value set by the application is shown.
2. The application loads a new value into the HFIR register.
3. The HFIR downcounter is reloaded, so it will immediately restart counting to cause SOF synchronization loss.
4. Restart HFIR counter.

5. The HFIR register receives a new programmed value.
  6. Obtain SOF synchronization again after the first SOF is generated using the HFIR new feature.
- Figure 244 shows the HFIR behavior when HFIRRLDCTRL = 0x1 in the OTGFS\_HFIR register.



**Figure 244. HFIR behavior when HFIRRLDCTRL = 0x1**

The sequence of operation is as follows:

1. After power-on reset, the current HFIR value set by the application is shown
2. The application loads a new HFIR value; the HFIR counter does not apply this new value, but continues counting until it reaches 0
3. The counter generates a SOF when it reaches 0 using the old HFIR value
4. The HFIR counter applies a new value
5. New HFIR value takes effect

The SOF synchronization resumes after going through above-mentioned steps.

#### Initialize bulk and control IN transfers

Figure 245 shows a typical bulk or control IN transfer operation. Refer to channel 2 (ch\_2) for more information. The assumptions are as follows:

- The application is attempting to receive two largest-packet-size packets (transfer size is 64 bytes).
- The receive FIFO contains at least one largest-packet-size packet and two status WORDs per each packet (72 bytes for full-speed transfer).
- The non-periodic request queue depth is 4.

## (1) Operation process for common bulk and control IN transfers

The sequence of operations shown in Figure 245 is as follows:

1. Initialize channel 2 (according to OTGFS channel initialization requirements).
2. Set the CHENA bit in the OTGFS\_HCCHAR2 register to write an IN request to the non-periodic request queue.
3. The controller issues an IN token after completing the current OUT transfer.
4. The controller generates a RXFLVL interrupt as soon as the receive packet is written into the receive FIFO.
5. To handle the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, and then read the receive FIFO. Following this step to unmask the RXFLVL interrupt.
6. The controller generates the RXFLVL interrupt when the transfer complete status is written into the receive FIFO.
7. The application must read the receive packet status, and ignore it when the receive packet status is not an IN data packet.
8. The controller generates the XFERRC interrupt as soon as the receive packet is read.
9. To handle the XFERRC interrupt, disable the channel (see Halting a channel) and stop writing the OTGFS\_HCCHAR2 register. The controller writes a channel halted request to the non-periodic request queue once the OTGFS\_HCCHAR2 register is written.
10. The controller generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The controller generates a CHHLTD interrupt as soon as the halt status is read from the receive FIFO.
13. In response to the CHHLTD interrupt, the processor does not allocate the channel for other transfers.

## (2) Handling interrupts

The following code describes the interrupt service routine related to the channel during bulk and control IN transfers

```
Unmask (XACTERR/XFERC/BBLERR/STALL/DATATGLERR)
if (XFERC)
{
    Reset Error Count
    Unmask CHHLTD
    Disable Channel
    Reset Error Count
    Mask ACK
}
else if (XACTERR or BBLERR or STALL)
{
    Unmask CHHLTD
    Disable Channel
    if (XACTERR)
    {
        Increment Error Count
        Unmask ACK
    }
}
```

```

    }
}
else if (ChHltd)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
else if (DATATGLERR)
{
    Reset Error Count
}

```

### Initialize bulk and control OUT/SETUP transfers

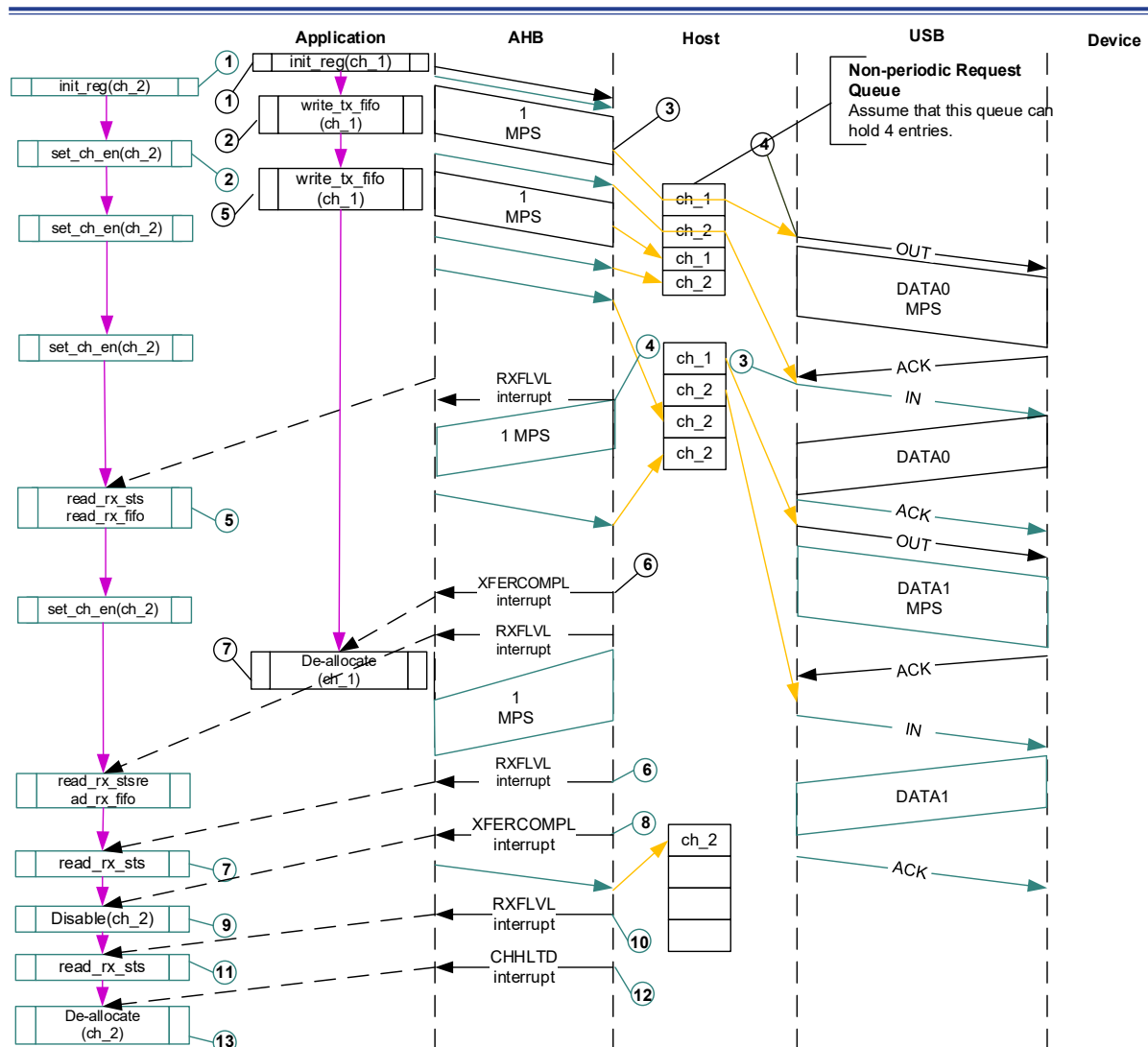
Figure 245 shows a typical bulk or control transfer OUT/SETUP transfer operation. Refer to channel 1 (ch\_1) for more information. It is necessary to send two bulk transfer OUT packets. The control transfer SETUP operation is the same, just the fact that it has only one packet. The assumptions are as follows:

- The application is attempting to send two largest-packet-size packets (transfer size is 64 bytes).
- The non-periodic transmit FIFO can store two packets (128 bytes for full-speed transfer).
- The non-periodic request queue depth is 4.

#### (1) OUT/SETUP operation process for common bulk and control transfer

The sequence of operations shown in Figure 245 is as follows:

1. Initialize channel 1 (according to OTGFS channel initialization requirements).
2. Write the first packet for channel 1.
3. Along with the last WORD write, the controller writes a request to the non-periodic request queue.
4. The controller sends an OUT token in the current frame as soon as the non-periodic queue becomes empty.
5. Write the second packet (the last one) to the channel 1.
6. The controller generate an XFERC interrupt as soon as the previous transfer is completed successfully.
7. In response to the XFERC interrupt, the processor does not allocate the channel for other transfers.



**Figure 245. Example of common Bulk/Control OUT/SETUP and Bulk/Control IN transfer**

## (2) Handling interrupts

The following code describes the interrupt service routine related to the channel during bulk and control transfer OUT/SETUP operation.

```
Unmask (NAK/XACTERR/NYET/STALL/XFERC)
if (XFERC)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else if (STALL)
{
    Transfer Done = 1
    Unmask CHHLTD
    Disable Channel
}
```



```

    }
    else if (NAK or XACTERR or NYET)
    {
        Rewind Buffer Pointers
        Unmask CHHLTD
        Disable Channel
        if (XactErr)
        {
            Increment Error Count
            Unmask ACK
        }
        else
        {
            Reset Error Count
        }
    }
    else if (CHHLTD)
    {
        Mask CHHLTD
        if (Transfer Done or (Error_count == 3))
        {
            De-allocate Channel
        }
        else
        {
            Re-initialize Channel (Do ping protocol for HS)
        }
    }
    else if (ACK)
    {
        Reset Error Count
        Mask ACK
    }
}

```

- Note:** 1. The application can only write the transmit FIFO when the transmit FIFO and request queue has free spaces. The application must check whether there is a free space in the transmit FIFO through the NPTXFEMP bit in the OTGFS\_GINTSTS register.
2. The application can only write a request when the request queue has free spaces and wait until an XFERRC interrupt is received.

### Initialize interrupt IN transfers

Figure 246 shows the operation process of a typical interrupt IN transfer. Refer to channel 2 (ch\_2). The assumptions are as follows:

- The application is attempting to receive one largest-packet-size packet (transfer size is 64 bytes) from an odd frame.
- The receive FIFO can store at least one largest-packet-size packet and two status WORDs per packet (1031 bytes for full-speed transfer).
- The periodic request queue depth is 4.

#### (1) Common interrupt IN operation process

The sequence of operations shown in Figure 246 (channel 2) is as follows:

1. Initialize channel 2 (according to OTGFS channel initialization requirements). The application must set the ODDFRM bit in the OTGFS\_HCCHAR2 register.

2. Set the CHENA bit in the OTGFS\_HCCHAR2 register to write an IN request to the periodic request queue.
3. The OTGFS host writes an IN request to the periodic request queue each time the CHENA is set in the OTGFS\_HCCHAR2 register.
4. The OTGFS host attempts to send an IN token in the next frame (odd).
5. The OTGFS host generates a RXFLVL interrupt as soon as an IN packet is received and written to the receive FIFO.
6. To handle the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask the interrupt after reading the entire packet.
7. The controller generates the RXFLVL interrupt when the transfer complete status is written to the receive FIFO. The application must read and ignore the receive packet when the receive packet is not an IN packet.
8. The controller generates an XFERC interrupt as soon as the receive packet is read.
9. To handle the XFERC interrupt, read the PKTCN bit in the OTGFS\_HCTSIZ2 register. If the PKTCNT bit in the OTGFS\_HCTSIZ2 is not equal to 0, disable the channel before re-initializing the channel for the next transfer. If PKTCNT == 0 in the OTGFS\_HCTSIZ2 register, re-initialize the channel for the next transfer. In this case, the application must reset the ODDFRM bit in the OTGFS\_HCCHAR2 register.

## (2) Handling interrupts

The following code describes the interrupt service routine related to the channel during interrupt IN transfer

```
Unmask (NAK/XACTERR/XFERC/BBLERR/STALL/FRMOVRUN/DATATGLERR)
if (XFERC)
{
    Reset Error Count
    Mask ACK
    if (HCTSIZx.PKTCNT == 0)
    {
        De-allocate Channel
    }
    else
    {
        Transfer Done = 1
        Unmask CHHLTD
        Disable Channel
    }
}
else if (STALL or FRMOVRUN or NAK or DATATGLERR or BBLERR)
{
    Mask ACK
    Unmask CHHLTD
    Disable Channel
    if (STALL or BBLERR)
    {
        Reset Error Count
        Transfer Done = 1
    }
    else if (!FRMOVRUN)
    {
        Reset Error Count
    }
}
```

```

    }
}
else if (XACTERR)
{
    Increment Error Count
    Unmask ACK
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else Re-initialize Channel (in next b_interval - 1 uF/F)
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}

```

The application can only write a request to the same channel when the remaining space in the request queue reaches the number defined in the MC field, before switching to other channels (if any).

### Initialize interrupt OUT transfers

Figure 246 shows a typical interrupt OUT transfer operation. Refer to channel 1 (ch\_1). The assumptions are as follows:

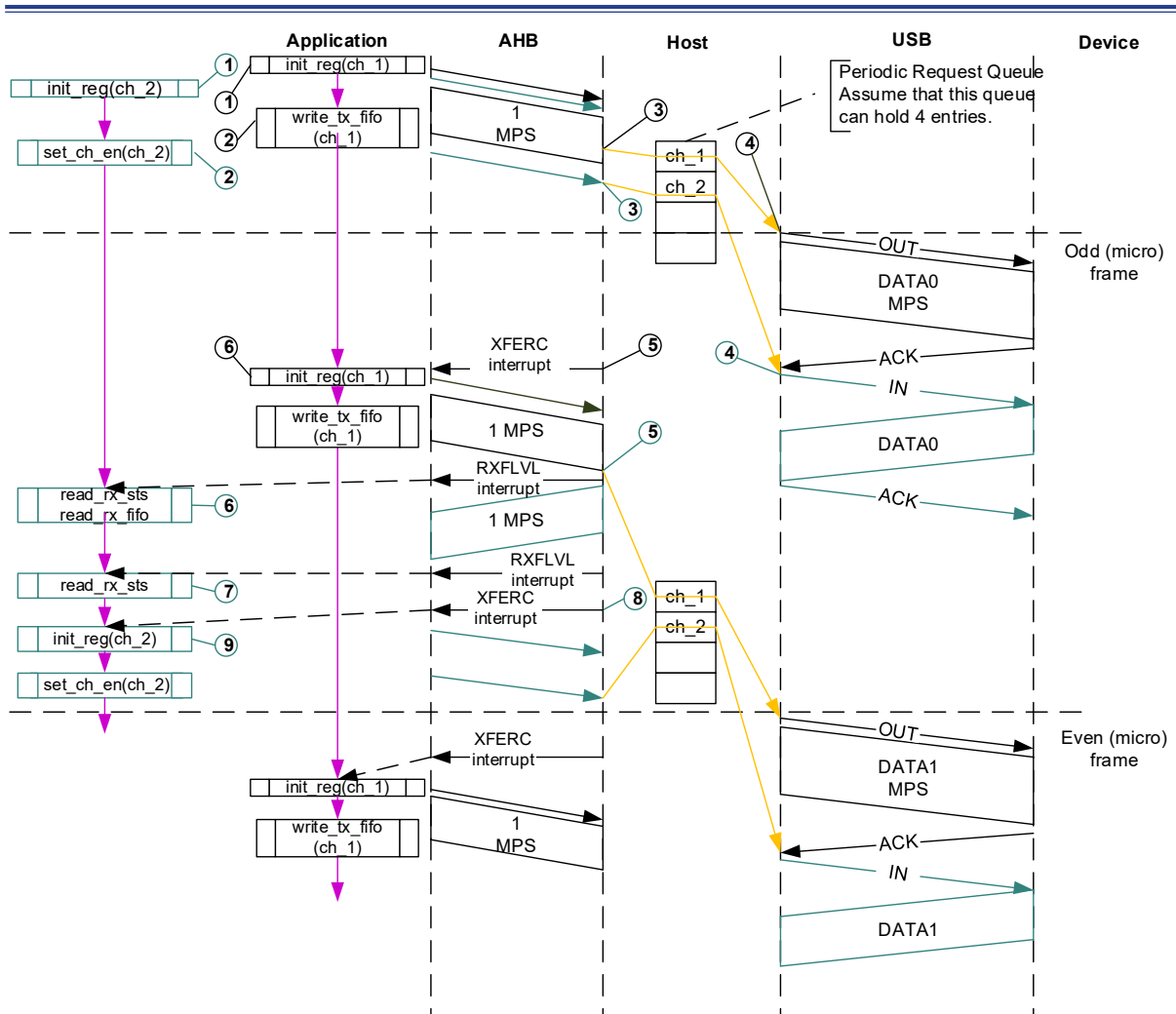
- The application is attempting to send one largest-packet-size packet (transfer size is 64 bytes) to every frame.
- The periodic transmit FIFO can store one packet (1KB bytes for full-speed transfer).
- The periodic request queue depth is 4.

#### (1) Common interrupt IN operation process

The sequence of operations shown in 246 (channel 1) is as follows:

1. Initialize channel 1 (according to OTGFS channel initialization requirements). The application must set the ODDFRM bit in the OTGFS\_HCCHAR2 register.
2. Write the first packet to the channel 1.
3. Along with the last WORD write of each packet, the host writes a request to the periodic request queue.
4. The host sends an OUT token in the next frame (odd).
5. The host generates an XFERC interrupt after the last packet is transmitted successfully.
6. In response to the XFERC interrupt, re-initialize the channel for the next transfer.

## (2) Handling interrupts



**Figure 246. Example of common interrupt OUT/IN transfers**

The following code describes the interrupt service routine related to the channel during interrupt OUT transfers.

```
Unmask (NAK/XACTERR/STALL/XFERC/FRMOVRUN)
if (XFERC)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else if (STALL or FRMOVRUN)
{
    Mask ACK
    Unmask CHHLTD
    Disable Channel
    if (STALL)
    {
```

```

        Transfer Done = 1
    }
}
else if (NAK or XACTERR)
{
    Rewind Buffer Pointers
    Reset Error Count
    Mask ACK
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel (in next b_interval - 1 uF/F)
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}

```

Before switching to other channels (if any), the application can only write packets based on the number defined in the MC filed to the transmit FIFO and request queue when the transmit FIFO has free spaces. The application can determine whether the transmit FIFO has free spaces through the NPTXFEMP bit in the OTGFS\_GINTSTS register.

### Initialize synchronous IN transfers

Figure 247 shows the operation process of a typical synchronous IN transfer. Refer to channel 2 (ch\_2). The assumptions are as follows:

- The application is attempting to receive one largest-packet-size packet (transfer size is 1023 bytes) from the next odd frame.
- The receive FIFO can store at least one largest-packet-size packet and two status WORDs per packet (1031 bytes for full-speed transfer).
- The periodic request queue depth is 4.

#### (1) Common interrupt IN operation process

The sequence of operations shown in Figure 247 (channel 2) is as follows:

1. Initialize channel 2 (according to OTGFS channel initialization requirements). The application must set the ODDFRM bit in the OTGFS\_HCCHAR2 register.
2. Set the CHENA bit in the OTGFS\_HCCHAR2 register to write an IN request to the periodic request queue.
3. The OTGFS host writes an IN request to the periodic request queue each time the CHENA is set in the OTGFS\_HCCHAR2 register.
4. The OTGFS host attempts to send an IN token in the next frame (odd).

5. The OTGFS host generates a RXFLVL interrupt as soon as an IN packet is received and written to the receive FIFO.
6. To handle the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask the interrupt after reading the entire packet.
7. The controller generates the RXFLVL interrupt when the transfer complete status is written to the receive FIFO. The application must read and ignore the receive packet when the receive packet is not an IN packet (GRXSTSR.PKTSTS! = 0x0010).
8. The controller generates an XFERC interrupt as soon as the receive packet is read.
9. To handle the XFERC interrupt, read the PKTCN bit in the OTGFS\_HCTSIZ2 register. If the PKTCNT bit in the OTGFS\_HCTSIZ2 is not equal to 0, disable the channel before re-initializing the channel for the next transfer. If PKTCNT == 0 in the OTGFS\_HCTSIZ2 register, re-initialize the channel for the next transfer. In this case, the application must reset the ODDFRM bit in the OTGFS\_HCCHAR2 register.

## (2) Handling interrupts

The following code describes the interrupt service routine related to the channel during synchronous IN transfers.

```
Unmask (XACTERR/XFERC/FRMOVRUN/BBLERR)
if (XFERC or FRMOVRUN)
{
    if (XFERC and (HCTSIZx.PKTCNT == 0))
    {
        Reset Error Count
        De-allocate Channel
    }
    else
    {
        Unmask CHHLTD
        Disable Channel
    }
}
else if (XACTERR or BBLERR)
{
    Increment Error Count
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}
```

### Initialize synchronous OUT transfers

Figure 247 shows a typical synchronous OUT transfer operation. Refer to channel 1 (ch\_1). The assumptions are as follows:

- The application is attempting to send one largest-packet-size packet (transfer size is 1023 bytes) to every frame from the next odd frame.
- The periodic transmit FIFO can store one packet (1KB bytes for full-speed transfer).
- The periodic request queue depth is 4.

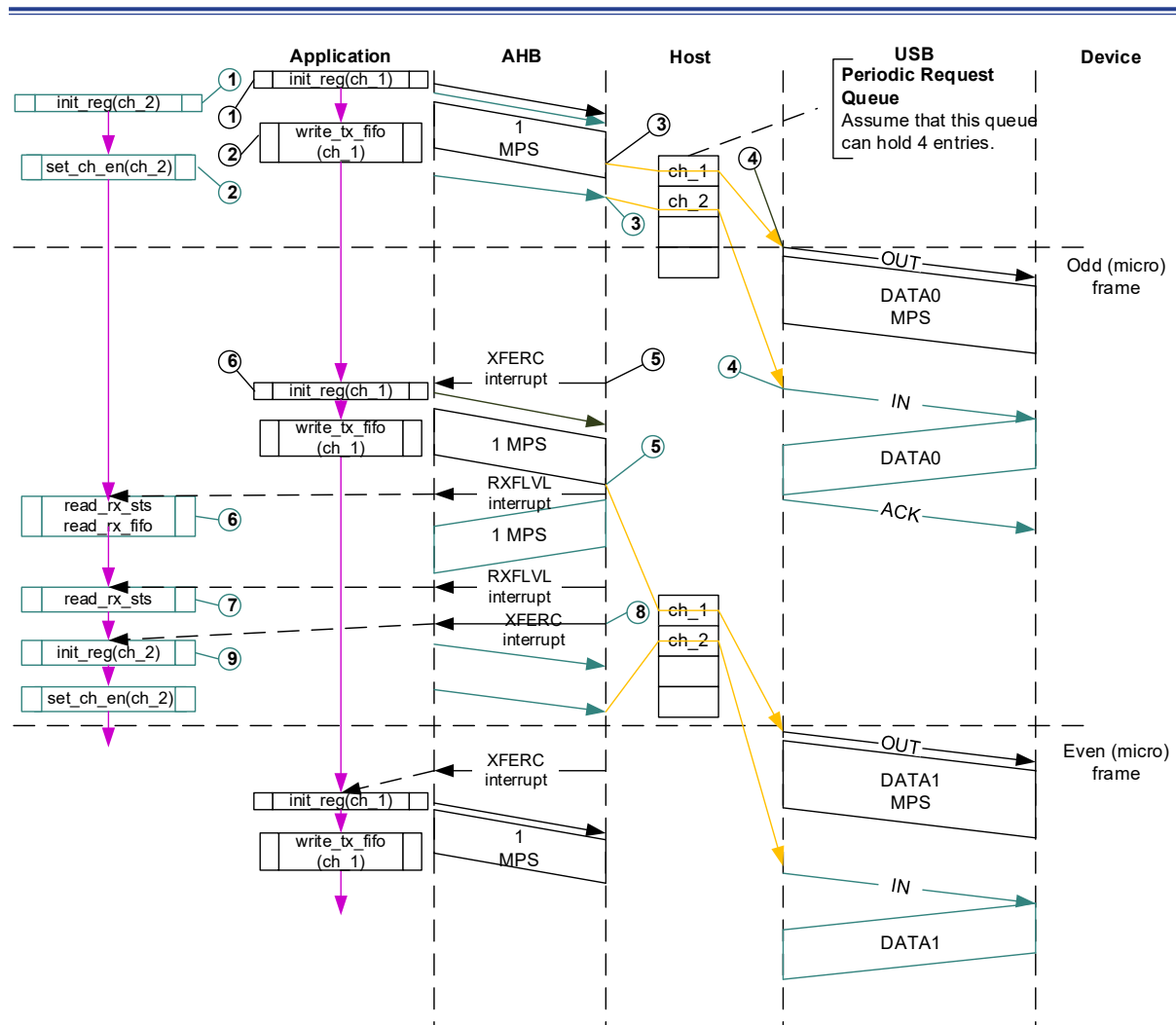
#### (1) Common interrupt IN operation process

The sequence of operations shown in Figure 247 (channel 2) is as follows:

1. Initialize channel 1 (according to OTGFS channel initialization requirements). The application must set the ODDFRM bit in the OTGFS\_HCCHAR2 register.
2. Write the first packet to the channel 1.
3. Along with the last WORD write of each packet, the host writes a request to the periodic request queue.
4. The OTGFS host sends an OUT token in the next frame (odd).
5. The host generates an XFERC interrupt after the last packet is transmitted successfully.
6. In response to the XFERC interrupt, re-initialize the channel for the next transfer.

#### (2) Handling interrupts

Figure 247 shows an example of common synchronous OUT transfers.



**Figure 247. Example of common synchronous OUT/IN transfers**

The following code describes the interrupt service routine related to the channel during synchronous OUT transfers.

```
Unmask (FRMOVRUN/XFERC)
if (XFERC)
{
    De-allocate Channel
}
else if (FRMOVRUN)
{
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    De-allocate Channel
}
```



## OTGFS device mode

### Device initialization

The application must perform the following steps to initialize the controller during power-on or after switching a mode from host to device:

1. Program the following fields in the OTGFS\_DCFG register.
  - Device speed
  - Non-zero-length status OUT handshake
  - Periodic frame interval
2. Clear the SFTDISCON bit in the OTGFS\_DCTL register. The controller will start connection after clearing this bit.
3. Program the OTGFS\_GINTMSK register to unmask the following interrupts:
  - USB reset
  - Enumeration done
  - Early suspend
  - USB suspend
  - SOF
4. Wait for the USBRESET interrupt in the OTGFS\_GINTSTS register. It indicates that a reset signal has been detected on the USB (lasting for about 10ms). Upon receiving this interrupt, the application must follow the steps defined in USB initialization on USB reset.
5. Wait for the ENUMDONE interrupt in the OTGFS\_GINTSTS register. It indicates the end of USB reset. Upon receiving this interrupt, the application must read the OTGFS\_DSTS register to determine the enumeration speed and perform the steps defined in Endpoint initialization on enumeration completion. At this time, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

### Endpoint initialization on USB reset

This section describes the operations required for the application to perform when a USB reset signal is detected:

1. Set the NAB bit for all OUT endpoints.
  - OTGFS\_DOEPCTLx.SNAK = 0x1 (for all OUT endpoints)
2. Unmask the following interrupt bits.
  - OTGFS\_DAINMSK.INEP0 = 0x1 (control IN endpoint 0)
  - OTGFS\_DAINMSK.OUTEP0 = 0x1 (control OUT endpoint 0)
  - OTGFS\_DOEPMSK.SETUP = 0x1
  - OTGFS\_DOEPMSK.XFERC = 0x1
  - OTGFS\_DIEPMSK.XFERC = 0x1
  - OTGFS\_DIEPMSK.TIMEOUT = 0x1
3. To receive/transmit data, the device must perform Device initialization steps to initialize registers.

4. Allocate SRAM for each endpoint.

- Program the OTGFS\_GRXFSIZ register to be able to receive control OUT data and SETUP data. If the allocated SRAM is equal to at least 1 largest-packet-size of control endpoint 0 + 2 WORDs (for the status of the control OUT data packet) +10 WORDs (for setup packets)
- Program the OTGFS\_DIEPTXF0 register to be able to transmit control IN data. The allocated SRAM is equal to at least 1 largest-packet-size of control endpoint 0

5. Reset the device address in the device configuration register.

6. Program the following fields in the endpoint-specific registers to ensure that control OUT endpoint 0 is able to receive a SETUP packet.

- OTGFS\_DOEPTSIZE0.SUPCNT = 0x3(to receive up to 3 consecutive SETUP packets)

At this point, all initialization required to receive SETUP packets is done.

### Endpoint initialization on enumeration completion

This section describes the operations required for the application to perform when an enumeration completion interrupt signal is detected:

- Upon detecting the enumeration completion interrupt signal, read the OTGFS\_DSTS register to get the enumeration speed.
- Program the MPS bit in the OTGFS\_DIEPCTL0 register to set the maximum packet size. This operation is used to configure control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
- Unmask SOF interrupts.

At this point, the device is ready to receive SOF packets and has been configured to perform control transfers on control endpoint 0.

### Endpoint initialization on SetAddress command

This section describes the operations required for the application to perform when the application receives a SetAddress command in a SETUP packet

- Program the OTGFS\_DCFG register with the device address received in the SetAddress command.
- Program the controller to send an IN packet.

### Endpoint initialization on SetConfiguration/SetInterface command

This section describes the operations required for the application to perform when the application receives a SetConfiguration / SetInterface command in a SETUP packet

- When a SetConfiguration command is received, the application must program the endpoint registers according to the characteristics of the valid endpoints defined in the new configuration.
- When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
- Some endpoints that were valid in the previous configuration are not valid in the new configuration. These invalid endpoints must be disabled.
- Refer to Endpoint activation and USB endpoint deactivation for more information on how to activate or disable a certain endpoint.

- Unmask the interrupt for each valid endpoint and mask the interrupts for all invalid endpoints in the DAINMSK register.
- Refer to OTGFS FIFO configuration for more information on how to program SRAM for each FIFO.
- After all required endpoints are configured, the application must program the controller to send a status IN packet.

At this point, the device controller has been ready to receive and transmit any type of data packet.

### Endpoint activation

This section describes how to activate a device endpoint or configure an existing device endpoint to a new type.

1. Program the following bits in the OTGFS\_DIEPCTLx register (for IN or bidirectional endpoints) or the OTGFS\_DOEPCTLx register (for OUT or bidirectional endpoints).
  - Largest packet size
  - USB valid endpoint = 0x1
  - Endpoint start data toggle (for interrupt and bulk endpoints)
  - Endpoint type
  - Transmit FIFO number
2. Once the endpoint is activated, the controller starts decoding the tokens issued to this endpoint and sends out a valid handshake for each valid token received for the endpoint.

### USB endpoint deactivation

This section describes how to deactivate an existing endpoint. Disable the suspended transfer before performing endpoint deactivation.

- Clear the USB valid endpoint bit in the OTGFS\_DIEPCTLx register (for IN or bidirectional endpoints) or the OTGFS\_DOEPCTLx register (for OUT or bidirectional endpoints)
- Once the endpoint is deactivated, the controller will ignore the tokens issued to this endpoint, which causes a USB timeout.

### Control write transfers (SETUP/Data OUT/Status IN)

This section describes the steps required for control write transfers.

The application programming process is as follows:

1. When the SETUP bit is set in the OTGFS\_DOEPINTx register, it indicates that a valid SETUP packet has been sent to the application, and data stage is initiated, see OUT data transfers. At the end of the SETUP stage, the application must rewrite 3 to the SUPCNT bit in the OTGFS\_DOEPSIZx register to receive the subsequent SETUP packet.
2. If the last SETUP packet received before the generation of the SETUP interrupt indicates data OUT stage, program the controller to perform OUT transfers based on Asynchronous OUT data transfer operation
3. The application can receive up to 64-byte data for a single OUT data transfer of control endpoint 0. If the application expects to receive more than 64-byte data during data OUT stage, it must re-enable the endpoint to receive another 64-byte data, and it must continue this operation until the completion of all data reception in data stage

4. When the XFERC interrupt is set in the OTGFS\_DOEPINTx register during the last OUT transfer, it indicates the end of data OUT stage of control transfer
5. Once the completion of data OUT stage, the application must perform the following steps:
  - If the application needs to transfer a new SETUP packet, it must re-enable control OUT endpoints (refer to OUT data transfers)  
OTGFS\_DOEPCTLx.EPENA = 0x1
  - To execute the received SETUP commands, the application must configure the corresponding registers in the controller. This is optional, depending on the received SETUP command type
6. During status IN stage, the application must follow the requirements of Non-periodic (for bulk and control) IN data transfers to program registers to perform data IN transfers
7. When the XFERC interrupt is set in the OTGFS\_DOEPINTx register is set, it indicates that the status stage of control transfers is started. As soon as Data transfer complete mode and Status stage start bit are set in the receive FIFO packet status register, the controller generates an interrupt. The Transfer complete interrupt can be cleared through the XFERC bit in the OTGFS\_DOEPINTx register

Repeat above-mentioned steps until an interrupt (XFERC bit in the OTGFS\_DIEPINTx register) is generated on the endpoint, which indicates the end of control write transfers.

### Control read transfers (SETUP/Data IN/Status OUT)

This section describes the steps required for control read transfers.

The application programming process is as follows:

- When the SETUP bit is set in the OTGFS\_DOEPINTx register, it indicates that a valid SETUP packet has been sent to the application, and data stage is initiated, see OUT data transfers. At the end of the SETUP stage, the application must rewrite 3 to the SUPCNT bit in the OTGFS\_DOEPSIZx register to receive the subsequent SETUP packet.
- If the last SETUP packet received before the generation of the SETUP interrupt indicates data IN stage, program the controller to perform IN transfers based on Non-periodic IN data transfer operation.
- The application can receive up to 64-byte data for a single IN data transfer of control endpoint 0. If the application expects to receive more than 64-byte data during data IN stage, it must re-enable the endpoint to receive another 64-byte data, and it must continue this operation until the completion of all data transfers in data stage.
- Repeat above-mentioned steps until the XFERC interrupt is generated in the OTGFS\_DIEPINTx register for each IN transfer on the endpoint.
- When the XFERC interrupt is set in the OTGFS\_DOEPINTx register during the last IN transfer, it indicates the end of data OUT stage of control transfer.
- To execute data OUT transfer at status OUT stage, the application must configure the controller. This is optional.

The application must program the NZSTSOUTHSHK bit in the OTGFS\_DCFG register, and then send data OUT transfer at status stage.

The XFERC interrupt bit is set in the OTGFS\_DOEPINTx register to indicate the end of status OUT stage of control transfer, marking the completion of control read transfers.

## Control transfers (SETUP/Status IN)

This section describes the two-phase control transfer operation..

The application programming process is as follows:

1. When the SETUP bit is set in the OTGFS\_DOEPINTx register, it indicates that a valid SETUP packet has been sent to the application, and data stage is initiated, see OUT data transfers. At the end of the SETUP stage, the application must rewrite 3 to the SUPCNT bit in the OTGFS\_DOEPSIZx register to receive the subsequent SETUP packet.
2. The application decodes the last SETUP packet received before the generation of the SETUP interrupt. If the SETUP packet indicates two-level control commands, the application must perform the following steps:
  - Set OTGFS\_DOEPCTLx.EPENA = 0x1
  - The application must program the registers in the controller to perform the received SETUP commands.
3. For status IN stage, the application must program the registers based on Non-periodic (bulk and control) IN data transfers to perform data IN transfers.
4. The XFERC interrupt bit is set in the OTGFS\_DIEPINTx register to indicate the end of status IN stage of control transfers.

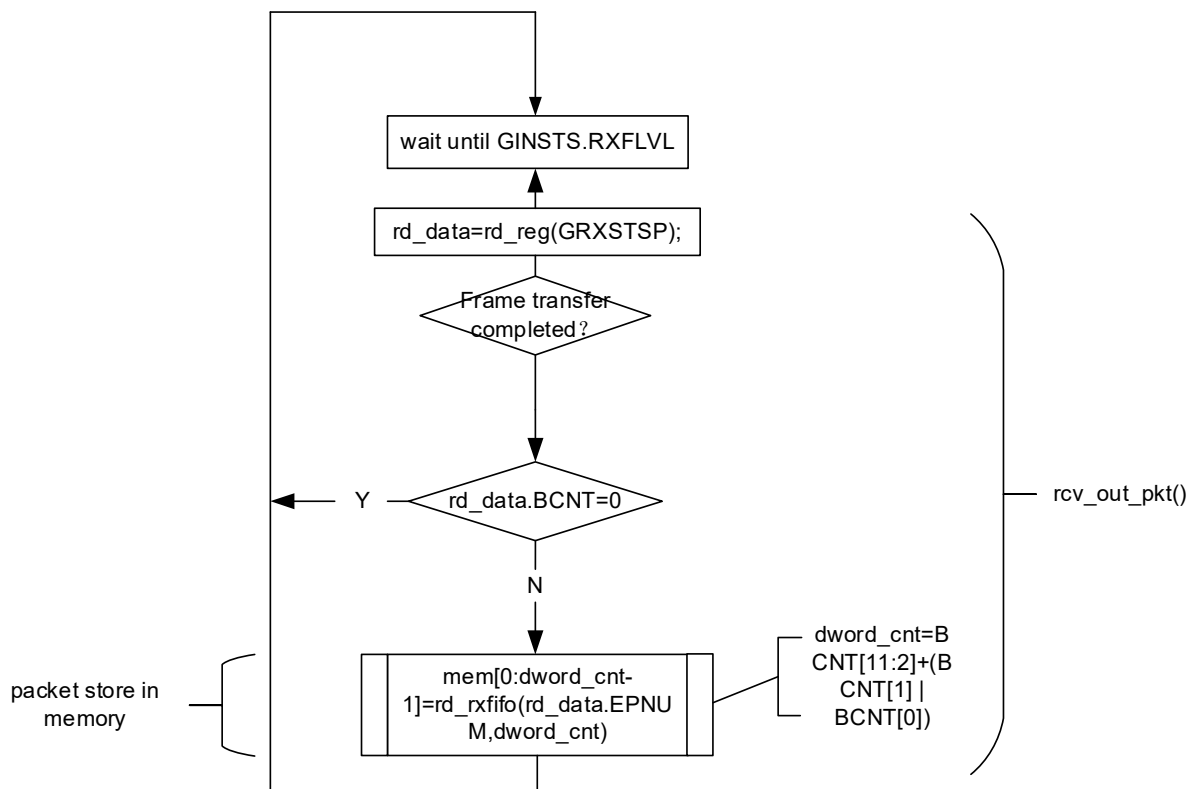
## Read FIFO packets

This section describes how to read FIFO packets (OUT data and SETUP packets)

1. The application must read the OTGFS\_GRXSTSP register as soon as the RXFLVL interrupt bit is detected in the OTGFS\_GINTSTS register.
2. The application can mask the RXFLVL interrupt bit in the OTGFS\_GINTSTS register by setting RXFLVL = 0x0 in the OTGFS\_GINTMSK register, until it has read the data packets from the receive FIFO.
3. If the received packet byte is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is read from the receive data FIFO.
4. The receive FIFO packet status reading indicates one of the following conditions:
5. Global OUT NAK mode: PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = Don't Care (0x0) and DPID = Don't Care (0x00), indicating that the global OUT NAK bit has taken effect.
  - SETUP packet mode: PKTSTS = SETUP, BCnt = 0x008, EPNUM = Control EP Num and DPID = D0, indicating that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
  - Setup stage done mode: PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num and DPID = Don't Care (0x00), indicating the completion of the Setup stage for the specified endpoint, and the start of the data stage. After this request is popped from the receive FIFO, the controller triggers a Setup interrupt on the specified control OUT endpoint.
  - Data OUT packet mode: PKTSTS = DataOUT, BCnt = size of the received data OUT packet (0 ≤ BCNT ≤ 1024), EPNUM = Endpoint number on which the data packet was received, DPID = Actual data PID.
  - Data transfer complete mode: PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT endpoint number on which the data transfer is complete, DPID = Don't Care (0x00). These data indicate that an OUT data transfer for the specified OUT endpoint has been

complete. After this request is popped from the receive FIFO, the controller triggers a Transfer Completed interrupt on the specified OUT endpoint. PKTSTS code can be found in the OTGFS\_GRXSTSR / OTGFS\_GRXSTSP register.

6. After the valid data is popped from the receive FIFO, the RXFLVL interrupt bit in the OTGFS\_GINTSTS register must be unmasked.
7. Step 1-5 must be repeated each time the application detects the interrupt line due to the RXFLVL bit in the OTGFS\_GINTSTS register. Reading an empty receive FIFO will result in unexpected behavior. Figure 248 shows a flowchart.



**Figure 248. Read receive FIFO**

## OUT data transfers

This section describes the internal data flow during data OUT and SETUP transfers, and how the application handles SETUP transfers.

### (1) Setup transfers

This section describes how to handle SETUP data packets and the application's operating sequence of handling SETUP transfers. After power-on reset, the application must follow the OTGFS Initialization process to initialize the controller. Before communicating with the host, the application must initialize the endpoints based on Device Initialization, and refer to Read FIFO packets for more information.

### 【Application requirements】

1. To receive a SETUP packet, the SUPCNT bit (OTGFS\_DOEPTSIZE<sub>x</sub>) on a control OUT endpoint must be programmed to be a non-zero value. When the application sets the SUPCNT bit to a non-zero value, the controller receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status bit and EPENA bit in the OTGFS\_DOEPTCTL<sub>x</sub> register. The SUPCNT bit is decremented each time the control endpoint receives a SETUP packet. If the SUPCNT bit is not programmed to a proper value before receiving a SETUP packet, the controller still receives the SETUP packet and decrements the SUPCNT bit, but the application may not be able to determine the exact number of SETUP packets received in the SETUP stage of a control transfer.
  - OTGFS\_DOEPTSIZE<sub>x</sub>.SUPCNT = 0x3
2. The application must allocate some extra space for the receive data FIFO to ensure that up to three SETUP packets can be received on a control endpoint.
  - The space to be reserved is 13 WORDs. Four WORDs are required for one SETUP packet, one WORD is required for the Setup stage and 8 WORDs are required to store two extra SETUP packets among all control endpoints.
  - Four WORDs per SETUP packet are required to store 8-byte SETUP data and 4-byte Transfer completed status and 4-byte SETUP status (SETUP packet mode). The controller must reserve this space to receive data.
  - FIFO is used to write SETUP data only, and never for data packets.
3. The application must read 2-WORDs SETUP packet from the receive data.
4. The application must read and discard the Transfer Completed status WORD from the receive FIFO, and ignore the Transfer Completed interrupt due to this read operation.

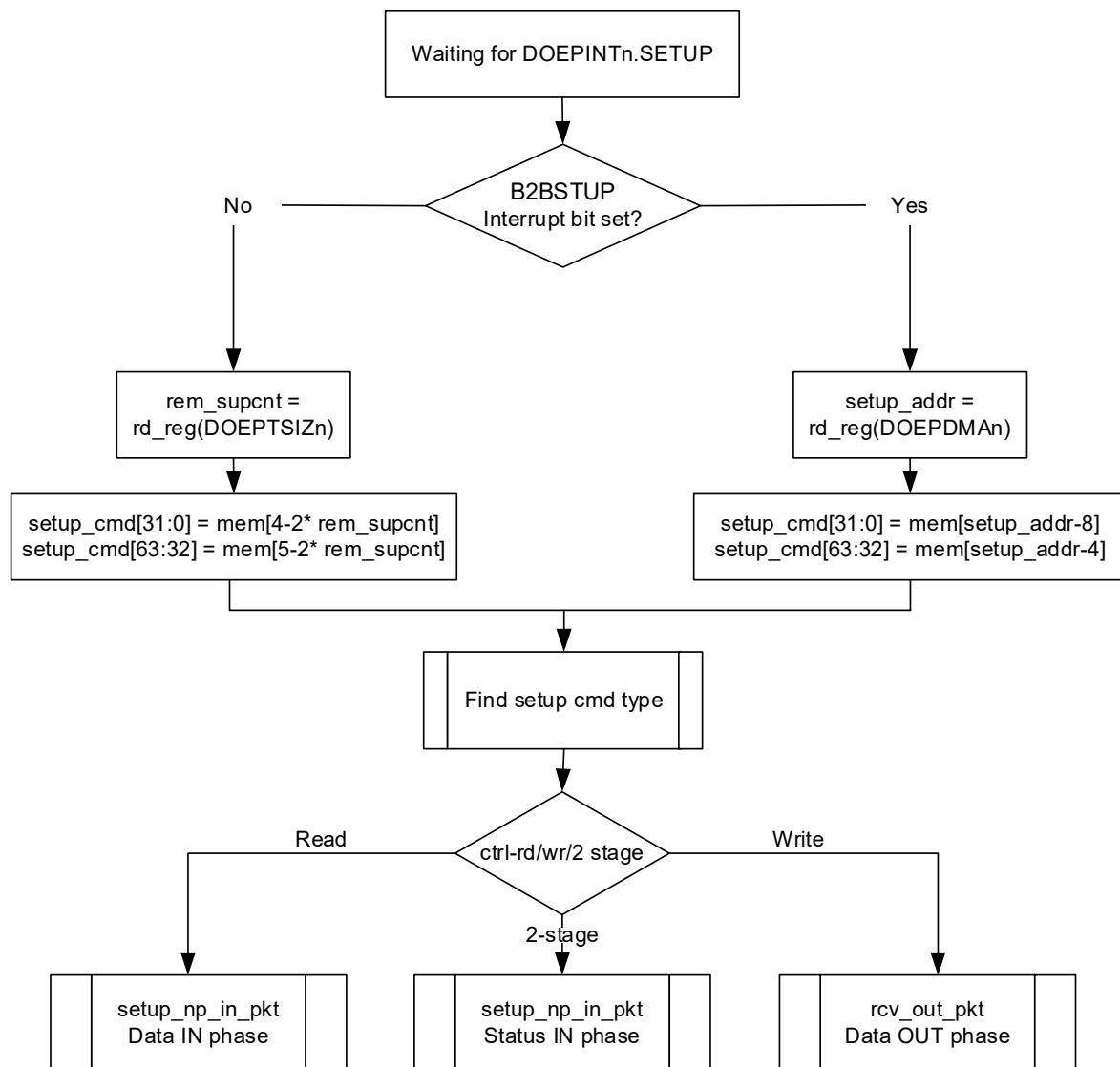
### 【Internal data flow】

1. When a SETUP packet is received, the controller writes the received data to the receive FIFO, without checking whether there is available space in the receive FIFO, irrespective of the NAK and Stall bits on the control endpoints.
  - The controller sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB line, 3 WORDs of data are written to the receive FIFO, and the SUPCNT bit is decremented by 1 automatically.
  - The first WORD contains control information used internally by the controller
  - The second WORD contains the first 4 bytes of the SETUP command
  - The third WORD contains the last 4 bytes of the SETUP command
3. When the SETUP stage switches to data IN/OUT stage, the controller writes a SETUP status done WORD to the receive FIFO, indicating the end of the SETUP stage.
4. The application reads the SETUP packages through the AHB bus.
5. When the application pops the Setup stage done WORD from the receive FIFO, the controller interrupts the application through the SETUP interrupt bit in the OTGFS\_DOEPTINT<sub>x</sub> register, indicating that the application can start processing the SETUP packet received.
6. The controller clears the endpoint enable bit for control OUT endpoints.



### 【Application programming process】

1. Program the OTGFS\_DOEPTSIZx register.
  - OTGFS\_DOEPTSIZx.SUPCNT = 0x3
2. Wait for the RXFLVL interrupt bit in the OTGFS\_GINTSTS register and read and empty the data packets from the receive FIFO (Refer to Read FIFO packets for details). This operation can be repeated several times.
3. When the SETUP interrupt bit is set in the OTGFS\_DOEPINTx register, it indicates that the SETUP data transfer has been completed successfully. Upon this interrupt, the application must read the OTGFS\_DOEPTSIZx register to determine the number of SETUP packets received, and process the last received SETUP packet.



**Figure 249. SETUP data packet flowchart**



## (2) Handling more than three consecutive SETUP packets

Per the USB 2.0 specification, typically, a host does not send more than three consecutive SETUP packets to the same endpoint during a SETUP packet error. However, the USB2.0 specification does not limit the number of consecutive SETUP packets a host can send to the same endpoint. If this condition occurs, the OTGFS controller generates an interrupt (B2BSTUP bit in the OTGFS\_DOEPINTx register).

## IN data transfers

This section describes the internal data flow during IN data transfers and how the application handles IN data transfers.

- The application can either select a polling or an interrupt mode.
  - In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTGFS\_DTXFSTSx register to determine whether there is enough space in the data FIFO.
  - In interrupt mode, the application must wait for the TXFEMP interrupt bit in the OTGFS\_DIEPINTx register, and then read the OTGFS\_DTXFSTSx register to determine whether there is enough space in the data FIFO.
  - To write a single non-zero-length data packet, there must be enough space to write the entire data packet in the data FIFO.
  - To write zero-length data packet, the application does not need to check the FIFO space.
- Either way, when the application determines that there is enough space to write a transmit packet, the application can first write into the endpoint control register before writing the data into the data FIFO. Normally, except for setting the endpoint enable bit, the application must do a read modify write on the OTGFS\_DIEPCTLx register to avoid modifying the contents of the register. If the space is enough, the application can write multiple data packets for the same endpoint into the transmit FIFO. For the periodic IN endpoints, the application must write packets for only one frame. It can write packets for the next periodic transfer only after the previous transfer has been completed.

## Non-periodic (bulk and control) IN data transfers

To initialize the controller after power-on reset, the application must perform the steps list in OTGFS Initialization. Before communicating with a host, the controller must follow the steps defined in Device Initialization to initialize endpoints.

### 【Application requirements】

- For IN transfers, the Transfer Size bit in the Endpoint Transfer Size register indicates a payload that contains multiple largest-packet-size packets and a short packet. This short packet is transmitted at the end of the transfer.
  - To transmit several largest-packet-size packets and a short packet:  

$$\text{Transfer size [epnum]} = n \times \text{mps[epnum]} + \text{sp} \quad (n \text{ is an integer } \geq 0 \text{ and } 0 \leq \text{sp} < \text{mps[epnum]})$$
 If (sp > 0), then packet count [epnum] = n + 1. Otherwise, packet count [epnum] = n
  - To transmit a single zero-length data packet:  

$$\text{Transfer size [epnum]} = 0x0$$

$$\text{Packet count [epnum]} = 0x1$$

- To transmit several largest-packet-size packets and a zero-length data packet (at the end of the transfer), the application must split the transfer into two parts. First send the largest-packet-size packets and then the zero-length data packet alone.

First transfer: Transfer size [epnum] =  $n \times \text{mps}[\text{epnum}]$ ; Packet count =  $n$ ;

Second transfer: Transfer size [epnum] =  $0x0$ ; Packet count =  $0x1$ ;

- If an endpoint is enabled for data transfers, the controller updates the Transfer size register. At the end of the IN transfer (indicated by endpoint disable interrupt bit), the application must read the Transfer size register to determine how much data in the transmit FIFO have already been sent on the USB line.
- Data fetched in the transmit FIFO = Application-programmed initial transfer size – Controller-updated final transfer size
  - Data transmitted on USB = (Application-programmed initial packet count – Controller-updated final packet count)  $\times$  mps[epnum]
  - Data to be transmitted on USB = Application-programmed initial transfer size – Data transmitted on USB

#### 【Internal data flow】

- The application must set the transfer size and packet count bits in the endpoint control registers and enable the endpoint to transmit the data.
- The application must also write the required data to the transmit FIFO of the endpoint.
- Each time a data packet is sent to the transmit FIFO by the application the transfer size for this endpoint is decremented with the packet size. The application must continue to write data until the transfer size of the endpoint becomes 0. After writing data to the FIFO, the “packet count in the FIFO” is incremented (this is a 3-bit count for each IN endpoint transmit FIFO data packet, which is internally maintained by the controller. For an IN endpoint FIFO, the maximum number of packets maintained by the controller at any time is 8). For non-zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
- After the data is written to the transmit FIFO, the controller reads them upon receiving an IN token. For each non-synchronous IN data packet transmitted with an ACK handshake signal, the number of packets for the endpoint is decremented by 1, until the packet count becomes 0. The packet count is not decremented due to a timeout.
- For zero-length data packets (indicated by an internal zero-length flag), the controller sends zero-length packets according to the IN token, and the packet count is decremented automatically.
- If there are no data in the FIFO on a received IN token and the packet count for the endpoint is 0, the controller generates an “IN token received when FIFO is empty” interrupt, and the NAK bit for the endpoint is not set. The controller responds with a NAK handshake signal to the non-synchronous endpoints on the USB.
- The controller rewinds the FIFO pointers internally and no timeout interrupt is generated except for the control IN endpoints.
- When the transfer size is 0 and the packet count is also 0, the Transfer completed interrupt is generated and the endpoint enable bit is cleared.

#### 【Application programming sequence】

- Program the OTGFS\_DIEPTSIZE register according to the transfer size and the corresponding packet count.

2. Program the OTGFS\_DIEPCTLx register according to the endpoint characteristics and set the CNAK and endpoint enable bits.
3. While sending non-zero-length data packets, the application must poll the OTGFS\_DTXFSTSx register (where n is the FIFO number related to that endpoint) to determine whether there is enough space in the data FIFO. The application can also use the TXFEMP bit in the OTGFS\_DIEPINTx register before writing data.

### Non-synchronous OUT data transfers

To initialize the controller after power-on reset, the application must perform the steps list in “OTGFS Initialization”. Before communicating with a host, the application must initialize endpoints based on the process described in “Endpoint Initialization” and by referring to “Read FIFO packets”. This section describes a regular non-synchronous OUT transfers (control, bulk or interrupt transfers).

#### 【Application requirements】

1. For OUT data transfers, the transfer size of the endpoint transfer register must be set to a multiple of the largest packet size for the endpoint, and adjusted to the WORD boundary.

```
if (mps[epnum] mod 4) == 0
    transfer size[epnum] = n × (mps[epnum]) //WORD Aligned
else
    transfer size[epnum] = n × (mps[epnum] + 4 - (mps[epnum] mod 4)) //Non WORD
    Aligned
packet count[epnum] = n
n > 0
```

2. When an OUT endpoint interrupt occurs, the application must read the endpoint’s transfer size register to calculate the size of the data in the memory. The received payload size must be less than the programmed transfer size.

- Payload size in memory = Application-programmed initial transfer size – Controller-updated final transfer size
- Number of USB packets the payload was received = Application-programmed initial packet count – Controller-updated final packet count

#### 【Internal data flow】

1. The application must set the transfer size and packet count bits in the endpoint control registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the controller starts receiving data and writes it to the receive FIFO as long as there is available space in the receive FIFO. For each data packet received on the USB line, the data packet and its status are written to the receive FIFO. The packet count is decremented by 1 each time a packet (largest packet size or a short packet) is written to the receive FIFO.

- OUT data packets received with Bad Data CRC are emptied from the receive FIFO
- After sending an ACK to the data packet on the USB, the controller discards non-synchronous OUT data packets that the host (which cannot detect the ACK) re-transmits. The application does not detect multiple consecutive OUT data packets on the same endpoint with the same data PID. In this case, the packet count is not decremented.
- If there is no space in the receive FIFO, synchronous or non-synchronous data packets are ignored and not written to the receive FIFO. Besides, the non-synchronous OUT tokens receive a NAK handshake response.
- In all the above-mentioned cases, the packet count is not decremented because no data is written to the receive FIFO.

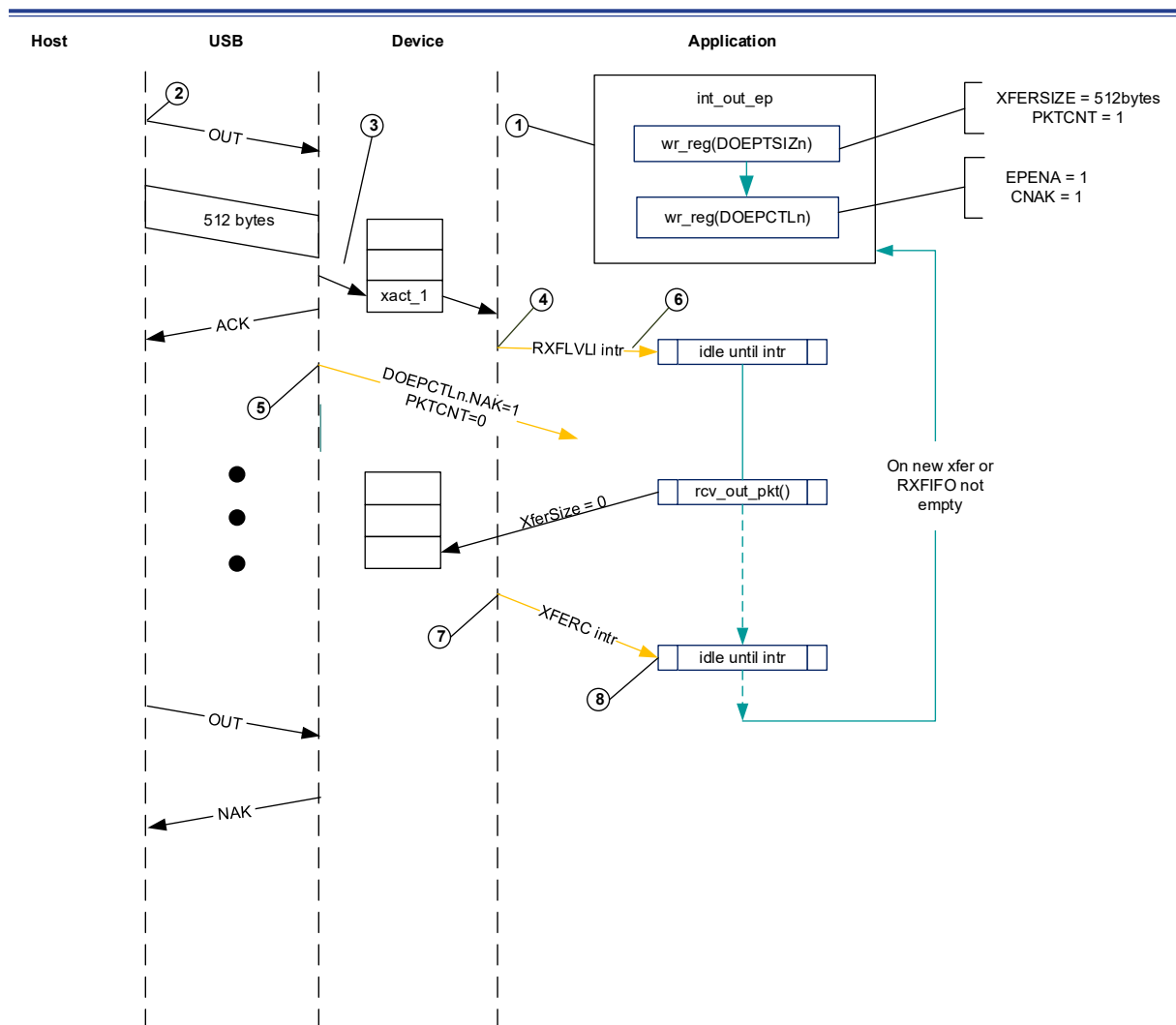
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for the endpoint is set. Once the NAK bit is set, the synchronous or non-synchronous data packets are ignored and not written to the receive FIFO, and non-synchronous OUT tokens receive a NAK handshake response.
4. After the data is written to the receive FIFO, the application reads the data from the receive FIFO and writes it to the external memory, once packet at a time per endpoint.
5. At the end of data packet write to the external memory, the transfer size for the endpoint is decremented with the size of the written packet.
6. The OUT data transfer completed mode for an OUT endpoint is written to the receive FIFO in one of the following conditions:
  - The transfer size and packet count are both 0
  - The last OUT data packet written to the receive FIFO is a short packet ( $0 \leq \text{data packet size} < \text{largest packet size}$ )
7. When the application pops this entry (OUT data transfer complete), a transfer completed interrupt is generated and the endpoint enable bit is cleared.

#### **【Application programming sequence】**

1. Program the OTGFS\_DOEPTSTZx register with the transfer size and the corresponding packet count.
2. Program the OTGFS\_DOEPCTLx register with the endpoint characteristics, and set the endpoint enable and ClearNAK bits.
  - OTGFS\_DOEPCTLx.EPENA = 0x1
  - OTGFS\_DOEPCTLx.CNAK = 0x1
3. Wait for the RXFLVL interrupt in the OTGFS\_GINTSTS register, and read out all data packets from the receive FIFO.
  - This step can be repeated, depending on the transfer size
4. When the XFERC interrupt is set in the OTGFS\_DOEPINTx register, it indicates a successful completion of the non-synchronous OUT data transfer. Read the OTGFS\_DOEPTSTZx register to determine how much data has been received.

#### **【Bulk OUT transfer】**

Figure 250 describes the reception of a single bulk OUT data packet from the USB to the AHB and shows the events involved in the process.



**Figure 250. BULK OUT transfer block diagram**

After a SetConfiguration/SetInterface command is received, the application initializes all OUT endpoints by setting CNAK = 0x1 and EPENA = 0x1 in the OYG\_DOEPCTLx register, and setting the XFERSIZE and PKTCNT bits in the OTGFS\_DOEPSIZx register.

1. The host attempts to send data (OUT token) to the endpoint
2. When the controller receives the OUT token on the USB, it stores data in the receive FIFO because the FIFO has free space.
3. Upon writing the complete data in the receive FIFO, the controller then triggers the RXFLVLJ interrupt bit in the OTGFS\_GINTSTS register.
4. Upon receiving the packet count of USB packets, the controller internally sets the NAK bit for the endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the receive FIFO.
6. When the application reads all the data (equivalent to XFERSIZE), the controller generates an XFERC interrupt in the OTGFS\_DOEPINTx register.

7. The application processes the interrupt and uses the XFERC bit in the OTGFS\_DOEPINTx register to judge that the expected transfer is already complete.

### Synchronous OUT data transfers

To initialize the controller after power-on reset, the application must perform the steps list in “OTGFS Initialization”. Before communicating with a host, the application must initialize endpoints based on the process described in “Endpoint Initialization” and by referring to “Read FIFO packets”. This section describes a regular synchronous OUT transfers.

#### 【Application requirements】

1. All the application requirements are the same as that of non-synchronous OUT data transfers.
2. For synchronous OUT data transfers, the transfer size and packet count must be set to the number of the largest-packet-size packets that can be received in a single frame and not exceed this size. Synchronous OUT data transfer cannot span more than one frame.
  - $1 \leq \text{packet count [epnum]} \leq 3$
3. If the device supports the synchronous OUT endpoints, the application must read all synchronous OUT data packets from the receive FIFO before the end of the periodic frame (EOPF interrupt in the OTGFS\_GINTSTS register).
4. To receive data in the subsequent frame, a synchronous OUT endpoint must be enabled before the generation of the EOPF and SOF interrupt in the OTGFS\_GINTSTS register.

#### 【Internal data flow】

1. The internal data flow for the synchronous OUT endpoints is the same as that for the non-synchronous OUT endpoints, just for a few differences.
2. When the synchronous OUT endpoint is enabled by setting the endpoint enable bit and by clearing the NAK bit, the even/odd frame bits are also set properly. The controller can receive data on an synchronous OUT endpoint in a particular frame only when the following condition is met:
  - Even/Odd microframe (OTGFS\_DOEPCTLx) = SOFFN[0] (OTGFS\_DSTS)
3. When the application completely reads the synchronous OUT data packet (data and status) from the receive FIFO, the controller updates the RXDPID bit in the OTGFS\_DOEPSIZx register based on the data PID of the last synchronous OUT data packet read from the receive FIFO.

#### 【Application programming sequence】

1. Program the transfer size and the corresponding packet count of the OTGFS\_DOEPSIZx register
2. Program the OTGFS\_DOEPCTLx register with the endpoint enable, ClearNAK and Even/Odd frame bits
  - Endpoint enable = 0x1
  - CNAK = 0x1
  - Even/Odd frame = (0x0: Even; 0x1: Odd)
3. Wait for the RXFLVL interrupt in the OTGFS\_GINTSTS register, and read all the data packets from the receive FIFO. See “Read FIFO” for more information
  - This step can be repeated several times, depending on the transfer size
4. When the XFERC interrupt is set in the OTGFS\_DOEPINTx register, it indicates the completion of the synchronous OUT data transfers. But this interrupt does not necessarily mean that the data in memory are good.

5. This interrupt signal cannot always be detected by the synchronous OUT data transfers. However, the application can detect the INCOMPSOOUT interrupt in the OTGFS\_GINTSTS register. See “Incomplete synchronous OUT data transfers” for more information.
6. Read the OTGFS\_DOEPTISIZx register to determine the received transfer size and to determine whether the data received in the frame are valid or not. The application must treat the data received in memory as valid only when one of the following conditions is met:
  - OTGFS\_DOEPTISIZx.RxDPID = 0xD0 and the USB packet count in which the payload was received = 0x1
  - OTGFS\_DOEPTISIZx.RxDPID = 0xD1 and the USB packet count in which the payload was received = 0x2
  - OTGFS\_DOEPTISIZx.RxDPID = 0xD2 and the USB packet count in which the payload was received = 0x3

The number of USB packets in which the payload was received = Application-programed initial packet count – Controller-updated final packet count

The application discards invalid data packets.

### Enable synchronous endpoints

After sending a Set interface control command to the device, a host enables the synchronous endpoints. Then the host can send the initial synchronous IN token in any frame before transmission in the sequence of BInterval.

Instead, synchronous support in the OTGFS controller is based on a single-transfer level. The application must re-configure the controller on every frame. The OTGFS controller enables the synchronous endpoint of the frame before the frame to be transmitted.

For example, to send data on the frame n, enable the endpoint of the frame n-1. Additionally, the OTGFS controller schedules the synchronous transfers by setting Even/Odd frame bits.

#### 【Synchronous IN transfer interrupt】

The following interrupts must be processed to ensure successful scheduling of the synchronous transfers.

- XFERC interrupt in the OTGFS\_DIEPINTx register (for endpoints)
- OTG INCOMPSOIN interrupt in the OTGFS\_GINTSTS register (for global interrupts)

#### 【Handling synchronous IN transfers】

The following steps must be performed to handle a synchronous IN transfer:

1. Unmask the incompISOOUT interrupt in the OTGFS\_GINTSTS register by setting the INCOMISOINMSK interrupt bit in the OTGFS\_GINTMSK register
2. Unmask the XFERC interrupt in the OTGFS\_DIEPINTx register by setting the XFERCMSK bit in the OTGFS\_DIEPMSK register
3. Enable synchronous endpoints with the following steps:
  - Program the OTGFS\_DIEPTISIZx register
$$\text{OTGFS\_DIEPTISIZx.XFERSIZE} = n \times \text{OTGFS\_DIEPCTLx.MPS} + \text{sp}, \text{ where } 0 \leq n \leq 3 \text{ and } 0 \leq \text{sp} < \text{OTGFS\_DIEPCTLx.MPS}.$$

When the transfer size in a frame is less than that of the MPS bit in the OTGFS\_DIEPCTLx register,  $n = 0$ ; When the transfer size in a frame is a multiple of that of the MPS bit in the OTGFS\_DIEPCTLx register,  $\text{sp} = 0$ .



OTGFS\_DIEPTSIZE.PKTCNT = 0x1

The MC bit in the OTGFS\_DIEPTSIZE register is set the same value as that of the PKTCNT bit in the OTGFS\_DIEPTSIZE register.

- Program the OTGFS\_DIEPCTLx register

Read the OTGFS\_DSTS register to determine the current frame number

Program the OTGFS\_DIEPCTLx with the maximum packet size (MPS bit)

Set USBACTEP = 0x1 in the OTGFS\_DIEPCTLx register

Set EPTYPE = 0x1 in the OTGFS\_DIEPCTLx register, marking synchronization

Set the FIFO number of the endpoint through the TXFNUM bit in the OTGFS\_DIEPCTLx register

Set CNAK = 0x1 in the OTGFS\_DIEPCTLx register

If SOFFN[0] = 0x0 in OTGFS\_DSTS, then SETEVENFR = 0x1 in OTGFS\_DIEPCTLx (otherwise, SETEVENFR = 0x1 in OTGFS\_DIEPCTLx)

If SOFFN[0] = 0x1 in OTGFS\_DSTS, then SETODDFR = 0x1 in OTGFS\_DIEPCTLx (otherwise, SETODDFR = 0x0 in OTGFS\_DIEPCTLx)

Set EPENA = 0x1 in OTGFS\_DIEPCTLx

#### 4. Write endpoint data to the corresponding transmit FIFO

For example, write address ranges are as follows:

- EP1 corresponding to 0x2000 - 0x2FFC
- EP2 corresponding to 0x3000 - 0x3FFC
- EP3 corresponding to 0x3000 - 0x3FFC
- ...

#### 5. Wait for interrupts

- When an interrupt is generated (XFERC bit in OTGFS\_DIEPINTx register), clear the XFERC interrupt. For the following transaction, repeat step 3-5 until the completion of data transfers.
- When an interrupt is generated (INCOMPISOIN bit in OTGFS\_GINTSTS register), clear the INCOMPISOIN interrupt; For any synchronous IN endpoint, when Odd/Even bits match the current frame number bit 0, and when the endpoint remains enabled, the controller generates an interrupt at the end of the frame. This interrupt is generated on one of the following conditions:
  - (1) There is no token in a frame
  - (2) Late data write to the receive FIFO. An IN token has arrived before the completion of data write
  - (3) IN token error

The INCOMPISOIN interrupt in the OTGFS\_GINTSTS register is a global interrupt. Therefore, when more than one synchronous endpoints are in active state, the application must determine which one of the synchronous IN endpoints has not yet completed data transfers.

To achieve this, read the DSTS and DIEPCTLx bits of all synchronous endpoints. If the current endpoint has been enabled, and the read value of the SOFFN bit in the OTGFS\_DSTS register is equal to the target frame number of the endpoint, it indicates that this endpoint has not finished data transfers. The application must keep track of and update the target frame number of the synchronous endpoint.



If data transfer is not yet complete on an endpoint, then Odd/Even bits have to be toggled.

Next:

- (1) When the DPID is set to 1 (an odd frame) in the OTGFS\_DIEPCTLx register, write 1 to the SETD0PID bit in the OTGFS\_DIEPCTLx register makes it an even frame, then data transmission starts when there is an IN token input in the next frame.
- (2) When the DPID is set to 0 in the OTGFS\_DIEPCTLx register, write 1 to the SETD1PID bit in the OTGFS\_DIEPCTLx register makes it an odd frame, then data transmission starts when there is an IN token input in the next frame.

### Incomplete synchronous OUT data transfers

To initialize the controller after power-on reset, the application must perform the steps list in OTGFS Initialization. Before communicating with a host, the controller must follow the steps defined in Endpoint Initialization to initialize endpoints. This section describes the application programming sequence when the controller drops synchronous OUT data packets.

#### 【Internal data flow】

1. For synchronous OUT endpoints, the XFERC interrupt (in the OTGFS\_DOEPINTx register) may not always be generated. If the controller drops synchronous OUT data packets, the application may fail to detect the XFERC interrupt in the OTGFS\_DOEPINTx register.
  - When the receive FIFO cannot accommodate the complete ISO OUT data packet, the controller drops the received ISO OUT data.
  - When the synchronous OUT data packet is received with CRC errors.
  - When the synchronous OUT token received by the controller is corrupted.
  - When the application is very slow in reading the receive FIFO
2. When the controller detects the end of periodic frames before transfer complete to all synchronous OUT endpoints, an interrupt of incomplete synchronous OUT data is generated, indicating that an XFERC interrupt in the OTGFS\_DOEPINTx register is not set on at least one of the synchronous OUT endpoints. At this point, the endpoint with the incomplete data transfer remains enabled, but no valid transfers are in progress on this endpoint.

#### 【Application programming sequence】

1. The assertion of the incomplete synchronous OUT data interrupt indicates that at least one synchronous OUT endpoint has an incomplete data transfer in the current frame.
2. If this occurs because the synchronous OUT data is not completely read out from the endpoint, the application must empty all synchronous OUT data (data and status) in the receive FIFO before proceeding.
  - When all data are read from the receive FIFO, the application can detect the XFERC interrupt in the OTGFS\_DOEPINTx register. In this case, the application must re-enable the endpoint to receive the synchronous OUT data in the next frame by following the steps listed in “SETUP/Data IN/Status OUT”.
3. When it receives an incomplete synchronous OUT data interrupt, the application must read the control registers of all synchronous OUT endpoints to determine which one of the endpoints has an incomplete data transfer in the current frame. An endpoint transfer is regarded as incomplete if both of the following conditions are met:
  - OTGFS\_DOEPCTLx. Even/Odd frame bit = OTGFS\_DSTS.SOFFN[0]
  - OTGFS\_DOEPCTLx. Endpoint enable = 0x1

4. The pervious step must be performed before the SOF interrupt of the GINTSTS register is detected to ensure that the current frame number is not changed.
5. For synchronous OUT endpoints with incomplete transfers, the application must drop the data in memory, and disable the endpoint through the endpoint disable bit in the OTGFS\_DOEPCTLx register.
6. Wait for the endpoint disable interrupt in the OTGFS\_DOEPINTx register, and enable the endpoint to receive new data in the next frame by following the steps listed in “SETUP/Data IN/Status OUT”. Because the controller can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving wrong synchronous data.

### Incomplete synchronous IN data transfers

This section describes how the application behaves on incomplete synchronous IN transfers.

#### 【Internal data flow】

1. Synchronous IN transfers are incomplete on one of the following conditions:
  - The controller receives corrupted synchronous IN tokens from more than one synchronous IN endpoints. In this case, the application can detect the incomplete synchronous IN transfer interrupt in the GINTSTS register.
  - The application is slow in writing complete data to the transmit FIFO, and an IN token is received before the completion of data write. In this case, the application can detect the INTKNTXFEMP interrupt in the OTGFS\_DIEPINTx register. The application ignores this interrupt, which will result in the generation of the incomplete synchronous IN transfer interrupt (in OTGFS\_GINTSTS register). The controller responds to the received IN token by sending a zero-length data packet to the USB.
2. Either way, the application must stop writing the transmit FIFO as soon as possible.
3. The application must set the NAK and disable bits of the endpoints.
4. The controller disables the endpoint, clears the disable bit, and triggers the endpoint disable interrupt.

#### 【Application programming sequence】

1. When the transmit FIFO becomes empty, the application ignores the INTKNTXFEMP interrupt (in the OTGFS\_DIEPINTx register) from any synchronous IN endpoint because this can trigger the incomplete synchronous IN interrupt.
2. The incomplete synchronous IN transfer interrupt (in the OTGFS\_GINTSTS register) indicates that at least one synchronous IN endpoint is with incomplete synchronous IN transfers.
3. The application must read the endpoint control registers of all synchronous IN endpoints to determine which one is with incomplete synchronous IN transfers.
4. The application must write data to the periodic transmit FIFO of the endpoint.
5. Disable theses endpoints by setting the following bits in the OTGFS\_DIEPCTLx register
  - OTGFS\_DIEPCTLx.SETNAK = 0x1
  - OTGFS\_DIEPCTLx.endpoint enable = 0x1
6. The endpoint disable interrupt in the DIEPINTx register indicates that the controller has disabled the endpoint.
7. At this point, the application must empty the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next frame. The application must refresh the data through the OTGFS\_GRSTCTL register.

## Periodic IN (interrupt and synchronous) data transfers

This section describes a typical periodic IN data transfer.

To initialize the controller after power-on reset, the application must perform the steps list in OTGFS Initialization. Before communicating with a host, the controller must follow the steps defined in Endpoint Initialization to initialize endpoints.

### 【Application requirements】

- Application requirements in “Non-periodic (bulk and control) IN data transfers” also apply to periodic IN data transfers, except for a slight difference of requirement 2.
  - The application can only transmit multiples of largest-packet-size data packets, and a short packet. To transmit several largest-packet-size data packets and a short packet, the following conditions must be met:  
 Transfer size [epnum] =  $n \times \text{mps}[\text{epnum}] + \text{sp}$  (where  $n$  and  $i$  are integers  $\geq 0$ , and  $0 \leq \text{sp} < \text{mps}[\text{epnum}]$ )  
 If ( $\text{sp} > 0$ ), packet count [epnum] =  $n + 1$ . Otherwise, packet count [epnum] =  $n$ ,  $\text{mc}[\text{epnum}] = \text{packet count}[\text{epnum}]$
  - The application cannot transmit a zero-length data packet at the end of a transfer. But it can transmit a single zero-length data packet in itself, provided packet count [epnum] = 1,  $\text{mc}[\text{epnum}] = \text{packet count}[\text{epnum}]$
- The application can only schedule data transfers of one frame at a time
  - $(\text{OTGFS\_DIEPTSIZE}x.\text{MC} - 1) \times \text{OTGFS\_DIEPCTL}x.\text{MPS} \leq \text{OTGFS\_DIEPTSIZE}x.\text{XFERSIZ} \leq \text{OTGFS\_DIEPTSIZE}x.\text{MC} \times \text{OTGFS\_DIEPCTL}x.\text{MPS}$
  - $\text{OTGFS\_DIEPTSIZE}x.\text{PKTCNT} = \text{OTGFS\_DIEPTSIZE}x.\text{MC}$
  - If  $\text{OTGFS\_DIEPTSIZE}x.\text{XFERSIZ} < \text{OTGFS\_DIEPTSIZE}x.\text{MC} \times \text{OTGFS\_DIEPCTL}x.\text{MPS}$ , the last data packet of the transfer is a short packet.
- For periodic IN endpoints, one-frame data must be prefetched before the data transfer in the next frame. This can be done by enabling periodic IN endpoint 1 frame before the scheduling of the frame to be transmitted.
- The complete data to be transmitted in a frame must be written to the transmit FIFO by the application before the periodic IN token is received. Even when one-WORD data to be transmitted per frame is missing in the transmit FIFO while the periodic IN token is received, the controller behaves as when the FIFO is empty. When the transmit FIFO is empty, a zero-length data packet would be transmitted on the USB, and An NAK handshake signal would be transmitted for INTR IN endpoints.

### 【Internal data flow】

- The application must set the transfer size and packet count bits of the endpoint registers, and enable the endpoint to transmit the data.
- The application must also write the required data to the associated transmit FIFO.
- Each time the application writes a packet to the transmit FIFO, the transfer size for the endpoint is decremented by the packet size. Continue to write data until the transfer size for the endpoint becomes 0
- When an IN token for a periodic endpoint is received, the application writes the data to the FIFO (If any). If the complete data for the frame is not present in the FIFO, the controller generates an INTKNTXFEMP interrupt.

- A zero-length data packet is transmitted on the USB for synchronous IN endpoints
  - An NAK handshake signal is transmitted on the USB for interrupt IN endpoints.
5. The packet count for the endpoints is decremented by one under the following conditions:
- For synchronous endpoints, when a zero-or non-zero-length data packet is transmitted
  - For interrupt endpoints, when an ACK handshake is transmitted
  - When the transfer size and packet count are both 0, the transfer complete interrupt for the endpoint is generated and the endpoint enable bit is cleared.
6. In the “Periodic frame interval” (by the PERFRINT bit in the OTGFS\_DCFG register), when the controller finds non-empty any one of the IN endpoint FIFOs scheduled for the current frame non-empty, the controller generates an INCOMPISOIN interrupt in the OTGFS\_GINTSTS register.

#### **【Application programming sequence (frame transfers)】**

1. Program the OTGFS\_DIEPTSIZE register
2. Program the OTGFS\_DIEPTCTL register based on endpoint characteristics, and set the CNAK and endpoint enable bits
3. Write the data to be transmitted into the transmit FIFO.
4. The assertion of the INTKNTXFEMP interrupt indicates that the application has not yet written all data to be transferred into the transmit FIFO.
5. If the interrupt endpoint is already enabled while this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint to transmit data on the next IN token. If it is enabled while the interrupt is detected, refer to “Incomplete synchronous IN data transfers”.
6. When the interrupt IN endpoint is set as a periodic endpoint, the controller internally can process the timeout on the interrupt IN endpoint, without the need of the application intervention. Therefore, the application can never detect the TIMEOUT interrupt (in the OTGFS\_DIEPINTx register) on the periodic interrupt IN endpoints.
7. The assertion of the XFERC interrupt in the OTGFS\_DIEPINTx register but without the INTKNTXFEMP interrupt indicates the successful completion of a synchronous IN transfer. When reading the OTGFS\_DIEPTSIZE register, only transfer size = 0 and packet count = 0 indicate that all data are transmitted on the USB line.
8. The assertion of the XFERC interrupt in the OTGFS\_DIEPINTx register, with or without the INTKNTXFEMP interrupt, indicates the successful completion of an interrupt IN transfer. When reading the OTGFS\_DIEPTSIZE register, only transfer size = 0 and packet count = 0 indicate that all data are transmitted on the USB line.
9. The assertion of the INCOMPISOIN interrupt but without the above-mentioned interrupts indicates that the controller did not receive at least one periodic IN token in the current frame. Refer to “Incomplete synchronous IN data transfers” for more information on synchronous IN endpoints.

## OTGFS control and status registers

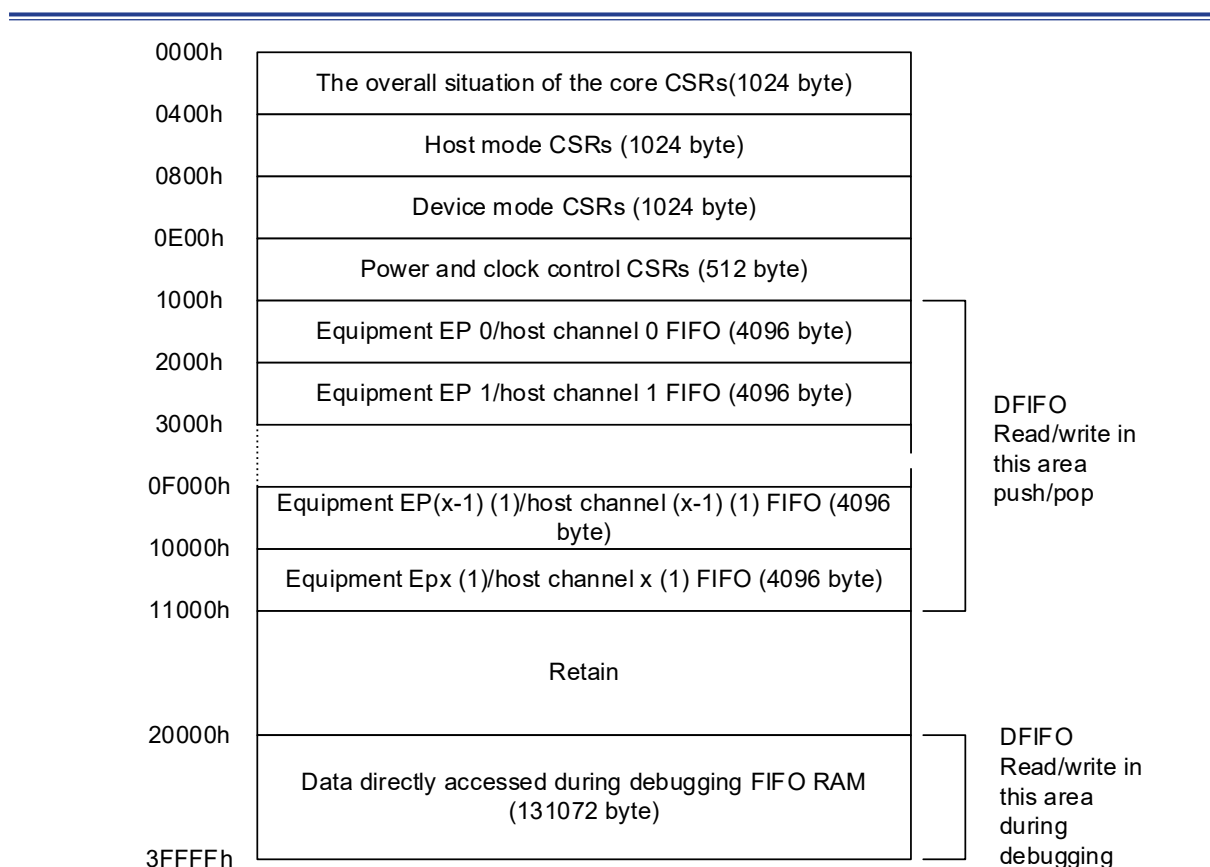
The application controls the OTGFS controller by reading from and writing to the control and status registers (CSRx) through the AHB slave interface. These registers are accessible by 32 bits, and the addresses are 32-bit aligned.

Only the controller global, power and clock control, data FIFO access and host port control and status registers are active in both host and device modes. When the OTGFS controller operates in either host or device mode, the application must not access the register group from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and the MODMIS bit (in the OTGFS\_GINTSTS register) is affected.

When the controller switches from one mode to the other, the registers in the new mode must be re-initialized as they are after a power-on reset. These peripheral registers must be accessed by words (32-bit)

### CSR register map

The host and device mode registers occupy different addresses. All registers are located in the AHB clock domain



**Figure 251. CSR memory map**

x = 7 in device mode, x = 15 in host mode.

The OTGFS control and status registers contain OTGFS global register, host mode register, device mode register, data FIFO register, power and clock control register.

1. OTGFS global registers: They are active in both host and device modes. The register acronym is G.
2. Host-mode registers: They must be programmed every time the controller changes to host mode, The register acronym is H.
3. Device-mode registers: They must be programmed every time the controller changes to device mode, The register acronym is D.
4. Data FIFO access registers: These registers are valid in both in host and device modes, and are used to read or write the FIFO for a specific endpoint or channel in a given direction. If a host channel is of type IN, the FIFO can only be read. Similarly, if a host channel is of type OUT, the FIFO can only be written.
5. Power and clock control register: There is only one register for power and clock control. It is valid in both host and device modes.

## OTGFS register address map

Table 78 shows the USB OTG register map and their reset values.

These peripheral registers must be accessed by words (32-bit)

**Table 78. OTGFS register map and reset values**

| Register        | Offset | Reset value |
|-----------------|--------|-------------|
| OTGFS_GOTGCTL   | 0x000  | 0x0001 0000 |
| OTGFS_GOTGINT   | 0x004  | 0x0000 0000 |
| OTGFS_GAHBCFG   | 0x008  | 0x0000 0000 |
| OTGFS_GUSBCFG   | 0x00C  | 0x0000 1440 |
| OTGFS_GRSTCTL   | 0x010  | 0x8000 0000 |
| OTGFS_GINTSTS   | 0x014  | 0x0400 0020 |
| OTGFS_GINTMSK   | 0x018  | 0x0000 0000 |
| OTGFS_GRXSTSR   | 0x01C  | 0x0000 0000 |
| OTGFS_GRXSTSP   | 0x020  | 0x0000 0000 |
| OTGFS_GRXFSIZ   | 0x024  | 0x0000 0200 |
| OTGFS_GNPTXFSIZ | 0x028  | 0x0000 0200 |
| OTGFS_GNPTXSTS  | 0x02C  | 0x0008 0200 |
| OTGFS_GCCFG     | 0x038  | 0x0000 0000 |
| OTGFS_GUID      | 0x03C  | 0x0000 1000 |
| OTGFS_HPTXFSIZ  | 0x100  | 0x0200 0600 |
| OTGFS_DIEPTXF1  | 0x104  | 0x0200 0400 |
| OTGFS_DIEPTXF2  | 0x108  | 0x0200 0400 |
| OTGFS_DIEPTXF3  | 0x10C  | 0x0200 0400 |
| OTGFS_DIEPTXF4  | 0x110  | 0x0200 0400 |
| OTGFS_DIEPTXF5  | 0x114  | 0x0200 0400 |
| OTGFS_DIEPTXF6  | 0x118  | 0x0200 0400 |
| OTGFS_DIEPTXF7  | 0x11C  | 0x0200 0400 |
| OTGFS_DIEPTXF8  | 0x120  | 0x0200 0400 |
| OTGFS_DIEPTXF9  | 0x124  | 0x0200 0400 |

| Register        | Offset | Reset value |
|-----------------|--------|-------------|
| OTGFS_DIEPTXF10 | 0x128  | 0x0200 0400 |
| OTGFS_DIEPTXF11 | 0x12C  | 0x0200 0400 |
| OTGFS_DIEPTXF12 | 0x130  | 0x0200 0400 |
| OTGFS_DIEPTXF13 | 0x134  | 0x0200 0400 |
| OTGFS_DIEPTXF14 | 0x138  | 0x0200 0400 |
| OTGFS_DIEPTXF15 | 0x13C  | 0x0200 0400 |
| OTGFS_DIEPTXF16 | 0x140  | 0x0200 0400 |
| OTGFS_HCFG      | 0x400  | 0x0000 0000 |
| OTGFS_HFIR      | 0x404  | 0x0000 EA60 |
| OTGFS_HFNUM     | 0x408  | 0x0000 3FFF |
| OTGFS_HPTXSTS   | 0x410  | 0x0008 0100 |
| OTGFS_HAINT     | 0x414  | 0x0000 0000 |
| OTGFS_HAINTMSK  | 0x418  | 0x0000 0000 |
| OTGFS_HPRT      | 0x440  | 0x0000 0000 |
| OTGFS_HCCHAR0   | 0x500  | 0x0000 0000 |
| OTGFS_HCINT0    | 0x508  | 0x0000 0000 |
| OTGFS_HCINTMSK0 | 0x50C  | 0x0000 0000 |
| OTGFS_HCTSIZ0   | 0x510  | 0x0000 0000 |
| OTGFS_HCCHAR1   | 0x520  | 0x0000 0000 |
| OTGFS_HCINT1    | 0x528  | 0x0000 0000 |
| OTGFS_HCINTMSK1 | 0x52C  | 0x0000 0000 |
| OTGFS_HCTSIZ1   | 0x530  | 0x0000 0000 |
| OTGFS_HCCHAR2   | 0x540  | 0x0000 0000 |
| OTGFS_HCINT2    | 0x548  | 0x0000 0000 |
| OTGFS_HCINTMSK2 | 0x54C  | 0x0000 0000 |
| OTGFS_HCTSIZ2   | 0x550  | 0x0000 0000 |
| OTGFS_HCCHAR3   | 0x560  | 0x0000 0000 |
| OTGFS_HCINT3    | 0x568  | 0x0000 0000 |
| OTGFS_HCINTMSK3 | 0x56C  | 0x0000 0000 |
| OTGFS_HCTSIZ3   | 0x570  | 0x0000 0000 |
| OTGFS_HCCHAR4   | 0x580  | 0x0000 0000 |
| OTGFS_HCINT4    | 0x588  | 0x0000 0000 |
| OTGFS_HCINTMSK4 | 0x58C  | 0x0000 0000 |
| OTGFS_HCTSIZ4   | 0x590  | 0x0000 0000 |
| OTGFS_HCCHAR5   | 0x5A0  | 0x0000 0000 |
| OTGFS_HCINT5    | 0x5A8  | 0x0000 0000 |
| OTGFS_HCINTMSK5 | 0x5AC  | 0x0000 0000 |
| OTGFS_HCTSIZ5   | 0x5B0  | 0x0000 0000 |
| OTGFS_HCCHAR6   | 0x5C0  | 0x0000 0000 |
| OTGFS_HCINT6    | 0x5C8  | 0x0000 0000 |
| OTGFS_HCINTMSK6 | 0x5CC  | 0x0000 0000 |
| OTGFS_HCTSIZ6   | 0x5D0  | 0x0000 0000 |
| OTGFS_HCCHAR7   | 0x5E0  | 0x0000 0000 |
| OTGFS_HCINT7    | 0x5E8  | 0x0000 0000 |
| OTGFS_HCINTMSK7 | 0x5EC  | 0x0000 0000 |



| Register         | Offset | Reset value |
|------------------|--------|-------------|
| OTGFS_HCTSIZ7    | 0x5F0  | 0x0000 0000 |
| OTGFS_HCCHAR8    | 0x600  | 0x0000 0000 |
| OTGFS_HCINT8     | 0x608  | 0x0000 0000 |
| OTGFS_HCINTMSK8  | 0x60C  | 0x0000 0000 |
| OTGFS_HCTSIZ8    | 0x610  | 0x0000 0000 |
| OTGFS_HCCHAR9    | 0x620  | 0x0000 0000 |
| OTGFS_HCINT9     | 0x628  | 0x0000 0000 |
| OTGFS_HCINTMSK9  | 0x62C  | 0x0000 0000 |
| OTGFS_HCTSIZ9    | 0x630  | 0x0000 0000 |
| OTGFS_HCCHAR10   | 0x640  | 0x0000 0000 |
| OTGFS_HCINT10    | 0x648  | 0x0000 0000 |
| OTGFS_HCINTMSK10 | 0x64C  | 0x0000 0000 |
| OTGFS_HCTSIZ10   | 0x650  | 0x0000 0000 |
| OTGFS_HCCHAR11   | 0x660  | 0x0000 0000 |
| OTGFS_HCINT11    | 0x668  | 0x0000 0000 |
| OTGFS_HCINTMSK11 | 0x66C  | 0x0000 0000 |
| OTGFS_HCTSIZ11   | 0x670  | 0x0000 0000 |
| OTGFS_HCCHAR12   | 0x680  | 0x0000 0000 |
| OTGFS_HCINT12    | 0x688  | 0x0000 0000 |
| OTGFS_HCINTMSK12 | 0x68C  | 0x0000 0000 |
| OTGFS_HCTSIZ12   | 0x690  | 0x0000 0000 |
| OTGFS_HCCHAR13   | 0x6A0  | 0x0000 0000 |
| OTGFS_HCINT13    | 0x6A8  | 0x0000 0000 |
| OTGFS_HCINTMSK13 | 0x6AC  | 0x0000 0000 |
| OTGFS_HCTSIZ13   | 0x6B0  | 0x0000 0000 |
| OTGFS_HCCHAR14   | 0x6C0  | 0x0000 0000 |
| OTGFS_HCINT14    | 0x6C8  | 0x0000 0000 |
| OTGFS_HCINTMSK14 | 0x6CC  | 0x0000 0000 |
| OTGFS_HCTSIZ14   | 0x6D0  | 0x0000 0000 |
| OTGFS_HCCHAR15   | 0x6E0  | 0x0000 0000 |
| OTGFS_HCINT15    | 0x6E8  | 0x0000 0000 |
| OTGFS_HCINTMSK15 | 0x6EC  | 0x0000 0000 |
| OTGFS_HCTSIZ15   | 0x6F0  | 0x0000 0000 |
| OTGFS_DCFG       | 0x800  | 0x0820 0000 |
| OTGFS_DCTL       | 0x804  | 0x0000 0002 |
| OTGFS_DSTS       | 0x808  | 0x0000 0010 |
| OTGFS_DIEPMSK    | 0x810  | 0x0000 0000 |
| OTGFS_DOEPMSK    | 0x814  | 0x0000 0000 |
| OTGFS_DAIN       | 0x818  | 0x0000 0000 |
| OTGFS_DAINMSK    | 0x81C  | 0x0000 0000 |
| OTGFS_DIEPEMPMSK | 0x834  | 0x0000 0000 |
| OTGFS_DIEPCTL0   | 0x900  | 0x0000 0000 |
| OTGFS_DIEPINT0   | 0x908  | 0x0000 0080 |
| OTGFS_DIEPTSIZ0  | 0x910  | 0x0000 0000 |
| OTGFS_DTXFSTS0   | 0x918  | 0x0000 0200 |



| Register        | Offset | Reset value |
|-----------------|--------|-------------|
| OTGFS_DIEPCTL1  | 0x920  | 0x0000 0000 |
| OTGFS_DIEPINT1  | 0x928  | 0x0000 0080 |
| OTGFS_DIEPTSIZ1 | 0x930  | 0x0000 0000 |
| OTGFS_DTXFSTS1  | 0x938  | 0x0000 0200 |
| OTGFS_DIEPCTL2  | 0x940  | 0x0000 0000 |
| OTGFS_DIEPINT2  | 0x948  | 0x0000 0080 |
| OTGFS_DIEPTSIZ2 | 0x950  | 0x0000 0000 |
| OTGFS_DTXFSTS2  | 0x958  | 0x0000 0200 |
| OTGFS_DIEPCTL3  | 0x960  | 0x0000 0000 |
| OTGFS_DIEPINT3  | 0x968  | 0x0000 0080 |
| OTGFS_DIEPTSIZ3 | 0x970  | 0x0000 0000 |
| OTGFS_DTXFSTS3  | 0x978  | 0x0000 0200 |
| OTGFS_DIEPCTL4  | 0x980  | 0x0000 0000 |
| OTGFS_DIEPINT4  | 0x988  | 0x0000 0080 |
| OTGFS_DIEPTSIZ4 | 0x990  | 0x0000 0000 |
| OTGFS_DTXFSTS4  | 0x998  | 0x0000 0200 |
| OTGFS_DIEPCTL5  | 0x9A0  | 0x0000 0000 |
| OTGFS_DIEPINT5  | 0x9A8  | 0x0000 0080 |
| OTGFS_DIEPTSIZ5 | 0x9B0  | 0x0000 0000 |
| OTGFS_DTXFSTS5  | 0x9B8  | 0x0000 0200 |
| OTGFS_DIEPCTL6  | 0x9C0  | 0x0000 0000 |
| OTGFS_DIEPINT6  | 0x9C8  | 0x0000 0080 |
| OTGFS_DIEPTSIZ6 | 0x9D0  | 0x0000 0000 |
| OTGFS_DTXFSTS6  | 0x9D8  | 0x0000 0200 |
| OTGFS_DIEPCTL7  | 0x9E0  | 0x0000 0000 |
| OTGFS_DIEPINT7  | 0x9E8  | 0x0000 0080 |
| OTGFS_DIEPTSIZ7 | 0x9F0  | 0x0000 0000 |
| OTGFS_DTXFSTS7  | 0x9F8  | 0x0000 0200 |
| OTGFS_DOEPCTL0  | 0xB00  | 0x0000 8000 |
| OTGFS_DOEPINT0  | 0xB08  | 0x0000 0000 |
| OTGFS_DOEPTSIZ0 | 0xB10  | 0x0000 0000 |
| OTGFS_DOEPCTL1  | 0xB20  | 0x0000 0000 |
| OTGFS_DOEPINT1  | 0xB28  | 0x0000 0000 |
| OTGFS_DOEPTSIZ1 | 0xB30  | 0x0000 0000 |
| OTGFS_DOEPCTL2  | 0xB40  | 0x0000 0000 |
| OTGFS_DOEPINT2  | 0xB48  | 0x0000 0000 |
| OTGFS_DOEPTSIZ2 | 0xB50  | 0x0000 0000 |
| OTGFS_DOEPCTL3  | 0xB60  | 0x0000 0000 |
| OTGFS_DOEPINT3  | 0xB68  | 0x0000 0000 |
| OTGFS_DOEPTSIZ3 | 0xB70  | 0x0000 0000 |
| OTGFS_DOEPCTL4  | 0xB80  | 0x0000 0000 |
| OTGFS_DOEPINT4  | 0xB88  | 0x0000 0000 |
| OTGFS_DOEPTSIZ4 | 0xB90  | 0x0000 0000 |
| OTGFS_DOEPCTL5  | 0xBA0  | 0x0000 0000 |
| OTGFS_DOEPINT5  | 0xBA8  | 0x0000 0000 |

| Register        | Offset | Reset value |
|-----------------|--------|-------------|
| OTGFS_DOEPTSZ5  | 0xBB0  | 0x0000 0000 |
| OTGFS_DOEPTCTL6 | 0xBC0  | 0x0000 0000 |
| OTGFS_DOEPINT6  | 0xBC8  | 0x0000 0000 |
| OTGFS_DOEPTSZ6  | 0xBD0  | 0x0000 0000 |
| OTGFS_DOEPTCTL7 | 0xBE0  | 0x0000 0000 |
| OTGFS_DOEPINT7  | 0xBE8  | 0x0000 0000 |
| OTGFS_DOEPTSZ7  | 0xBF0  | 0x0000 0000 |
| OTGFS_PCGCCTL   | 0xE00  | 0x0000 0000 |

## OTGFS global registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between two modes.

### OTGFS status and control register (OTGFS\_GOTGCTL)

This register controls the OTG function and reflects its status.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:22 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 21    | CURMOD   | 0x0         | ro   | Current Mode of Operation<br>Accessible in both host and device modes<br>This bit indicates the current operation mode.<br>0: Device mode<br>1: Host mode  |
| Bit 20:17 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 16    | CONIDSTS | 0x1         | ro   | Accessible in both host and device modes<br>Connector ID status<br>This bit indicates the connector ID status.<br>0: OTGFS controller is in A-device mode<br>1: OTGFS controller is in B-device mode |
| Bit 15:0  | Reserved | 0x0000      | resd | Kept at its default value.   |

### OTGFS interrupt status control register (OTGFS\_GOTGINT)

The application reads this register to know about which kind of OTG interrupt is generated, and writes this register to clear the OTG interrupt.

| Bit      | Name       | Reset value | Type | Description  |
|----------|------------|-------------|------|--|
| Bit 31:3 | Reserved   | 0x0000      | resd | Kept at its default value.   |
| Bit 2    | SESENDDDET | 0x0         | rw1c | Available in both host and device modes<br>Session end detected<br>The controller sets this bit when a Bvalid (Vbus) signal is disconnected. This register can only be set by hardware. Writing 1 by software clears this bit. |
| Bit 1:0  | Reserved   | 0x0000      | resd | Kept at its default value.   |

### OTGFS AHB configuration register (OTGFS\_GAHBCFG)

This register is used to configure the controller after power-on or mode change. This register mainly contains AHB-related parameters. Do not change this register after the initial configuration. The application must configure this register before starting transmission on either the AHB or USB.

| Bit      | Name        | Reset value | Type | Description   |
|----------|-------------|-------------|------|---|
| Bit 31:9 | Reserved    | 0x000000    | resd | Kept at its default value.  |
| Bit 8    | PTXFEMPLVL  | 0x0         | rw   | Accessible in host mode only<br>Periodic TxFIFO empty level<br>It indicates when the periodic TxFIFO empty interrupt bit in the GINTSTS register is triggered.<br>0: PTXFEMP (GINTSTS) interrupt indicates that the periodic TxFIFO is half empty<br>1: PTXFEMP (GINTSTS) interrupt indicates that the periodic TxFIFO is fully empty   |
| Bit 7    | NPTXFEMPLVL | 0x0         | rw   | Accessible in both host mode and device modes<br>Non-Periodic TxFIFO empty level<br>In host mode, this bit indicates when the non-periodic TxFIFO empty interrupt (NPTXFEMP in GINTSTS) is triggered.<br>In device mode, this bit indicates when the IN endpoint TxFIFO empty interrupt (TXFEMP bit in DIEPINTn) is triggered.<br>0: The TxFEMP (in DIEPINTn) interrupt indicates that the IN endpoint TxFIFO is half empty<br>1: The TxFEMP (in DIEPINTn) interrupt indicates that the IN endpoint TxFIFO is fully empty |
| Bit 6:1  | Reserved    | 0x00        | resd | Kept at its default value.  |
| Bit 0    | GLBINTMSK   | 0x0         | rw   | Accessible in both host mode and device modes<br>Global interrupt mask<br>The application uses this bit to mask or unmask the interrupts sent by the interrupt line to itself.<br>0: Mask the interrupts sent to the application<br>1: Unmask the interrupts sent to the application  |

### OTGFS USB configuration register (OTGFS\_GUSBCFG)

This register is used to configure the controller after power-on or a change between host mode and device mode. This register contains USB and USB-PHY related parameters. The application must program the register before handling any transaction on either the AHB or USB. Do not change this register after the initial configuration.

| Bit    | Name     | Reset value | Type | Description  |
|--------|----------|-------------|------|--|
| Bit 31 | COTXPKT  | 0x0         | rw   | Accessible in both host mode and device modes<br>Corrupt Tx packet<br>This bit is for debug purpose only. Do not set this bit to 1.  |
| Bit 30 | FDEVMODE | 0x0         | rw   | Accessible in both host mode and device modes<br>Force device mode<br>Writing 1 to this bit forces the controller to go into device mode, irrespective of the status of the ID input point.<br>0: Normal mode<br>1: Force device mode<br>After setting this bit, the application must wait at least 25 ms before the configuration takes effect. |

| Bit       | Name      | Reset value | Type | Description   |
|-----------|-----------|-------------|------|---|
| Bit 29    | FHSTMODE  | 0x0         | rw   | <p>Accessible in both host mode and device modes</p> <p>Force host mode</p> <p>Writing 1 to this bit forces the controller to go into host mode, irrespective of the status of the ID input point.</p> <p>0: Normal mode<br/>1: Force host mode</p> <p>After setting this bit, the application must wait at least 25 ms before the configuration takes effect.</p>  |
| Bit 28:15 | Reserved  | 0x0000      | resd | Kept at its default value.  |
| Bit 14    | Reserved  | 0x0         | resd | Kept at its default value.  |
| Bit 13:10 | USBTRDTIM | 0x5         | rw   | <p>Accessible in device mode</p> <p>USB Turnaround Time</p> <p>This field sets the turnaround time in PHY clocks. It defines the response time when the MAC sends a request to the packet FIFO controller (PFC) to fetch data from the DFIFO (SPRAM). These bits must be configured as follows:</p> <p>0101: When the MAC interface is 16-bit UTMI+<br/>1001: When the MAC interface is 8-bit UTMI+</p> <p>Note: The aforementioned values are calculated based on a minimum of 30 MHz AHB frequency. The USB turnaround time is critical for certifications with long cables and 5-Hub. If you want the AHB to run below 30 MHz, and don't care about the USB turnaround time, you can set larger values for these bits.</p> |
| Bit 9:3   | Reserved  | 0x08        | resd | Kept at its default value.  |
| Bit 2:0   | TOUTCAL   | 0x0         | rw   | <p>Accessible in both host mode and device modes</p> <p>FS Timeout calibration</p> <p>The number of PHY clocks that the application programs in these bits is added to the full-speed interpacket timeout duration in order to compensate for any additional latency introduced by the PHY. This action can be required, because the delay triggered by the PHY while generating the line state condition can vary from one PHY to another.</p> <p>In full-speed mode, the USB standard timeout value is 16 ~ 18 (inclusive) bit times. The application must program these bits based on the enumeration speed. The number of bit times added per PHY clock is 0.25 bit times.</p>  |

### OTGFS reset register (OTGFS\_GRSTCTL)

The application resets various hardware modules in the controller through this register.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31    | AHBIDLE  | 0x1         | ro   | <p>Accessible in both host mode and device modes</p> <p>AHB master Idle</p> <p>This bit indicates that the AHB master state machine is in idle condition.</p> |
| Bit 30:11 | Reserved | 0x000       | resd | Kept at its default value.  |

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 10:6 | TXFNUM   | 0x00        | rw   | <p>Accessible in both host mode and device modes</p> <p>TxFIFO number</p> <p>This field indicates the FIFO number that must be refreshed through the TxFIFO Flush bit. Do not make changes to this field until the controller clears the TxFIFO Flush bit.</p> <p>00000:<br/>– Non-periodic TxFIFO in host mode<br/>– Tx FIFO 0 in device mode</p> <p>00001:<br/>– Periodic TxFIFO in host mode<br/>– TxFIFO 1 in device mode</p> <p>00010:<br/>– TxFIFO 2 in device mode</p> <p>...</p> <p>01111:<br/>– TxFIFO 15 in device mode</p> <p>10000:<br/>– Refresh all the transmit FIFOs in device or host mode</p>   |
| Bit 5    | TXFFLSH  | 0x0         | rw1s | <p>Accessible in both host mode and device modes</p> <p>TxFIFO Flush</p> <p>This bit selectively refreshes a single or all transmit FIFOs, but can do so when the controller is not in the process of a transaction.</p> <p>The application must write this bit only after checking that the controller is neither writing to nor reading from the TxFIFO.</p> <p>Verify using these registers:<br/>Read: NAK effective interrupt (NAK Effective Interrupt) ensures that the controller is not reading from the FIFO<br/>Write: AHBIDLE bit in GRSTCTL ensures that the controller is not writing to the FIFO.</p> <p>For FIFO reprogramming, it is usually recommended to carry out flushing operation.</p> <p>In device endpoint disable state, it is also advised to use FIFO flushing operation. The application must wait until the controller clears this bit, before performing other operations. It takes 8 clocks to clear this bit (slowest of phy_clk or hclk)</p> |
| Bit 4    | RXFFLSH  | 0x0         | rw1s | <p>Accessible in both host mode and device modes</p> <p>RxFIFO flush</p> <p>The application can refresh the entire RxFIFO using this bit, but must first ensure that the controller is not in the process of a transaction. The application must only write to this bit after checking that the controller is neither reading from nor writing to the RxFIFO.</p> <p>The application must wait until the controller clears this bit, before performing other operations. It takes 8 clocks to clear this bit (slowest of PHY or AHB)</p>  |
| Bit 3    | Reserved | 0x0         | resd | Kept at its default value.  |

| Bit   | Name      | Reset value | Type | Description  |
|-------|-----------|-------------|------|--|
| Bit 2 | FRMCNTRST | 0x0         | rw1s | <p>Accessible in both host mode and device modes</p> <p>Host frame counter reset</p> <p>The application uses this bit to reset the frame number counter inside the controller. After the frame counter is reset, the subsequent SOS sent out by the controller has a frame number of 0.</p> <p>If the application writes 1 to this bit, it may not be able to read the value, because this bit is cleared after a few clock cycles by the controller</p>   |
| Bit 1 | PIUSFTRST | 0x0         | rw1s | <p>Accessible in both host mode and device modes</p> <p>PIU FS dedicated controller soft reset</p> <p>This bit is used to reset PIU full-speed dedicated controller</p> <p>All state machines in the PIU full-speed dedicated controller are reset to the idle state. When the PHY remains in the receive state for more than one-frame time due to PHY errors (such as operation interrupted or babble), this bit can be used to reset the PIU full-speed dedicated controller.</p> <p>This is can be cleared automatically, the controller this clear this bit after all the necessary logic is reset in the controller.</p>   |
| Bit 0 | CSFTRST   | 0x0         | rw1s | <p>Accessible in both host mode and device modes</p> <p>Controller soft reset</p> <p>Resets the hclk and phy_clock domain as follows:</p> <p>Clears all interrupts and CSR registers except for the following bits:</p> <ul style="list-style-type: none"> <li>- HCFG.FSLSPCS</li> <li>- DCFG.DECSPD</li> <li>- DCTL.SFTDIS</li> </ul> <p>Resets all state machines (except AHB slave) to the idle state, and clears all the transmit and receive FIFOs.</p> <p>All transactions on the AHB master are terminated as soon as possible after completing the last phase of an AHB data transfer. All transactions on the USB are terminated immediately.</p> <p>The application can write to this bit at any time to reset the controller. This is can be cleared automatically, the controller this clear this bit after all the necessary logic is reset in the controller. The controller could take several clocks to clear this bit, depending on the current state of the controller. Once this bit is cleared, the application must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay).</p> <p>Additionally, the application must ensure that the bit 31 in this register is set (AHB master is in idle state) before performing other operations.</p> <p>Typically, the software set is used during software development and also when the user dynamically changes the PHY selection bits in the above-listed USB configuration registers. To change the PHY, the corresponding PHY clock is selected and used in the PHY domain. After a new clock is selected, the PHY domain has to be reset for normal operation.</p> |

### OTGFS interrupt register (OTGFS\_GINTSTS)

This register interrupts the application due to system-level events in the current mode (device or host mode), as shown in Figure 240.

Some of the bits in this register are valid only in host mode, while others are valid in device mode only. Besides, this register indicates the current mode.

The FIFO status interrupts are read-only. The FIFO interrupt conditions are cleared automatically as soon as the software reads from or writes to the FIFO while processing these interrupts.

The application must clear the GINTSTS register at initialization before enabling an interrupt bit to avoid any interrupt generation prior to initialization.

| Bit    | Name      | Reset value | Type | Description   |
|--------|-----------|-------------|------|---|
| Bit 31 | WKUPINT   | 0x0         | rw1c | Accessible in both host mode and device modes<br>Resume/Remote wakeup detected interrupt)<br>In device mode, this interrupt is generated only when a resume signal (triggered by host) is detected on the USB bus.<br>In host mode, this interrupt is generated only when a remote wakeup signal (triggered by device) is detected on the USB bus.  |
| Bit 30 | Reserved  | 0x0         | resd | Kept at its default value.  |
| Bit 29 | DISCONINT | 0x0         | rw1c | Accessible in host mode only<br>Disconnect detected interrupt<br>The interrupt is generated when a device disconnect is detected.   |
| Bit 28 | CONIDSCHG | 0x0         | rw1c | Accessible in both host mode and device modes<br>Connector ID status change<br>This bit is set by the controller when there is a change in connector ID status.   |
| Bit 27 | Reserved  | 0x0         | resd | Kept at its default value.  |
| Bit 26 | PTXFEMP   | 0x1         | ro   | Accessible in host mode only<br>Periodic Tx FIFO Empty<br>The interrupt is generated when the Periodic Transmit FIFO is either half or completely empty and there is space for a request to be written in the periodic request queue. The half or completely empty status depends on the periodic transmit FIFO empty level bit in the AHB configuration register.  |
| Bit 25 | HCHINT    | 0x0         | ro   | Host channel interrupt<br>The controller sets this bit to indicate that an interrupt is pending on one of the channels in the controller (in host mode). The application must read the Host All Channels Interrupt register to determine the exact number of the channel on which the interrupt occurred, and then read the Host Channel-n Interrupt register to determine the interrupt event source.<br>The application must clear the corresponding status bit in the HCINTn (Host All Channels Interrupt) register to clear this bit. |

| Bit       | Name                     | Reset value | Type | Description   |
|-----------|--------------------------|-------------|------|---|
| Bit 24    | PRTINT                   | 0x0         | ro   | Host port interrupt<br>The controller sets this bit to indicate a change in port status one of the ports. The application must read the Host Port Control and Status register to determine the exact event source. The application must clear the Host Port Control and Status register to clear this bit.  |
| Bit 23:22 | Reserved                 | 0x0         | resd | Kept at its default value.  |
| Bit 21    | INCOMPIP<br>INCOMPISOOUT | 0x0         | rw1c | Incomplete periodic transfer<br>Accessible in host mode only<br>In host mode, the controller sets this interrupt bit when there are incomplete periodic transfers still pending in the current frame.<br>Incomplete Isochronous OUT Transfer<br>Accessible in device mode only<br>In device mode, the controller sets this interrupt bit to indicate that there is at least one synchronous OUT endpoint with incomplete transfers in the current frame. This interrupt is generated along with the End of Periodic Frame Interrupt bit in this register.   |
| Bit 20    | INCOMPISOIN              | 0x0         | rw1c | Accessible in device mode only<br>Incomplete Isochronous IN Transfer<br>The controller sets this interrupt to indicate that there is at least one synchronous IN endpoint with incomplete transfers in the current frame. This interrupt is generated along with the End of Periodic Frame Interrupt bit in this register.  |
| Bit 19    | OEPTINT                  | 0x0         | ro   | Accessible in device mode only<br>OUT endpoints interrupt<br>The controller sets this bit to indicate that an interrupt is pending on one of the OUT endpoints in the controller. The application must read the Device All Endpoints Interrupt register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding Device OUT Endpoint-n Interrupt register to determine the exact source of the interrupt. The application must clear the corresponding status bit in the corresponding Device OUT Endpoint-n Interrupt register to clear this bit.          |
| Bit 18    | IEPTINT                  | 0x0         | ro   | Accessible in device mode only<br>IN Endpoints interrupt<br>The controller sets this bit to indicate that an interrupt is pending one of the IN endpoints in the controller (in device mode). The application must read the Device All Endpoints Interrupt register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding Device IN Endpoint-n Interrupt register to determine the exact source of the interrupt. The application must clear the corresponding status bit in the corresponding Device IN Endpoint-n Interrupt register to clear this bit. |
| Bit 17:16 | Reserved                 | 0x0         | resd | Kept at its default value.  |



| Bit     | Name       | Reset value | Type | Description  |
|---------|------------|-------------|------|--|
| Bit 15  | EOPF       | 0x0         | rw1c | Accessible in device mode only<br>End of periodic frame interrupt<br>This bit indicates that the period programmed in the periodic frame interval bit of the Device Configuration register has been reached in the current frame.  |
| Bit 14  | ISOOUTDROP | 0x0         | rw1c | Accessible in device mode only<br>Isochronous OUT packet dropped interrupt)<br>The controller sets this bit on the following condition: the controller fails to write a synchronous OUT packet into the receive FIFO because the receive FIFO does not have enough space to accommodate a maximum size packet for the synchronous OUT endpoint.  |
| Bit 13  | ENUMDONE   | 0x0         | rw1c | Accessible in device mode only<br>Enumeration done<br>The controller sets this bit to indicate that speed enumeration is done.<br>The application must read the Device Status register to obtain the enumeration speed.  |
| Bit 12  | USBRST     | 0x0         | rw1c | Accessible in device mode only<br>USB Reset<br>The controller sets this bit to indicate that a reset is detected on the USB bus.   |
| Bit 11  | USBSUSP    | 0x0         | rw1c | Accessible in device mode only<br>USB Suspend<br>The controller sets this bit to indicate that a suspend is detected on the USB bus. The controller enters the Suspend state when there is no activity on the bus for a long period of time.   |
| Bit 10  | ERLYSUSP   | 0x0         | rw1c | Accessible in device mode only<br>Early suspend<br>The controller sets this bit to indicate that the idle state has been detected on the USB bus for 3 ms.   |
| Bit 9:8 | Reserved   | 0x0         | resd | Kept at its default value.   |
| Bit 7   | GOUTNAKEFF | 0x0         | ro   | Accessible in device mode only<br>Global OUT NAK effective<br>This bit indicates that the Set Global OUT NAK bit in the Device Control register (set by the application) has taken effect. This bit can be cleared by writing the Clear Global OUT NAK bit in the Device Control register.   |
| Bit 6   | GINNAKEFF  | 0x0         | ro   | Accessible in device mode only<br>Global IN Non-periodic NAK effective<br>This bit indicates that the Set Global Non-periodic IN NA bit in the Device Control register (set by the application) has taken effect. That is, the controller has sampled the Global IN NAK bit set by the application. This bit can be cleared by writing the Clear Global Non-periodic IN NA bit in the Device Control register. This interrupt does not necessarily mean that a NAK handshake signal is sent out on the USB bus. The STALL bit has priority over the NAK bit. |

| Bit   | Name     | Reset value | Type | Description  |
|-------|----------|-------------|------|--|
| Bit 5 | NPTXFEMP | 0x1         | ro   | <p>Accessible in both host and device modes</p> <p>Non-periodic TxFIFO empty</p> <p>This interrupt is generated when the Non-periodic TxFIFO is either half or completely empty and there is enough space for at least one request to be written to the Non-periodic Transmit Request Queue. The half or completely empty depends on the Non-periodic TxFIFO Empty Level bit in the Core AHB Configuration register.</p>   |
| Bit 4 | RXFLVL   | 0x0         | ro   | <p>Accessible in both host and device modes</p> <p>RxFIFO Non-Empty</p> <p>Indicates that there is at least one packet to be read from the receive FIFO.</p>   |
| Bit 3 | SOF      | 0x0         | rw1c | <p>Accessible in both host and device modes</p> <p>Start of Frame</p> <p>In host mode, the controller sets this bit to indicate that an SOF (full-speed) or Keep-Alive (low-speed) is transmitted on the USB bus. The application must set this bit to 1 to clear this interrupt.</p> <p>In device mode, the controller sets this bit to indicate that an SOF token has been received on the USB bus. The application must read the Device Status register to get the current frame number. This interrupt can be generated only when the controller is running in FS mode. This bit is set by the controller. The application must write 1 to clear this bit.</p> <p>Note: Reading this register immediately after power-on reset may return the value 0x1. If this register is read as 0x1 immediately after power-on reset, it does not mean that an SOF has been transmitted (in host mode) or received (in device mode). The reading of this register is valid only when an effective connection has been established between the host and the device. If this bit is set after power-on reset, the application can clear this bit.</p> |
| Bit 2 | OTGINT   | 0x0         | ro   | <p>Accessible in both host and device modes</p> <p>OTG interrupt</p> <p>The controller sets this bit to indicate that an OTG protocol event is generated. The application must read the OTGFS_GOTGINT register to determine the exact source that caused this interrupt. The application must clear the corresponding status bit in the OTGFS_GOTGINT register to clear this bit.</p>  |

| Bit   | Name    | Reset value | Type | Description   |
|-------|---------|-------------|------|---|
| Bit 1 | MODEMIS | 0x0         | rw1c | <p>Accessible in both host and device modes</p> <p>Mode mismatch interrupt</p> <p>The controller sets this bit when the application is attempting to access:</p> <p>A host-mode register, when the controller is running in device mode</p> <p>A device-mode register, when the controller is running in host mode</p> <p>An OKAY response occurs when the register access is completed on the AHB, but it is ignored by the controller internally, and does not affect the operation of the controller.</p> <p>This bit can be set by the controller only. The application must write 1 to clear this bit.</p> |
| Bit 0 | CURMOD  | 0x0         | ro   | <p>Accessible in both host and device modes</p> <p>Current mode of operation</p> <p>This bit indicates the current mode.</p> <p>0: Device mode</p> <p>1: Host mode</p>  |

#### OTGFS interrupt mask register (OTGFS\_GINTMSK)

This register works with the Interrupt Register to interrupt the application. When an interrupt bit is masked, the interrupt related to this interrupt bit is not generated. However, the Interrupt Register bit corresponding to this interrupt is still set.

Interrupt mask: 0

Interrupt unmask: 1

| Bit       | Name                            | Reset value | Type | Description  |
|-----------|---------------------------------|-------------|------|--|
| Bit 31    | WKUPINTMSK                      | 0x0         | rw   | <p>Accessible in both host and device modes</p> <p>Resume/Remote wakeup detected interrupt mask</p>  |
| Bit 30    | Reserved                        | 0x0         | resd | Kept at its default value.   |
| Bit 29    | DISCONINTMSK                    | 0x0         | rw   | <p>Accessible in both host and device modes</p> <p>Disconnect detected interrupt mask</p>  |
| Bit 28    | CONIDSCHGMSK                    | 0x0         | rw   | <p>Accessible in both host and device modes</p> <p>Connector ID status change mask</p>   |
| Bit 27    | Reserved                        | 0x0         | resd | Kept at its default value.   |
| Bit 26    | PTXFEMPMSK                      | 0x0         | rw   | <p>Accessible in host mode only</p> <p>Periodic TxFIFO empty mask</p>  |
| Bit 25    | HCHINTMSK                       | 0x0         | rw   | <p>Accessible in host mode only</p> <p>Host channels interrupt mask</p>  |
| Bit 24    | PRTINTMSK                       | 0x0         | ro   | <p>Accessible in host mode only</p> <p>Host port interrupt mask</p>  |
| Bit 23:22 | Reserved                        | 0x0         | resd | Kept at its default value.   |
| Bit 21    | INCOMPIPMASK<br>INCOMPISOOUTMSK | 0x0         | rw   | <p>Incomplete periodic transfer mask</p> <p>Accessible in host mode only</p> <p>Incomplete isochronous OUT transfer mask</p> <p>Accessible in device mode only</p> |
| Bit 20    | INCOMISOINMSK                   | 0x0         | rw   | <p>Accessible in device mode only</p> <p>Incomplete isochronous IN transfer mask</p>   |

| Bit     | Name          | Reset value | Type | Description   |
|---------|---------------|-------------|------|---|
| Bit 19  | OEPTINTMSK    | 0x0         | rw   | Accessible in device mode only<br>OUT endpoints interrupt mask              |
| Bit 18  | IEPTINTMSK    | 0x0         | rw   | Accessible in device mode only<br>IN endpoints interrupt mask               |
| Bit 17  | Reserved      | 0x0         | rw   | Kept at its default value.  |
| Bit 16  | Reserved      | 0x0         | resd | Kept at its default value.  |
| Bit 15  | EOPFMSK       | 0x0         | rw   | Accessible in device mode only<br>End of periodic frame interrupt mask      |
| Bit 14  | ISOOUTDROPMSK | 0x0         | rw   | Device only isochronous OUT packet dropped interrupt mask                   |
| Bit 13  | ENUMDONEMSK   | 0x0         | rw   | Accessible in device mode only<br>Enumeration done mask                     |
| Bit 12  | USBRSTMSK     | 0x0         | rw   | Accessible in device mode only<br>USB Reset mask                            |
| Bit 11  | USBSUSPMSK    | 0x0         | rw   | Accessible in device mode only<br>USB suspend interrupt mask                |
| Bit 10  | ERLYSUSPMSK   | 0x0         | rw   | Accessible in device mode only<br>Early suspend interrupt mask              |
| Bit 9:8 | Reserved      | 0x0         | resd | Kept at its default value.  |
| Bit 7   | GOUTNAKEFFMSK | 0x0         | rw   | Accessible in device mode only<br>Global OUT NAK effective mask             |
| Bit 6   | GINNAKEFFMSK  | 0x0         | rw   | Accessible in device mode only<br>Global Non-periodic IN NAK effective mask |
| Bit 5   | NPTXFEMPMSK   | 0x0         | rw   | Accessible in both host and device modes<br>Non-periodic Tx FIFO empty mask |
| Bit 4   | RXFLVLMSK     | 0x0         | rw   | Accessible in both host and device modes<br>Receive FIFO Non-empty mask     |
| Bit 3   | SOFMSK        | 0x0         | rw   | Accessible in both host and device modes<br>Start of Frame mask             |
| Bit 2   | OTGINTMSK     | 0x0         | rw   | Accessible in both host and device modes<br>OTG interrupt mask              |
| Bit 1   | MODEMISMSK    | 0x0         | rw   | Accessible in both host and device modes<br>Mode mismatch interrupt mask    |
| Bit 0   | Reserved      | 0x0         | resd | Kept at its default value.  |

## OTGFS receive status debug read/OTG status read and POP registers (OTGFS\_GRXSTSR / OTGFS\_GRXSTSP)

A read to the Receive Status Debug Read register returns the data of the top of the Receive FIFO. A read to the Receive Status Read and Pop register pops the data of the top of the Receive FIFO.

The receive status contents are interpreted differently in host and device modes. Then controller ignores the receive status pop/read when the receive FIFO is empty and returns the value of 0x0000 0000. The application can only pop the receive status FIFO when the receive FIFO non-empty bit of the Core Interrupt register is set.

### Host mode:

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:21 | Reserved | 0x000       | resd | Kept at its default value.  |
| Bit 20:17 | PKTSTS   | 0x0         | ro   | Packet status<br>Indicates the status of the received data packet.<br>0010: IN data packet received<br>0011: IN transfer completed (triggers an interrupt)<br>0101: Data toggle error (triggers an interrupt)<br>0111: Channel halted (triggers an interrupt)<br>Others: Reserved<br>Reset value: 0 |
| Bit 16:15 | DPID     | 0x0         | ro   | Data PID<br>Indicates the data PID of the received data packet.<br>00: DATA0<br>10: DATA1<br>01: DATA2<br>11: MDATA<br>Reset value: 0   |
| Bit 14:4  | BCNT     | 0x000       | ro   | Byte count<br>Indicates the byte count of the received IN data packet.  |
| Bit 3:0   | CHNUM    | 0x0         | ro   | Channel number<br>Indicates the channel number to which the currently received data packet belongs.   |

### Device mode:

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:25 | Reserved | 0x00        | resd | Kept at its default value.  |
| Bit 24:21 | FN       | 0x0         | ro   | Frame number<br>Indicates the least significant 4 bits of the frame number of the data packet received on the USB bus. This field is applicable only when the synchronous OUT endpoints are supported.  |
| Bit 20:17 | PKTSTS   | 0x0         | ro   | Packet status<br>Indicates the status of the received data packet.<br>0001: Global OUT NAK (triggers an interrupt)<br>0010: OUT data packet received<br>0011: OUT transfer completed (triggers an interrupt)<br>0100: SETUP transaction completed (triggers an interrupt)<br>0110: SETUP data packet received<br>Others: Reserved |

| Bit       | Name   | Reset value | Type | Description   |
|-----------|--------|-------------|------|---|
| Bit 16:15 | DPID   | 0x0         | ro   | Data PID<br>Indicates the data PID of the received OUT data packet.<br>00: DATA0<br>10: DATA1<br>01: DATA2<br>11: MDATA |
| Bit 14:4  | BCNT   | 0x000       | ro   | Byte count<br>Indicates the byte count of the received data packet.   |
| Bit 3:0   | EPTNUM | 0x0         | ro   | Endpoint number<br>Indicates the endpoint number to which the currently received data packet belongs.                   |

#### OTGFS receive FIFO size register (OTGFS\_GRXFSIZ)

The application can program the SRAM size that must be allocated to the receive FIFO.

| Bit       | Name     | Reset value | Type  | Description   |
|-----------|----------|-------------|-------|---|
| Bit 31:16 | Reserved | 0x0000      | resd  | Kept at its default value.  |
| Bit 15:0  | RXFDEP   | 0x0200      | ro/rw | RxFIFO Depth<br>This value is in terms of 32-bit words.<br>Minimum value is 16<br>Maximum value is 512<br>The power-on reset value of this register is defined as the largest receive data FIFO depth during the configuration. |

#### OTGFS non-periodic Tx FIFO size (OTGFS\_GNPTXFSIZ) / Endpoint 0 Tx FIFO size registers (OTGFS\_DIEPTXF0)

The application can program the SRAM size and start address of the non-periodic transmit FIFO. The fields of this register varies with host mode or device mode.

##### Host:

| Bit       | Name        | Reset value | Type  | Description  |
|-----------|-------------|-------------|-------|--|
| Bit 31:16 | NPTXFDEP    | 0x0000      | ro/rw | Non-periodic TxFIFO depth<br>This value is in terms of 32-bit words.<br>Minimum value is 16<br>Maximum value is 256              |
| Bit 15:0  | NPTXFSTADDR | 0x0200      | ro/rw | Non-periodic transmit SRAM start address<br>This field contains the memory start address of the Non-periodic Transmit FIFO SRAM. |

##### Device:

| Bit       | Name           | Reset value | Type  | Description   |
|-----------|----------------|-------------|-------|---|
| Bit 31:16 | INEPT0TXDEP    | 0x0000      | ro/rw | N Endpoint TxFIFO 0 depth<br>This value is in terms of 32-bit words.<br>Minimum value is 16<br>Maximum value is 256                   |
| Bit 15:0  | INEPT0TXSTADDR | 0x0200      | ro/rw | IN Endpoint FIFO0 transmit SRAM start address<br>This field contains the memory start address of the IN Endpoint FIFO0 transmit SRAM. |

### OTGFS non-periodic Tx FIFO size/request queue status register (OTGFS\_GNPTXSTS)

This register is valid in host mode only. It is a read-only register that contains the available space information for the Non-periodic Tx FIFO and the Non-periodic Transmit Request Queue.

| Bit       | Name          | Reset value | Type | Description  |
|-----------|---------------|-------------|------|--|
| Bit 31    | Reserved      | 0x0         | resd | Kept at its default value.   |
| Bit 30:24 | NPTXQTOP      | 0x00        | ro   | Top of the Non-periodic transmit request queue<br>Indicates that the MAC is processing the request from the non-periodic transmit request queue.<br>Bit [30:27]: Channel/Endpoint number<br>Bit [26:25]:<br>00: IN/OUT token<br>01: Zero-length transmit packet (device IN/host OUT)<br>10: PING/CSPLIT token<br>11: Channel halted command<br>Bit [24]: Terminate (last request for the selected channel/endpoint)  |
| Bit 23:16 | NPTXQSPCAVAIL | 0x08        | ro   | Non-periodic transmit request queue space available<br>Indicates the amount of space available in the non-periodic transmit request queue. This queue supports both IN and OUT requests in host mode.<br>00: Non-periodic transmit request queue is full<br>01: 1 location available<br>02: 2 locations available<br>N: n locations available ( $0 \leq n \leq 8$ )<br>Others: Reserved<br>Reset value: Configurable |
| Bit 15:0  | NPTXFSPCAVAIL | 0x0200      | ro   | Non-periodic Tx FIFO space available<br>Indicates the amount of space available in the non-periodic Tx FIFO. Values are in terms of 32-bit words.<br>00: Non-periodic transmit FIFO is full<br>01: 1 location available<br>02: 2 locations available<br>N: n locations available ( $0 \leq n \leq 256$ )<br>Others: Reserved<br>Reset value: Configurable  |

### OTGFS general controller configuration register (OTGFS\_GCCFG)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:22 | Reserved | 0x000       | resd | Kept at its default value.  |
| Bit 21    | VBUSIG   | 0x0         | rw   | VBUS ignored<br>When this bit is set, the OTGFS controller does not monitor the Vbus pin voltage, and assumes that the Vbus is always active in both host and device modes, and leaves the Vbus pin for other purposes.<br>0: Vbus is not ignored<br>1: Vbus is ignored, and is deemed as always active |
| Bit 20    | SOFOUTEN | 0x0         | rw   | SOF output enable<br>0: No SOF pulse output<br>1: SOF pulse output on PIN   |
| Bit 19:18 | Reserved | 0x0         | resd | Kept at its default value.  |

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 17   | LP_MODE  | 0x0         | rw   | Low-power mode<br>This bit is used to control the OTG PHY consumption. When this bit is set to 1 by software, the OTG PHY enters low-power mode; when this bit is cleared by software, the OTG PHY operates in normal mode.<br>0: Non-low-power mode<br>1: Low-power mode |
| Bit 16   | PWRDOWN  | 0x0         | rw   | Power down<br>This bit is used to activate the transceiver in transmission/reception. It must be pre-configured to allow USB communication.<br>0: Power down enable<br>1: Power down disable (Transceiver active)   |
| Bit 15:0 | Reserved | 0x0000      | resd | Kept at its default value.  |

#### OTGFS controller ID register (OTGFS\_GUID)

This is a read-only register containing the production ID.

| Bit  | Name   | Reset value | Type | Description   |
|------|--------|-------------|------|---|
| 31:0 | USERID | 0x0000 1000 | rw   | Product ID field<br>The application can program the ID field. |

#### OTGFS host periodic Tx FIFO size register (OTGFS\_HPTXFSIZ)

This register contains the size and memory start address of the periodic transmit FIFO.

| Bit       | Name       | Reset value | Type  | Description  |
|-----------|------------|-------------|-------|--|
| Bit 31:16 | PTXFSIZE   | 0x02000     | ro/rw | Host periodic TxFIFO depth<br>Values are in terms of 32-bit words.<br>Minimum value is 16<br>Maximum value is 512  |
| Bit 15:0  | PTXFSTADDR | 0x0600      | ro/rw | Host Periodic TxFIFO start address<br>The power-on reset value of this register is the sum of the largest receive FIFO depth and the largest non-periodic transmit FIFO depth. |

#### OTGFS device IN endpoint Tx FIFO size register (OTGFS\_DIEPTXFn) (x = 1...15, where n is the FIFO number)

This register holds the depth and memory start address of the IN endpoint transmit FIFO in device mode. Each of the FIFOs contains an IN endpoint data. This register can be used repeatedly for instantiated IN endpoint FIFO1~15. The GNPTXFSIZ register is used to program the depth and memory start address of the IN endpoint FIFO 0.

| Bit       | Name          | Reset value | Type  | Description  |
|-----------|---------------|-------------|-------|--|
| Bit 31:16 | INEPTXFDEP    | 0x0200      | ro/rw | IN Endpoint TxFIFO depth<br>Values are in terms of 32-bit words.<br>Minimum value is 16<br>Maximum value is 512<br>The reset value is the maximum possible IN endpoint transmit FIFO depth |
| Bit 15:0  | INEPTXFSTADDR | 0x0400      | ro/rw | IN Endpoint FIFO n transmit SRAM start address<br>This field contains the SRAM start address of the IN endpoint n transmit FIFO  |



## Host-mode registers

Host-mode registers affect the operation of the controller in host mode. Host-mode registers are not accessible in device mode (as the results are undefined in device mode). Host-mode registers contain as follows:

### OTGFS host mode configuration register (OTGFS\_HCFG)

This register is used to configure the controller after power-on. Do not change this register after initialization.

| Bit      | Name        | Reset value | Type | Description  |
|----------|-------------|-------------|------|--|
| Bit 31:3 | Reserved    | 0x0000 0000 | resd | Kept at its default value.   |
| Bit 2    | FSLSSUPP    | 0x0         | ro   | FS- and LS-only support<br>The application uses this bit to control the controller's enumeration speed. With this bit, the application can make the controller enumerate as a full-speed host mode, even if the connected device supports high-speed communication. Do not change this bit after initial programming.<br>0: FS/LS, depending on the largest speed supported by the connected device.<br>1: FS/LS-only, even if the connected device supports high-speed. |
| Bit 1:0  | FSLSPCLKSEL | 0x0         | rw   | FS/LS PHY clock select<br>When the controller is in FS host mode:<br>01: PHY clock is running at 48 MHz<br>Others: Reserved<br>When the controller is in LS host mode:<br>00: Reserved<br>01: PHY clock is running at 48 MHz<br>10: PHY clock is running at 6 MHz. If 6 MHz clock is selected, reset must be done by software.<br>11: Reserved   |

### OTGFS host frame interval register (OTGFS\_HFIR)

This register is used to program the current

| Bit       | Name        | Reset value | Type | Description  |
|-----------|-------------|-------------|------|--|
| Bit 31:17 | Reserved    | 0x0000      | resd | Kept at its default value.   |
| Bit 16    | HFIRRLDCTRL | 0x0         | rw   | Reload control<br>This bit is used to disable/enable dynamic reload for the host frame register at runtime.<br>1: Reload control disable<br>0: Reload control enable<br>This bit must be configured at initialization. Do not change its value at runtime. |

| Bit      | Name  | Reset value | Type | Description   |
|----------|-------|-------------|------|---|
| Bit 15:0 | FRINT | 0xEA60      | rw   | <p>Frame interval</p> <p>The application uses this field to program the interval between two consecutive SOFs (full speed)</p> <p>The number of PHY locks in this field indicates the frame interval. The application can write a value to the host frame interval register only after the port enable bit in the host port control and status register has been set.</p> <p>If no value is programmed, the controller calculates the value based on the PHY clock frequency defined in the FS/LS PHY clock select bit of the host configuration register. Do not change the value of this field after initial configuration.</p> <p>1 ms × (FS/LS PHY clock frequency)</p> |

#### OTGFS host frame number/frame time remaining register (OTGFS\_HFNUM)

This register indicates the current frame number, and also the time remaining in the current frame (in terms of the number of PHY clocks).

| Bit       | Name  | Reset value | Type | Description   |
|-----------|-------|-------------|------|---|
| Bit 31:16 | FTREM | 0x0000      | ro   | <p>Frame time remaining</p> <p>Indicates the time remaining in the current frame (FS/HS), in terms of the number of PHY clocks. This field decrements with the number of PHY clocks. When it reaches zero, this field is reloaded with the value of the frame interval register, and a new SOF is transmitted on the USB bus.</p> |
| Bit 15:0  | FRNUM | 0x3FFF      | ro   | <p>Frame number</p> <p>This field increments every time a new SOP is transmitted on the USB bus, and is cleared to 0 when the value reaches 16'h3FFF.</p>   |

#### OTGFS host periodic Tx FIFO/request queue register (OTGFS\_HPTXSTS)

This is a read-only register containing the free space information of the periodic Tx FIFO and the periodic transmit request queue.

| Bit       | Name    | Reset value | Type | Description   |
|-----------|---------|-------------|------|---|
| Bit 31:24 | PTXQTOP | 0x00        | ro   | <p>Top of the periodic transmit request queue)</p> <p>Indicates that the MAC is processing the request from the periodic transmit request queue. This register is used for debugging.</p> <p>Bit [31]: Odd/Even frame</p> <p>0: Transmit in even frame</p> <p>1: Transmit in odd frame</p> <p>Bit [30:27]: Channel/Endpoint number</p> <p>Bit [26:25]: Type</p> <p>00: IN/OUT</p> <p>01: Zero-length packet</p> <p>10: Reserved</p> <p>11: Channel command disable</p> <p>Bit [24]: Terminate (last request for the selected channel or endpoint)</p> |

| Bit       | Name         | Reset value | Type | Description  |
|-----------|--------------|-------------|------|--|
| Bit 23:16 | PTXQSPCAVAIL | 0x08        | ro   | Periodic transmit request queue space available<br>Indicates the number of free space available to be written in the periodic transmit request queue. This queue contains both IN and OUT requests.<br>00: Periodic transmit request queue is full<br>01: 1 space available<br>10: 2 space available<br>N: n space available ( $0 \leq n \leq 8$ )<br>Others: Reserved |
| Bit 15:0  | PTXFSPCAVAIL | 0x0100      | rw   | Periodic transmit data FIFO space available<br>Indicates the number of free space available to be written in the periodic transmit FIFO, in terms of 32-bit words.<br>0000: Periodic transmit FIFO is full<br>0001: 1 space available<br>0010: 2 space available<br>N: n space available ( $0 \leq n \leq 512$ )<br>Others: Reserved                                   |

#### OTGFS host all channels interrupt register (OTGFS\_HAINT)

When a flag event occurs on a channel, the host all channels interrupt register interrupts the application through the host channels interrupt bit of the controller interrupt register, as shown in Figure 240. There is one interrupt bit for each channel, up to 16 bits. The application sets or clears this register by setting or clearing the appropriate bit in the corresponding host channel-n interrupt register.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 15:0  | HAINT    | 0x0000      | ro   | Channel interrupts<br>One bit per channel: bit 0 for channel 0, bit 15 for channel 15. |

#### OTGFS host all channels interrupt mask register (OTGFS\_HAINTMSK)

The host all channels interrupt mask register works with the host all channels interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per one channel, 16 bits in total.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Kept at its default value.   |
| Bit 15:0  | HAINTMSK | 0x0000      | rw   | Channel interrupt mask<br>One bit per channel: bit 0 for channel 0, bit 15 for channel 15. |

## OTGFS host port control and status register (OTGFS\_HPRT)

This register is valid only in host mode. Currently, the OTG host supports only one port.

This register contains USB port-related information such as USB reset, enable, suspend, resume, connect status and test mode, as show in Figure 240. The register of type rw1c can interrupt the application through the host port interrupt bit in the controller interrupt register. Upon a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the register of type rw1c, the application must write 1 to clear the interrupt.

| Bit       | Name      | Reset value | Type | Description  |
|-----------|-----------|-------------|------|--|
| Bit 31:19 | Reserved  | 0x0000      | resd | Kept at its default value.   |
| Bit 18:17 | PRTSPD    | 0x0         | ro   | Port speed<br>Indicates the speed of the device connected to this port.<br>00: Reserved<br>01: Full speed<br>10: Low speed<br>11: Reserved   |
| Bit 16:13 | PRTTSTCTL | 0x0         | rw   | Port test control<br>The application writes a non-zero value to this field to put the port into test mode, and the port gives a corresponding signal.<br>0000: Test mode disabled<br>0001: Test_J mode<br>0010: Test_K mode<br>0011: Test_SE0_NAK mode<br>0100: Test_Packet mode<br>0101: Test_Force_Enable<br>Others: Reserved  |
| Bit 12    | PRTPWR    | 0x0         | rw   | Port power<br>The application uses this bit to control power supply to this port (by writing 1 or 0)<br>0: Power off<br>1: Power on<br><br>Note: This bit is not associated with interfaces. The application must follow the programming manual to set this bit for various interfaces.  |
| Bit 11:10 | PRTLNSTS  | 0x0         | ro   | Port line status<br>Indicates the current logic status of the USB data lines.<br>Bit [10]: Logic level of D+<br>Bit [11]: Logic level of D-  |
| Bit 9     | Reserved  | 0x0         | resd | Kept at its default value.   |
| Bit 8     | PRTRST    | 0x0         | rw   | Port reset<br>When this bit is set by the application, a reset sequence is started on this port. The application must calculate the time required for the reset sequence, and clear this bit after the reset sequence is complete.<br>0: Port not in reset<br>1: Port in reset<br><br>The application must keep this bit set for a minimum duration defined in Section 7.1.7.5 of USB 2.0 specification to start a reset on the port. In addition to this, the application can make this bit set for another 10 ms to the minimum duration, before clearing this bit. There is no maximum limit set by the USB standard. |

| Bit   | Name       | Reset value | Type | Description  |
|-------|------------|-------------|------|--|
| Bit 7 | PRTSUSP    | 0x0         | rw1s | <p>Port suspend</p> <p>The application sets this bit to put this port in suspend mode. In this case, the controller only stops sending SOF. The application must set the port clock stop bit in order to disable the PHY clock.</p> <p>The read value of this bit reflects the current suspend status of the port.</p> <p>This bit is cleared by the controller when a remote wakeup signal is detected or when the application sets the port reset bit or port resume bit in this register, or sets the resume/remote wakeup detected interrupt bit or disconnect detected interrupt bit in the controller interrupt register.</p> <p>The controller can still clear this bit, even if the device is disconnected with the host.</p> <p>0: Port not in suspend mode<br/>1: Port in suspend mode</p> |
| Bit 6 | PRTRES     | 0x0         | rw   | <p>Port resume</p> <p>The application sets this bit to drive resume signaling on the port. The controller continues to trigger the resume signal until the application clears this bit. If the controller detects a USB remote wakeup sequence (as indicated by the port resume/remote wakeup detected interrupt bit of the controller interrupt register), the controller starts driving resume signaling without the intervention of the application.</p> <p>The read value of this bit indicates whether the controller is currently driving resume signaling.</p> <p>0: No resume triggered<br/>1: Resume triggered</p>  |
| Bit 5 | PRTOVRCHNG | 0x0         | rw1c | <p>Port overcurrent change</p> <p>The controller sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes. This bit can only be set by the controller. The application must write 1 to clear this bit.</p>  |
| Bit 4 | PRTOVRCACT | 0x0         | ro   | <p>Port overcurrent active</p> <p>Indicates the overcurrent status of the port.</p> <p>0: No overcurrent<br/>1: Overcurrent condition</p>  |
| Bit 3 | PRTENCHNG  | 0x0         | rw1c | <p>Port enable/disable change</p> <p>The controller sets this bit when the status of the port enable bit 2 in this register changes. This bit can only be set by the controller. The application must write 1 to clear this bit.</p>   |
| Bit 2 | PRTENA     | 0x0         | rw1c | <p>Port enable</p> <p>A port is enabled only by the controller after a reset sequence. This port is enabled by an overcurrent condition, a disconnected condition or by the application. The application cannot set this bit by a register write operation. It can only clear this bit to disable the port. This bit does not trigger any interrupt.</p> <p>0: Port disabled<br/>1: Port enabled</p>   |

| Bit   | Name      | Reset value | Type | Description  |
|-------|-----------|-------------|------|--|
| Bit 1 | PRTCONDET | 0x0         | rw1c | Port connect detected<br>On a device connection detected, the controller sets this bit using the host port interrupt bit in the controller register. This bit can only be set by the controller. The application must write 1 to clear this bit. |
| Bit 0 | PRTCONSTS | 0x0         | ro   | Port connect status<br>0: No device is connected to the port<br>1: A device is connected to the port   |

**OTGFS host channelx characteristics register (OTGFS\_HCCHARx) (x = 0...15, where x = channel number)**

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31    | CHENA    | 0x0         | rw1s | Channel enable<br>This bit is set by the application and cleared by the OTG host.<br>0: Channel disabled<br>1: Channel enabled   |
| Bit 30    | CHDIS    | 0x0         | rw1s | Channel disable<br>The application sets this bit to stop transmitting or receiving data on a channel, even before the transfer on that channel is complete. The application must wait for the generation of the channel disabled interrupt before treating the channel as disabled.  |
| Bit 29    | ODDFRM   | 0x0         | rw   | Odd frame<br>This bit is set / cleared by the application to indicate that the OTG host must perform a transfer in an odd frame. This bit is applicable for periodic transfers (synchronous and interrupt) only.<br>0: Even frame<br>1: Odd frame  |
| Bit 28:22 | DEVADDR  | 0x00        | rw   | Device address<br>This field is used to select the device that can serve as the data source or receiver.   |
| Bit 21:20 | MC       | 0x0         | rw   | Multi count (MC)<br>This field indicates to the host the number of transfers that must be performed per frame for the periodic endpoint.<br>00: Reserved. This field generates undefined results.<br>01: 1 transaction<br>10: 2 transactions per frame<br>11: 3 transactions per frame<br>This field must be set to at least 0x01. |
| Bit 19:18 | EPTYPE   | 0x0         | rw   | Endpoint type<br>Indicates the transfer type selected.<br>00: Control transfer<br>01: Synchronous transfer<br>10: Bulk transfer<br>11: Interrupt transfer  |
| Bit 17    | LSPDDEV  | 0x0         | rw   | Low-speed device<br>The application sets this bit to indicate that this channel is communicating to a low-speed device.  |
| Bit 16    | Reserved | 0x0         | resd | Kept at its default value.   |

| Bit       | Name   | Reset value | Type | Description   |
|-----------|--------|-------------|------|---|
| Bit 15    | EPTDIR | 0x0         | rw   | Endpoint direction<br>Indicates whether the transfer is in IN or OUT.<br>0: OUT<br>1: IN            |
| Bit 14:11 | EPTNUM | 0x0         | rw   | Endpoint number<br>Indicates the endpoint number on the device (serving as data source or receiver) |
| Bit 10:0  | MPS    | 0x000       | rw   | Maximum packet size<br>Indicates the maximum packet size of the corresponding port.                 |

#### OTGFS host channelx interrupt register (OTGFS\_HCINTx) (x = 0...15, where x = channel number)

This register contains the status of a channel related to USB and AHB events, as shown in Figure 240. The application must read this register when the host channels interrupt bit is set in the controller interrupt register. Before reading this register, the application must read the host all channels interrupt register to get the exact channel number of the host channel-n interrupt register. The application must clear the corresponding bit in this register to clear the corresponding bits in the OTGFS\_HAIN and OTGFS\_GINTSTS registers.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:11 | Reserved | 0x000000    | resd | Kept at its default value.  |
| Bit 10    | DTGLERR  | 0x0         | rw1c | Data toggle error<br>This bit can only be set by the controller. The application must write 1 to clear this bit.  |
| Bit 9     | FRMOVRUN | 0x0         | rw1c | Frame overrun<br>This bit can only be set by the controller. The application must write 1 to clear this bit.  |
| Bit 8     | BBLERR   | 0x0         | rw1c | Babble error<br>This bit can only be set by the controller. The application must write 1 to clear this bit.   |
| Bit 7     | XACTERR  | 0x0         | rw1c | Transaction error<br>Indicates one of the following errors occurred on the USB bus:<br>CRC check failure<br>Timeout<br>Bit stuffing error<br>EOP error<br>This bit can only be set by the controller. The application must write 1 to clear this bit. |
| Bit 6     | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 5     | ACK      | 0x0         | rw1c | ACK response received/Transmitted interrupt<br>This bit can only be set by the controller. The application must write 1 to clear this bit.  |
| Bit 4     | NAK      | 0x0         | rw1c | NAK response received interrupt<br>This bit can only be set by the controller. The application must write 1 to clear this bit.  |
| Bit 3     | STALL    | 0x0         | rw1c | STALL response received interrupt<br>This bit can only be set by the controller. The application must write 1 to clear this bit.  |
| Bit 2     | Reserved | 0x0         | resd | Kept at its default value.  |

| Bit   | Name   | Reset value | Type | Description   |
|-------|--------|-------------|------|---|
| Bit 1 | CHHLTD | 0x0         | rw1c | Channel hated<br>Indicates that the transfer completed abnormally either because of any transfer error or in response to a disable request by the application.    |
| Bit 0 | XFERC  | 0x0         | rw1c | Transfer completed<br>Transfer completed normally, without any error. This bit can only be set by the controller. The application must write 1 to clear this bit. |

#### OTGFS host channelx interrupt mask register (OTGFS\_HCINTMSKx) (x = 0...15, where x = channel number)

This register is used to mask the channels described in the previous section.

| Bit       | Name        | Reset value | Type | Description                                      |
|-----------|-------------|-------------|------|--|
| Bit 31:11 | Reserved    | 0x000000    | resd | Kept at its default value.                       |
| Bit 10    | DTGLERRMSK  | 0x0         | rw   | Data toggle error mask                           |
| Bit 9     | FRMOVRUNMSK | 0x0         | rw   | Frame overrun mask                               |
| Bit 8     | BBLERRMSK   | 0x0         | rw   | Babble error mask                                |
| Bit 7     | XACTERRMSK  | 0x0         | rw   | Transaction error mask                           |
| Bit 6     | NYETMSK     | 0x0         | rw   | NYET response received interrupt mask            |
| Bit 5     | ACKMSK      | 0x0         | rw   | ACK response received/transmitted interrupt mask |
| Bit 4     | NAKMSK      | 0x0         | rw   | NAK response received interrupt mask             |
| Bit 3     | STALLMSK    | 0x0         | rw   | STALL response received interrupt mask           |
| Bit 2     | Reserved    | 0x0         | resd | Kept at its default value.                       |
| Bit 1     | CHHLTDMASK  | 0x0         | rw   | Channel halted mask                              |
| Bit 0     | XFERCMSK    | 0x0         | rw   | Transfer completed mask                          |

#### OTGFS host channelx transfer size register (OTGFS\_HCTSIZx) (x = 0...15, where x = channel number)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31    | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 30:29 | PID      | 0x0         | rw   | PID (Pid)<br>The application programs this field with the type of PID used for the initial transfer. The host controls this field for the rest of transfers.<br>00: DATA0<br>01: DATA2<br>10: DATA1<br>11: MDATA(non-control) / SETUP(control)   |
| Bit 28:19 | PKTCNT   | 0x000       | rw   | Packet count<br>The application programs this field with the expected number of packets to be transmitted or received. The host decrements the packet count on every successful transmission or reception of an OUT/IN packet. When this count reaches zero, the application is interrupted to indicate normal completion of the transfer. |



| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 18:0 | XFERSIZE | 0x00000     | rw   | Transfer size<br>For an OUT transfer, this field indicates the number of data bytes the host sends during a transfer.<br>For an IN transfer, this field indicates the buffer size that the application has reserved for the transfer.<br>For an IN transfer (periodic and non-periodic), the application must program this field as an integer multiple of the maximum packet size. |

## Device-mode registers

These registers are applicable in device mode only. They are not supported in host mode due to unknown access results. Some of the registers affect all the endpoints, while some affect only one endpoint.

### OTGFS device configure register (OTGFS\_DCFG)

This register configures the controller in device mode after power-on or after certain control commands or enumeration. Do not change this register after initial programming.

| Bit       | Name         | Reset value | Type | Description  |
|-----------|--------------|-------------|------|--|
| Bit 31:13 | Reserved     | 0x04100     | resd | Kept at its default value.   |
| Bit 12:11 | PERFRINT     | 0x0         | rw   | Periodic frame interval<br>This field indicates the time within a frame at which the periodic frame end interrupt is generated. The application can use this interrupt to determine if the synchronous transfer has been completed in a frame.<br>00: 80 % of the frame interval<br>01: 85 % of the frame interval<br>10: 90 % of the frame interval<br>11: 95 % of the frame interval   |
| Bit 10:4  | DEVADDR      | 0x00        | rw   | Device address<br>The application must program this field every time a SetAddress command is received.   |
| Bit 3     | Reserved     | 0x0         | resd | Kept at its default value.   |
| Bit 2     | NZSTSOUTHSHK | 0x0         | rw   | Non-zero-length status OUT handshake<br>The application can use this field to select the handshake the controller sends on receiving a non-zero-length data packet during a control transfer' status stage.<br>1: Send a STALL handshake on a non-zero-length status OUT transfer and do not send the received OUT packet to the application<br>0: Send the received OUT packet to the application (zero-length or non-zero-length), and send a handshake based on the NAK and STALL bits in the device endpoint control register. |

| Bit     | Name   | Reset value | Type | Description  |
|---------|--------|-------------|------|--|
| Bit 1:0 | DEVSPD | 0x0         | rw   | <p>Device speed</p> <p>This field indicates the speed at which the application needs the controller to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the entire sequence is complete, and is based on the speed of the USB host to which the controller is connected.</p> <p>00: Reserved<br/>01: Reserved<br/>10: Reserved<br/>11: Full speed (USB1.1 transceiver, clock is 48 MHz)</p> |

#### OTGFS device control register (OTGFS\_DCTL)

| Bit       | Name       | Reset value | Type | Description   |
|-----------|------------|-------------|------|---|
| Bit 31:12 | Reserved   | 0x00000     | resd | Kept at its default value.  |
| Bit 11    | PWROPRGDNE | 0x0         | wo   | <p>Power-on programming done</p> <p>The application uses this bit to indicate that the register configuration is complete after a wakeup from power-down mode.</p>  |
| Bit 10    | CGOUTNAK   | 0x0         | wo   | <p>Clear global OUT NAK</p> <p>Writing 1 to this bit clears the global OUT NAK.</p>   |
| Bit 9     | SGOUTNAK   | 0x0         | wo   | <p>Set global OUT NAK</p> <p>Writing to this bit sets the global OUT NAK.</p> <p>The application uses this bit to send a NAK handshake on all OUT endpoints. The application must set this bit only after checking that the global OUT NAK effective bit in the controller interrupt register is cleared.</p>                                     |
| Bit 8     | CGNPINNAK  | 0x0         | wo   | <p>Clear Global Non-periodic IN NAK</p> <p>Writing to this bit clears the global Non-periodic OUT NAK.</p>  |
| Bit 7     | SGNPINNAK  | 0x0         | wo   | <p>Set global Non-periodic IN NAK</p> <p>Writing to this bit sets the global Non-periodic OUT NAK.</p> <p>The application uses this bit to send a NAK handshake on all non-periodic IN endpoints. The application must set this bit only after checking that the global IN NAK effective bit in the controller interrupt register is cleared.</p> |
| Bit 6:4   | TSTCTL     | 0x0         | rw   | <p>Test control</p> <p>000: Test mode disabled<br/>001: Test_J mode<br/>010: Test_K mode<br/>011: Test_SE0_NAK mode<br/>100: Test_Packet mode<br/>101: Test_Force_Enable<br/>Others: Reserved</p>   |
| Bit 3     | GOUTNAKSTS | 0x0         | ro   | <p>Global OUT NAK status</p> <p>0: A handshake is sent based on the FIFO status, NAK and STALL bit settings.<br/>1: No data is written to the receive FIFO, irrespective of space availability. Sends a NAK handshake on all packets (except on SETUP transfers). Drops all synchronous OUT packets.</p>  |

| Bit   | Name        | Reset value | Type | Description  |
|-------|-------------|-------------|------|--|
| Bit 2 | GNPINNAKSTS | 0x0         | ro   | Global Non-periodic IN NAK status<br>0: A handshake is sent based on the data status in the transmit FIFO<br>1: A NAK handshake is sent on all non-periodic IN endpoints, irrespective of the data status in the transmit FIFO.  |
| Bit 1 | SFTDISCON   | 0x1         | rw   | Software disconnect<br>The application uses this bit to indicate the OTGFS controller to perform software disconnected. Once this bit is set, the host finds the device disconnected, and the device does not receive signals on the USB bus. The controller stays in the disconnected state until the application clears this bit.<br>0: Normal operation. When this bit is cleared after a software disconnect, the controller issues a device connect event to the host. Then the USB host restarts device enumeration. |
| Bit 0 | RWKUPSIG    | 0x0         | rw   | Remote wakeup signaling<br>When this bit is set by the application, the controller initiates a remote signal to wakeup the USB host. The application must set this bit to indicate the controller to exit the suspend mode. Per USB2.0 standards, the application must clear this bit 1-15 ms after setting it.  |

Table 79 lists the minimum duration at which the software disconnect bit must be set in various states for the USB host to detect a device disconnect. To accommodate clock jitter, it is advised that the application adds some extra delay to the specified minimum duration.

**Table 79. Minimum duration for software disconnect**

| Operating speed | Device state                              | Minimum duration   |
|-----------------|---|--------------------|
| Full speed      | Suspend                                   | 1 ms + 2.5 $\mu$ s |
| Full speed      | Idle                                      | 2.5 $\mu$ s        |
| Full speed      | No idle or suspend (performing transfers) | 2.5 $\mu$ s        |

### OTGFS device status register (OTGFS\_DSTS)

This register indicates the status of the controller related to OTGFS events. It must be read on interrupt events from the device all interrupts register (OTGFS\_DAINR).

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:22 | Reserved | 0x000       | resd | Kept at its default value.  |
| Bit 21:8  | SOFFN    | 0x0000      | ro   | Frame number of the received SOF<br>Note: The read value of this field immediately after power-on reset reflects a non-zero value. If a non-zero value is returned after reading this field immediately after power-on reset, it does not mean that the host has received a SOP. The read value of this field is valid only when the host is connected to the device. |
| Bit 7:4   | Reserved | 0x1         | resd | Kept at its default value.  |

| Bit     | Name    | Reset value | Type | Description  |
|---------|---------|-------------|------|--|
| Bit 3   | ETICERR | 0x0         | ro   | Erratic error<br>This error causes the controller to enter suspend mode, and interrupt is generated with the early suspend bit of the controller interrupt register. If the early suspend is asserted due to an erratic error, the application can only perform a software disconnect recover.   |
| Bit 2:1 | ENUMSPD | 0x0         | ro   | Enumerated speed<br>Indicates the speed at which the controller has determined after speed detection through a sequence.<br>01: Reserved<br>10: Reserved<br>11: Full speed (PHY clock is running at 48 MHz)<br>Others: Reserved  |
| Bit 0   | SUSPSTS | 0x0         | ro   | Suspend status<br>In device mode, this bit is set as long as a suspend condition is detected on the USB bus. The controller enters the suspend state when there is no activity on the USB bus.<br>The controller exits the suspend state on the following conditions:<br>When there is an activity on the USB bus<br>When the application writes to the remote wakeup signal bit in the device control register. |

#### OTGFS device OTGFSIN endpoint common interrupt mask register (OTGFS\_DIEPMSK)

This register works with each of the device IN endpoint interrupt register for all endpoints to generate an IN endpoint interrupt. The IN endpoint interrupt for a specific status in the OTGFS\_DIEPINTx register can be masked by writing to the corresponding bit in the OTGFS\_DIEPMSK register. Status bits are masked by default.

| Bit       | Name           | Reset value | Type | Description   |
|-----------|----------------|-------------|------|---|
| Bit 31:10 | Reserved       | 0x000000    | resd | Kept at its default value.  |
| Bit 9     | BNAINMSK       | 0x0         | rw   | BNA interrupt mask<br>0: Interrupt masked<br>1: Interrupt unmasked                                  |
| Bit 8     | TXFIFOUDRMSK   | 0x0         | rw   | FIFO underrun mask<br>0: Interrupt masked<br>1: Interrupt unmasked                                  |
| Bit 7     | Reserved       | 0x0         | resd | Kept at its default value.  |
| Bit 6     | INEPTNAKMSK    | 0x0         | rw   | IN endpoint NAK effective mask<br>0: Interrupt masked<br>1: Interrupt unmasked                      |
| Bit 5     | INTKNEPTMISMSK | 0x0         | rw   | IN token received with EP mismatch mask<br>0: Interrupt masked<br>1: Interrupt unmasked             |
| Bit 4     | INTKNTXFEMPMSK | 0x0         | rw   | IN token received when Tx FIFO empty mask<br>0: Interrupt masked<br>1: Interrupt unmasked           |
| Bit 3     | TIMEOUTMSK     | 0x0         | rw   | Timeout condition mask (Non-isochronous endpoints))<br>0: Interrupt masked<br>1: Interrupt unmasked |
| Bit 2     | Reserved       | 0x0         | resd | Kept at its default value.  |

| Bit   | Name      | Reset value | Type | Description   |
|-------|-----------|-------------|------|---|
| Bit 1 | EPTDISMSK | 0x0         | rw   | Endpoint disabled interrupt mask<br>0: Interrupt masked<br>1: Interrupt unmasked  |
| Bit 0 | XFERCMSK  | 0x0         | rw   | Transfer completed interrupt mask<br>0: Interrupt masked<br>1: Interrupt unmasked |

#### OTGFS device OUT endpoint common interrupt mask register (OTGFS\_DOEPMSK)

This register works with each of the OTGFS\_DOEPINTx registers for all endpoints to generate an OUT endpoint interrupt. Each of the bits in the OTGFS\_DOEPINTx registers can be masked by writing to the register. All interrupts are masked by default.

| Bit       | Name        | Reset value | Type | Description   |
|-----------|-------------|-------------|------|---|
| Bit 31:10 | Reserved    | 0x000000    | resd | Kept at its default value.  |
| Bit 9     | BNAOUTMSK   | 0x0         | rw   | BNA interrupt mask<br>0: Interrupt masked<br>1: Interrupt unmasked  |
| Bit 8     | OUTPERRMSK  | 0x0         | rw   | OUT packet error mask<br>0: Interrupt masked<br>1: Interrupt unmasked                                       |
| Bit 7     | Reserved    | 0x0         | resd | Kept at its default value.  |
| Bit 6     | B2BSETUPMSK | 0x0         | rw   | Back-to-back SETUP packets received mask<br>0: Interrupt masked<br>1: Interrupt unmasked                    |
| Bit 5     | Reserved    | 0x0         | resd | Kept at its default value.  |
| Bit 4     | OUTTEPDMSK  | 0x0         | rw   | OUT token received when endpoint disabled mask<br>0: Interrupt masked<br>1: Interrupt unmasked              |
| Bit 3     | SETUPMSK    | 0x0         | rw   | SETUP phase done mask<br>Applies to control endpoints only.<br>0: Interrupt masked<br>1: Interrupt unmasked |
| Bit 2     | Reserved    | 0x0         | resd | Kept at its default value.  |
| Bit 1     | EPTDISMSK   | 0x0         | rw   | Endpoint disabled interrupt mask<br>0: Interrupt masked<br>1: Interrupt unmasked                            |
| Bit 0     | XFERCMSK    | 0x0         | rw   | Transfer completed interrupt mask<br>0: Interrupt masked<br>1: Interrupt unmasked                           |

### OTGFS device all endpoints interrupt mask register (OTGFS\_DAIN)

When an event occurs on an endpoint, The IN/OUT endpoint interrupt bits in the OTGS\_DAIN register can be used to interrupt the application. There is one interrupt bit per endpoint, up to 8 interrupt bits for OUT endpoints and 8 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used at the same time. The corresponding bits in this register are set and cleared when the application sets and clears the bits in the corresponding device endpoint-x interrupt register.

| Bit       | Name      | Reset value | Type | Description  |
|-----------|-----------|-------------|------|--|
| Bit 31:24 | Reserved  | 0x0000      | resd | Kept at its default value.   |
| Bit 23:16 | OUTEPTINT | 0x0000      | ro   | OUT endpoint interrupt bits<br>One OUT endpoint per bit. Bit 16 for OUT endpoint 0, bit 18 for OUT endpoint 2. |
| Bit 15:8  | Reserved  | 0x0000      | resd | Kept at its default value.   |
| Bit 7:0   | INEPTINT  | 0x0000      | ro   | IN endpoint interrupt bits<br>One IN endpoint per bit. Bit 0 for IN endpoint 0, bit 7 for IN endpoint 7.       |

### OTGFS all endpoints interrupt mask register (OTGFS\_DAINMSK)

When an event occurs on a device endpoint, the device endpoint interrupt mask register works with the device endpoint interrupt register to interrupt the application. However, the device all endpoints interrupt register corresponding to this interrupt is still set.

| Bit       | Name      | Reset value | Type | Description   |
|-----------|-----------|-------------|------|---|
| Bit 31:24 | Reserved  | 0x0000      | resd | Kept at its default value.  |
| Bit 23:16 | OUTEPTMSK | 0x0000      | rw   | OUT EP interrupt mask bits<br>One OUT endpoint per bit. Bit 16 for OUT endpoint 0, bit 18 for OUT endpoint 2.<br>0: Interrupt masked<br>1: Interrupt unmasked |
| Bit 15:8  | Reserved  | 0x0000      | resd | Kept at its default value.  |
| Bit 7:0   | INEPTMSK  | 0x0000      | rw   | IN EP interrupt mask bits<br>One IN endpoint per bit. Bit 0 for IN endpoint 0, bit 7 for IN endpoint 7.<br>0: Interrupt masked<br>1: Interrupt unmasked       |

### OTGFS device IN endpoint FIFO empty interrupt mask register (OTGFS\_DIEPEMPMSK)

This register works with the TXFE\_OTGFS\_DIEPINTx register to generate an interrupt.

| Bit      | Name        | Reset value | Type | Description   |
|----------|-------------|-------------|------|---|
| Bit 31:8 | Reserved    | 0x0000      | resd | Kept at its default value.  |
| Bit 7:0  | INEPTXFEMSK | 0x0000      | rw   | IN endpoint Tx FIFO empty interrupt mask bits<br>These bits serve as mask bits for the device IN endpoint interrupt register.<br>A transmit FIFO empty interrupt bit per IN endpoint. Bit 0 for IN endpoint 0, bit 7 for IN endpoint 7.<br>0: Interrupt masked<br>1: Interrupt unmasked |

### OTGFS device control IN endpoint 0 control register (OTGFS\_DIEPCTL0)

This section describes the control IN endpoint 0 control register. Nonzero control endpoint uses registers for endpoints 1-7.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31    | EPTENA   | 0x0         | rw1s | Endpoint enable<br>The application sets this bit to start data transmission on the endpoint 0.<br>The controller clears this bit before generating the following interrupts:<br>Endpoint disabled<br>Transfer completed.  |
| Bit 30    | EPTDIS   | 0x0         | ro   | Endpoint disable<br>The application sets this bit to stop data transmission on an endpoint. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The controller clears this bit before setting the endpoint disabled interrupt. The application must set this bit only when the endpoint is enabled.   |
| Bit 29:28 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 27    | SNAK     | 0x0         | wo   | Set NAK<br>A write to this bit sets the NAK bit of the endpoint. The application can use this bit to control the transmission of NAK handshakes on the endpoint. The controller also sets this bit when a SETUP data packet is received on the endpoint.  |
| Bit 26    | CNAK     | 0x0         | wo   | Clear NAK<br>A write to this bit clears the NAK bit for the endpoint.   |
| Bit 25:22 | TXFNUM   | 0x0         | rw   | TxFIFO number<br>The endpoint 0 can only use FIFO0.   |
| Bit 21    | STALL    | 0x0         | rw1s | STALL handshake<br>The application sets this bit, and the controller clears this bit when a SETUP token is received. If a NAK bit, a global non-periodic IN NAK or global OUT NAK bit is set along with this bit, the STALL bit has priority.   |
| Bit 20    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 19:18 | EPTYPE   | 0x0         | ro   | Endpoint type<br>Set to 0 by hardware for control endpoints.  |
| Bit 17    | NAKSTS   | 0x0         | ro   | NAK status<br>Indicates the following:<br>0: The controller is transmitting non-NAK handshakes based on the FIFO status<br>1: The controller is transmitting NAK handshakes on this endpoint<br>When this bit is set, either by the application or controller, the controller stops transmitting data, even if there are space available in the receive FIFO. The controller always responds to SETUP data packets with an ACK handshake, irrespective of this bit's setting. |
| Bit 16    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 15    | USBACEPT | 0x1         | ro   | USB active endpoint<br>This bit is always set to 1, indicating that the control endpoint 0 is always active in all configurations and interfaces.   |
| Bit 14:2  | Reserved | 0x0000      | resd | Kept at its default value.  |

| Bit     | Name | Reset value | Type | Description  |
|---------|------|-------------|------|--|
| Bit 1:0 | MPS  | 0x0         | rw   | Applies to IN and OUT endpoints<br>The application uses this bit to program the maximum packet size for the current logical endpoint.<br>00: 64 bytes<br>01: 32 bytes<br>10: 16 bytes<br>11: 8 bytes |

#### OTGFS device IN endpoint-x control register (OTGFS\_DIEPCTLx) (x = 1...7, where x is endpoint number)

The application uses this register to control the behavior of the endpoints other than endpoint 0.

| Bit    | Name                   | Reset value | Type | Description   |
|--------|------------------------|-------------|------|---|
| Bit 31 | EPTENA                 | 0x0         | rw1s | Endpoint enable<br>The application sets this bit to start transmitting data on an endpoint. The controller clears this bit before the generation one of the following interrupts on this endpoint:<br>SETUP stage done<br>Endpoint disabled<br>Transfer completed   |
| Bit 30 | EPTDIS                 | 0x0         | rw1s | Endpoint disable<br>The application sets this bit to stop transmitting data on an endpoint, even if the transfer on that endpoint is incomplete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The controller clears this bit before setting the endpoint disabled interrupt. The application must set this bit only when the endpoint enabled set. |
| Bit 29 | SETD1PID/<br>SETODDFR  | 0x0         | wo   | Set DATA1 PID<br>Applies to interrupt/bulk IN endpoints only. Writing to this bit sets the endpoint data PID bit in this register to DATA1.<br>Set odd frame<br>Applies to synchronous IN endpoints only. Writing to this bit sets the Even/Odd frame to odd frame.<br>0: Disabled Set DATA1 PID disabled or Do not force odd frame<br>1: Set DATA1 PID enabled or forced odd frame                           |
| Bit 28 | SETD0PID/<br>SETEVENFR | 0x0         | rw   | Set DATA0 PID<br>Applies to interrupt/bulk IN endpoints only. Writing to this bit sets the endpoint data PID bit in this register to DATA0.<br>Set Even frame<br>Applies to synchronous IN endpoints only. Writing to this bit sets the Even/Odd frame to even frame.<br>0: Disabled Set DATA0 PID disabled or Do not force even frame<br>1: Set DATA0PID or set the EOFRNUM to even frame                    |



| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 27    | SNAK     | 0x0         | wo   | <p>Set NAK</p> <p>A write to this bit sets the NAK bit for the endpoint. The application uses this bit to control the transmission of NAK handshakes on an endpoint. The controller sets this bit on a Transfer completed interrupt or after receiving a SETUP packet.</p> <p>Values:</p> <p>0: Do not set NAK</p> <p>1: Set NAK</p>  |
| Bit 26    | CNAK     | 0x0         | wo   | <p>Clear NAK</p> <p>A write to this bit clears the NAK bit for this endpoint.</p> <p>0: Not clear NAK</p> <p>1: Clear NAK</p>   |
| Bit 25:22 | TXFNUM   | 0x0         | rw   | <p>TxFIFO number</p> <p>Allocate FIFO number to the corresponding endpoint. A separate FIFO number is allocated to each valid IN endpoint. This bit applies to IN endpoints only.</p>   |
| Bit 21    | STALL    | 0x0         | rw   | <p>STALL handshake</p> <p>Applies to non-control, non-synchronous IN and OUT endpoints.</p> <p>The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, global non-periodic IN NAK bit or global OUT NAK bit is set along with this bit, the STALL bit has priority. Only the application can clear this bit, but the controller never.</p> <p>0: Stall all invalid tokens</p> <p>1: Stall all valid tokens</p>  |
| Bit 20    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 19:18 | EPTYPE   | 0x0         | rw   | <p>Endpoint type</p> <p>This is the transfer type supported by this logical endpoint.</p> <p>00: Control</p> <p>01: Synchronous</p> <p>10: Bulk</p> <p>11: Interrupt</p>  |
| Bit 17    | NAKSTS   | 0x0         | ro   | <p>NAK status</p> <p>Indicates the following status:</p> <p>0: The controller is sending non-NAK handshakes based on the FIFO status</p> <p>1: The controller is sending NAK handshakes</p> <p>When this bit is set (either by the application or the controller), the controller stops receiving any data on an OUT endpoint, even if there is space in the receive FIFO to accommodate the incoming data packets.</p> <p>For non-synchronous IN endpoints: the controller stops transmitting data on the endpoint, even if there is data pending in the transmit FIFO.</p> <p>For synchronous IN endpoints: the controller sends a zero-length data packet, even if there is space in the transmit FIFO.</p> <p>The controller always responds to SETUP data packets with an ACK handshake, regardless of whether this bit is set or not.</p> |

| Bit       | Name             | Reset value | Type | Description   |
|-----------|------------------|-------------|------|---|
| Bit 16    | DPID/<br>EOFRNUM | 0x0         | ro   | <p>Endpoint data PID</p> <p>Applies to interrupt/bulk IN endpoints only.</p> <p>This bit contains the PID of the packet to be transmitted on this endpoint. The application must program the PID of the initial data packet to be received or transmitted on this endpoint, after the endpoint is enabled. The application programs DATA0 or DATA1 PID through the SetD1PID and SetD0PID of this register.</p> <p>0: DATA0<br/>1: DATA1</p> <p>Even/Odd frame</p> <p>Applies to synchronous IN endpoints only.</p> <p>Indicates the frame number in which the controller transmits synchronous data on this endpoint. The application must program the even/odd frame number in which it tends to transmit or receive synchronous data through the SETEVNFR and SETODDFR bits in this register.</p> <p>0: Even frame<br/>1: Odd frame</p> |
| Bit 15    | USBACEPT         | 0x0         | rw   | <p>USB active endpoint</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The controller clears this bit for all endpoints except for endpoint 0 after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program the endpoint registers and set this bit.</p> <p>0: Inactive<br/>1: Active</p>  |
| Bit 14:11 | Reserved         | 0x0         | resd | Kept at its default value.  |
| Bit 10:0  | MPS              | 0x000       | rw   | <p>Maximum packet size</p> <p>The application uses this field to set the maximum packet size for the current logical endpoint. The values are in bytes.</p>   |

#### OTGFS device control OUT endpoint 0 control register (OTGFS\_D0EPCCTL0)

This section describes the control OUT endpoint 0 control register. Non-zero control endpoints use registers for endpoints 1-7.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31    | EPTENA   | 0x0         | rw1s | <p>Endpoint enable</p> <p>The application sets this bit to start transmitting data on endpoint 0. The controller clears this bit before setting any one of the following interrupts on this endpoint:</p> <p>SETUP stage done<br/>Endpoint disabled<br/>Transfer completed</p> |
| Bit 30    | EPTDIS   | 0x0         | ro   | <p>Endpoint disable</p> <p>The application cannot disable control OUT endpoint 0.</p>  |
| Bit 29:28 | Reserved | 0x0         | resd | Kept at its default value.   |

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 27    | SNAK     | 0x0         | wo   | Set NAK<br>A write to this bit sets the NAK bit for this endpoint. The application can use this bit to control the transmission of NAK handshakes on an endpoint. The controller sets this bit on a transfer completed interrupt or when a SETUP data packet is received.   |
| Bit 26    | CNAK     | 0x0         | wo   | Clear NAK<br>A write to this bit clears the NAK for the endpoint.   |
| Bit 25:22 | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 21    | STALL    | 0x0         | rw1s | STALL handshake<br>The application sets this bit and the controller clears this bit when a SETUP token is received for this endpoint. If a NAK bit, global non-periodic OIT NAK bit is set along with this bit, the STALL bit has priority. The controller always responds to SETUP data packets, regardless of whether this bit is set or not.   |
| Bit 20    | SNP      | 0x0         | rw   | Snoop mode<br>This bit configures the endpoint to Snoop mode. In this mode, the controller does not check the correctness of OUT packets before transmitting OUT packets to the application memory.   |
| Bit 19:18 | EPTYPE   | 0x0         | ro   | Endpoint type<br>Hardware sets this bit to 0 to control endpoint type.  |
| Bit 17    | NAKSTS   | 0x0         | ro   | NAK status<br>Indicates the following:<br>0: The controller is sending non-NAK handshakes based on the FIFO status<br>1: The controller is sending NAK handshakes<br>When this bit is set (either by the application or the controller), the controller stops receiving any data on an OUT endpoint, even if there is space in the receive FIFO.<br>The controller always responds to SETUP data packets with an ACK handshake, regardless of whether this bit is set or not. |
| Bit 16    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 15    | USBACEPT | 0x1         | ro   | USB active endpoint<br>This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.   |
| Bit 14:2  | Reserved | 0x0000      | resd | Kept at its default value.  |
| Bit 1:0   | MPS      | 0x0         | ro   | Maximum packet size<br>The maximum packet size of the control OUT endpoint 0 is the same as that of the control IN endpoint 0.<br>00: 64 bytes<br>01: 32 bytes<br>10: 16 bytes<br>11: 8 bytes   |

### OTGFS device control OUT endpoint-x control register (OTGFS\_DOEPCTLx) (x = 1...7, where x is endpoint number)

This application uses this register to control the behavior of all endpoints other than endpoint 0.

| Bit    | Name                   | Reset value | Type | Description   |
|--------|------------------------|-------------|------|---|
| Bit 31 | EPTENA                 | 0x0         | rw1s | Endpoint enable<br>Indicates that the descriptor structure and data buffer for data reception has been configured. The controller clears this bit before setting any one of the following interrupts on this endpoint:<br>SETUP stage done<br>Endpoint disabled<br>Transfer completed   |
| Bit 30 | EPTDIS                 | 0x0         | ro   | Endpoint disable<br>The application sets this bit to stop transmitting data on an endpoint, even if the transfer on that endpoint is incomplete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The controller clears this bit before setting the endpoint disabled interrupt. The application must set this bit only when the endpoint enabled set.<br>0: No effect<br>1: Endpoint disabled |
| Bit 29 | SETD1PID/<br>SETODDFR  | 0x0         | rw   | Set DATA1 PID<br>Applies to interrupt/bulk OUT endpoints only. Writing to this bit sets the endpoint data PID bit in this register to DATA1.<br>Set odd frame<br>Applies to synchronous OUT endpoints only. Writing to this bit sets the Even/Odd frame to odd frame.<br>0: Disabled Set DATA1 PID disabled or Do not force odd frame<br>1: Set DATA1 PID enabled or forced odd frame   |
| Bit 28 | SETD0PID/<br>SETEVENFR | 0x0         | rw   | Set DATA0 PID<br>Applies to interrupt/bulk OUT endpoints only. Writing to this bit sets the endpoint data PID bit in this register to DATA0.<br>Set Even frame<br>Applies to synchronous OUT endpoints only. Writing to this bit sets the Even/Odd frame to even frame.<br>0: Disabled Set DATA0 PID disabled or Do not force even frame<br>1: Set DATA0PID or set the EOFRNUM to even frame  |
| Bit 27 | SNAK                   | 0x0         | wo   | Set NAK<br>A write to this bit sets the NAK bit for the endpoint. The application uses this bit to control the transmission of NAK handshakes on an endpoint. The controller sets this bit on a Transfer completed interrupt or after receiving a SETUP packet.<br>Values:<br>0: Do not set NAK<br>1: Set NAK   |
| Bit 26 | CNAK                   | 0x0         | wo   | Clear NAK<br>A write to this bit clears the NAK bit for the endpoint.<br>0: Not clear NAK<br>1: Clear NAK   |

| Bit       | Name             | Reset value | Type | Description  |
|-----------|------------------|-------------|------|--|
| Bit 25:22 | Reserved         | 0x0         | resd | Kept at its default value.   |
| Bit 21    | STALL            | 0x0         | rw   | Applies to non-control, non-synchronous IN and OUT endpoints.<br>The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, global non-periodic IN NAK bit or global OUT NAK bit is set along with this bit, the STALL bit has priority. Only the application can clear this bit, but the controller never.   |
| Bit 20    | SNP              | 0x0         | rw   | Snoop mode<br>This bit configures the endpoint to Snoop mode. In this mode, the controller does not check the correctness of OUT packets before transmitting OUT packets to the application memory.  |
| Bit 19:18 | EPTYPE           | 0x0         | rw   | Endpoint type<br>This is the transfer type supported by this logical endpoint.<br>00: Control<br>01: Synchronous<br>10: Bulk<br>11: Interrupt  |
| Bit 17    | NAKSTS           | 0x0         | ro   | NAK status<br>Indicates the following:<br>0: The controller is sending non-NAK handshakes based on the FIFO status<br>1: The controller is sending NAK handshakes<br>When this bit is set (either by the application or the controller), the controller stops receiving any data on an OUT endpoint, even if there is space in the receive FIFO to accommodate the incoming data packets.<br>For non-synchronous IN endpoints: the controller stops transmitting data on the endpoint, even if there is data pending in the transmit FIFO.<br>For synchronous IN endpoints: the controller sends a zero-length data packet, even if there is space in the transmit FIFO.<br>The controller always responds to SETUP data packets with an ACK handshake, regardless of whether this bit is set or not.  |
| Bit 16    | DPID/<br>EOFRNUM | 0x0         | ro   | Endpoint data PID<br>Applies to interrupt/bulk OUT endpoints only.<br>This bit contains the PID of the packet to be transmitted on this endpoint. The application must program the PID of the initial data packet to be received or transmitted on this endpoint, after the endpoint is enabled. The application programs DATA0 or DATA1 PID through the SetD1PID and SetD0PID of this register.<br>0: DATA0<br>1: DATA1<br>Even/Odd frame<br>Applies to synchronous OUT endpoints only.<br>Indicates the frame number in which the controller transmits synchronous data on this endpoint. The application must program the even/odd frame number in which it tends to transmit or receive synchronous data through the SETEVNFR and SETODDFR bits in this register.<br>0: Even frame<br>1: Odd frame |

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 15    | USBACEPT | 0x0         | rw   | USB active endpoint<br>Indicates whether this endpoint is active in the current configuration and interface. The controller clears this bit for all endpoints except for endpoint 0 after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program the endpoint registers and set this bit.<br>0: Inactive<br>1: Active |
| Bit 14:11 | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 10:0  | MPS      | 0x000       | rw   | Maximum packet size<br>The application uses this field to set the maximum packet size for the current logical endpoint. The values are in bytes.   |

#### OTGFS device IN endpoint-x interrupt register (OTGFS\_DIEPINTx) (x = 0...7, where x is endpoint number)

This register indicates the status of an endpoint when USB and AHB-related events occurs, as shown in Figure 240. When the IEPINT bit of the OTGFS\_GINTSTS register is set, the application must first read the OTGFS\_DAIN register to get the exact endpoint number in which the event occurs, before reading the endpoint interrupt registers. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTGFS\_DAIN and OTGFS\_GINTSTS registers.

| Bit      | Name        | Reset value | Type | Description  |
|----------|-------------|-------------|------|--|
| Bit 31:8 | Reserved    | 0x000000    | resd | Kept at its default value.   |
| Bit 7    | TXFEMP      | 0x1         | ro   | Transmit FIFO empty<br>This interrupt is generated when the transmit FIFO for this endpoint is half or completely empty. The half or completely empty status depends on the transmit FIFO empty level bit in the controller AHB configuration register.  |
| Bit 6    | INEPTNAK    | 0x0         | rw1c | IN endpoint NAK effective<br>This bit can be cleared by writing 1 to the CNAK bit in the DIEPCTLx register.<br>This interrupt indicates that the IN endpoint NAB bit set by the application has taken effect.<br>This interrupt does not guarantee that a NAK handshake is sent on the USB line. A STALL bit has priority over a NAK bit.<br>This bit applies to the scenario only when the endpoint is enabled. |
| Bit 5    | Reserved    | 0x0         | resd | Kept at its default value.   |
| Bit 4    | INTKNTXFEMP | 0x0         | rw1c | N token received when Tx FIFO is empty<br>Indicates that an IN token was received when the associated transmit FIFO (periodic or non-periodic) was empty. An interrupt is generated on the endpoint for which an IN token was received.  |
| Bit 3    | TIMEOUT     | 0x0         | rw1c | Timeout condition<br>Applies to control IN endpoints only. This bit indicates that the controller has detected a timeout condition for the last IN token on this endpoint.   |
| Bit 2    | Reserved    | 0x0         | resd | Kept at its default value.   |

| Bit   | Name    | Reset value | Type | Description   |
|-------|---------|-------------|------|---|
| Bit 1 | EPTDISD | 0x0         | rw1c | Endpoint disabled interrupt<br>This bit indicates that the endpoint is disabled according to the application's request.           |
| Bit 0 | XFERC   | 0x0         | rw1c | Transfer completed interrupt<br>Indicates that the programmed transfers are complete on the AHB and on the USB for this endpoint. |

#### OTGFS device OUT endpoint-x interrupt register (OTGFS\_DOEPINTx) (x = 0...7, where x is endpoint number)

This register indicates the status of an endpoint with respect to USB and AHB-related events, as shown in Figure 240. When the OEPIINT bit of the OTGFS\_GINTSTS register is set, the application must first read the OTGFS\_DAINTE register to get the exact endpoint number in which the event occurs, before reading the endpoint interrupt registers. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTGFS\_DAINTE and OTGFS\_GINTSTS registers.

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:7 | Reserved | 0x0000000   | resd | Kept at its default value.  |
| Bit 6    | B2BSTUP  | 0x0         | rw1c | Back-to-back SETUP packets received<br>Indicates that more than three back-to-back SETUP packets are received.  |
| Bit 5    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 4    | OUTTEPD  | 0x0         | rw1c | OUT token received when endpoint disabled<br>Applies to control OUT endpoints only.<br>Indicates that an OUT token was received when the endpoint has not yet been enabled. An interrupt is generated on the endpoint for which an OUT token was received.  |
| Bit 3    | SETUP    | 0x0         | rw1c | SETUP phase done<br>Applies to control OUT endpoints only.<br>Indicates that the SETUP stage for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. Upon this interrupt, the application can decode the received SETUP data packets. |
| Bit 2    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 1    | EPTDISD  | 0x0         | rw1c | Endpoint disabled interrupt<br>Indicates that the endpoint is disabled according to the application's request.  |
| Bit 0    | XFERC    | 0x0         | rw1c | Transfer completed interrupt<br>Indicates that the programmed transfers are complete on the AHB and on the USB for this endpoint.   |

### OTGFS device IN endpoint 0 transfer size register (OTGFS\_DIEPTSIZ0)

The application must set this register before enabling endpoint 0. Once the endpoint 0 is enabled using the endpoint enable pin in the device endpoint 0 control register, the controller modifies this register. The application can only read this register as long as the controller clears the endpoint enable bit.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:21 | Reserved | 0x000       | resd | Kept at its default value.   |
| Bit 20:19 | PKTCNT   | 0x0         | rw   | Packet count<br>Indicates the total number of USB packets that constitute the transfer size of data for the endpoint 0.<br>This field is decremented every time a packet is read from the transmit FIFO (maximum packet size or short packet)  |
| Bit 18:7  | Reserved | 0x000       | resd | Kept at its default value.   |
| Bit 6:0   | XFERSIZE | 0x00        | rw   | Transfer size<br>Indicates the transfer size (in bytes) for the endpoint 0. The controller interrupts the application when the transfer size becomes 0. The transfer size can be set to the maximum packet size of the endpoint at the end of each packet.<br>The controller decrements this field every time a packet from the external memory is written to the transmit FIFO. |

### OTGFS device OUT endpoint 0 transfer size register (OTGFS\_DOEPTSIZ0)

The application must set this register before enabling endpoint 0. Once the endpoint 0 is enabled using the endpoint enable pin in the device endpoint 0 control register, the controller modifies this register. The application can only read this register as long as the controller clears the endpoint enable bit.

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31    | Reserved | 0x0         | resd | Kept at its default value.   |
| Bit 30:29 | SUPCNT   | 0x0         | rw   | SETUP packet count<br>Indicates the number of back-to-back SETUP data packets the endpoint can receive.<br>01: 1 packet<br>10: 2 packets<br>11: 3 packets  |
| Bit 28:20 | Reserved | 0x000       | resd | Kept at its default value.   |
| Bit 19    | PKTCNT   | 0           | rw   | Packet count<br>This bit is decremented to 0 after a packet is written to the receive FIFO.  |
| Bit 18:7  | Reserved | 0x000       | resd | Kept at its default value.   |
| Bit 6:0   | XFERSIZE | 0x00        | rw   | Transfer size<br>Indicates the transfer size (in bytes) for the endpoint 0. The controller interrupts the application when the transfer size becomes 0. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.<br>The controller decrements this field every time a packet from the external memory is written to the transmit FIFO.<br>The controller decrements this field every time a packet from the receive FIFO is written to the external memory. |



### OTGFS device IN endpoint-x transfer size register (OTGFS\_DIEPTSIZx) (x = 1...7, where x is endpoint number)

The application must set this register before enabling endpoint x. Once the endpoint x is enabled using the endpoint enable pin in the device endpoint x control register, the controller modifies this register. The application can only read this register as long as the controller clears the endpoint enable bit.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 30:29 | MC       | 0x0         | rw   | Multi count<br>For periodic IN endpoints, this field indicates the number of packets to be transmitted on the USB for each frame. The controller uses this field to calculate the data PID transmitted on synchronous IN endpoints.<br>01: 1 packet<br>10: 2 packets<br>11: 3 packets   |
| Bit 28:19 | PKTCNT   | 0x000       | rw   | Packet count<br>Indicates the total number of USB packets (data transfer size on the endpoint) this field is decremented every time a packet is read from the transmit FIFO (maximum packet size and short packet).   |
| Bit 18:0  | XFERSIZE | 0x00000     | rw   | Transfer Size<br>Indicates the transfer size (in bytes) for the current endpoint. The controller interrupts the application when the transfer size becomes 0. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.<br>The controller decrements this field every time a packet from the external memory is written to the transmit FIFO. |

### OTGFS device IN endpoint transmit FIFO status register (OTGFS\_DTXFSTSx) (x = 1...7, where x is endpoint number)

This is a ready-only register containing the free space information for the device IN endpoint transmit FIFO.

| Bit       | Name       | Reset value | Type | Description   |
|-----------|------------|-------------|------|---|
| Bit 31:16 | Reserved   | 0x0000      | resd | Kept at its default value.  |
| Bit 15:0  | INEPTXFSAV | 0x0200      | ro   | IN endpoint TxFIFO space available<br>Indicates the amount of free space in the endpoint transmit FIFO. Values are in terms of 32-bit words.<br>0x0: Endpoint transmit FIFO is full<br>0x1: 1 word available<br>0x02: 2 words available<br>0xn: n words available (0 < n < 512)<br>0x200: Remaining 512 words<br>Others: Reserved |

### OTGFS device OUT endpoint-x transfer size register (OTGFS\_DOEPTSIZx) (x = 1...7, where x is endpoint number)

The application must set this register before enabling endpoint x. Once the endpoint x is enabled using the endpoint enable pin in the device endpoint x control register, the controller modifies this register. The application can only read this register as long as the controller clears the endpoint enable bit.

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31    | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 30:29 | RXDPID   | 0x0         | ro   | Received data PID<br>Applies to synchronous OUT endpoints only.<br>This is the data PID received in the last packet.<br>00: DATA0<br>01: DATA2<br>10: DATA1<br>11: MDATA<br>SETUP packet count<br>Applies to synchronous OUT endpoints only. Indicates the number of back-to-back SETUP data packets the endpoint can receive.<br>01: 1 packet<br>10: 2 packets<br>11: 3 packets                                  |
| Bit 28:19 | PKTCNT   | 0x000       | rw   | Packet count<br>Indicates the number of USB packets transmitted on the endpoint.<br>This field is decremented every time a packet is written to the receive FIFO (maximum packet size and short packet)   |
| Bit 18:0  | XFERSIZE | 0x00000     | rw   | Transfer size<br>Indicates the transfer size (in bytes) for the current endpoint. The controller interrupts the application when the transfer size becomes 0. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.<br>The controller decrements this field every time a packet is read from the receive FIFO and written to the external memory. |

## Power and clock control registers

### OTGFS power and clock gating control register (OTGFS\_PCGCCTL)

This register is available in host and device modes.

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:5 | Reserved | 0x0000000   | resd | Kept at its default value.  |
| Bit 4    | SUSPENDM | 0x0         | ro   | PHY suspend<br>Indicates that the PHY has been suspended.   |
| Bit 3:1  | Reserved | 0x0         | resd | Kept at its default value.  |
| Bit 0    | STOPPCLK | 0x0         | rw   | Stop PHY clock<br>The application uses this bit to stop PHY clock when the USB is suspended, session is invalid or device is disconnected. The application clears this bit when the USB is resumed or a new session starts. |

# 21 HICK auto clock calibration (ACC)

## ACC introduction

HICK auto clock calibration (HICK ACC), which uses the SOF signal (1 ms of period) generated as a reference signal, implements the sampling and calibration for the HICK clocks.

The main purpose of this module is to provide a clock of 48 MHz  $\pm$  0.25 % for the USB device.

It is able to make the calibrated frequency as close to the target frequency as possible by means of “cross and return” algorithm.

## Main features

- Programmable center frequency
- Programmable boundary frequency that triggers calibration function
- Center frequency precision  $\pm$ 0.25 %
- Status detection flags
  - Calibration ready flag
- Error detection flags
  - Reference signal lost error flag
- Two interrupt source flag
  - Calibration ready flag
  - Reference signal lost error flag
- Two calibration modes: coarse calibration and fine calibration

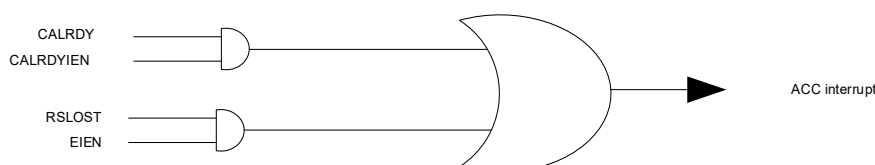
## Interrupt requests

**Table 80. ACC interrupt requests**

| Interrupt event       | Event flag | Enable bit |
|-----------------------|------------|------------|
| Calibration ready     | CALRDY     | CALRDYIEN  |
| Reference signal lost | RSLOST     | EIEN       |

ACC interrupt events are linked to the same interrupt vector (see Figure 252). Interrupt events include:

- During calibration process: When the calibration gets ready or reference signal lost occurs, the corresponding interrupt will be generated if the corresponding enable bit is enabled.



**Figure 252. ACC interrupt mapping diagram**

## Functional description

Auto clock calibration (HICK ACC), which uses the SOF signal (1 ms of period) generated as a reference signal, implements the sampling and calibration for the HICK clocks. In particular, the HICK clock frequency can be calibrated to a precision of  $\pm 0.25\%$  so as to meet the needs of the high-precision clock applications such as USB.

The signals of the module are connected to the CRM and HICK inside the microcontroller instead of being connected to the pins externally.

- **CRM\_HICKCAL**: the HICKCAL bit in the CRM module. This signal is used to calibrate the HICK in bypass mode. The value is defined by the HICKCAL[7:0] in the CRM\_CTRL register.
- **CRM\_HICKTRIM**: the HICKTRIM bit in the CRM module. This signal is used to calibrate the HICK in bypass mode. The value is defined by the HICKTRIM[5:0] in the CRM\_CTRL register.

The default value of the HICK is 32, which can be calibrated to  $8\text{ MHz} \pm 0.25\%$ . The HICK frequency can be adjusted by 20 kHz (design value) each time when the CRM\_HICKTRIM value changes. In other words, the HICK output frequency will increase by 20kHz each time the CRM\_HICKTRIM value is decremented by one; the HICK output frequency will reduce by 20kHz each time the CRM\_HICKTRIM value is incremented by one.

- **USB\_SOF**: USB Start-of-Frame signal given by the USB device. Its high-level width is 12 system clock cycles, a pulse signal of 1 ms.
- **HICKCLK**: HICK clock. The original HICK output frequency is 48MHz, but the sampling clock used by the HICK calibration module is frequency divider (1/6) clock, about 8MHz.
- **HICKCAL**: HICK module calibration signal. For the HICK clock after frequency division (1/6), the HICK clock frequency will change by 40 KHz (design value) each time the HICKCAL changes, which is positively correlated. In other words, the HICK clock frequency will increase by 40 KHz (design value) each time the HICKCAL is incremented by one; the HICK clock frequency will reduce by 40KHz each time the HICKCAL is decremented by one.
- **HICKTRIM**: HICK module calibration signal. For the HICK clock after frequency division (1/6), the HICK clock frequency will change by 20 KHz (design value) each time the HICKCAL changes, which is positively correlated.

Refer to Section 21.6 for more information about the bit definition in the registers.

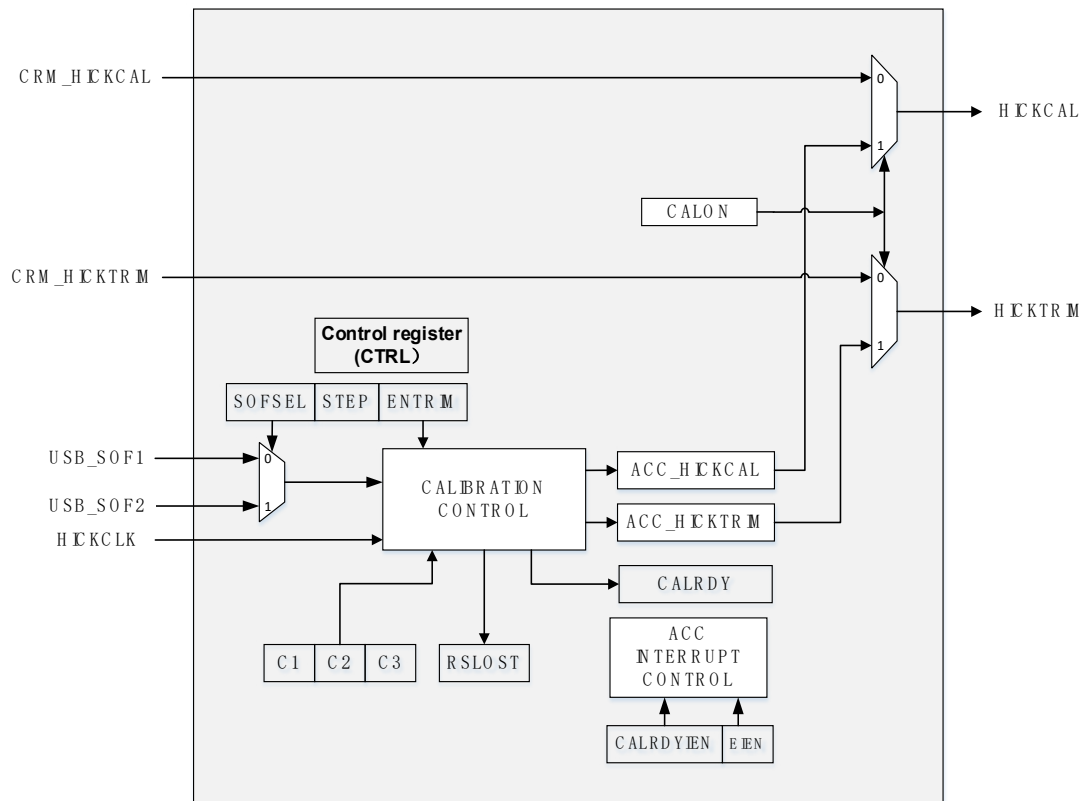


Figure 253. ACC block diagram

## Principle

USB\_SOF period signal: 1ms of period must be accurate, which is a prerequisite of the normal operation of an auto calibration module.

cross-return algorithm: This is used to calculate a calibration value closest to the theoretic value. In theory, the actual frequency after calibration can be adjusted to be within an accuracy range of about 0.5 steps from the target frequency (8 MHz).

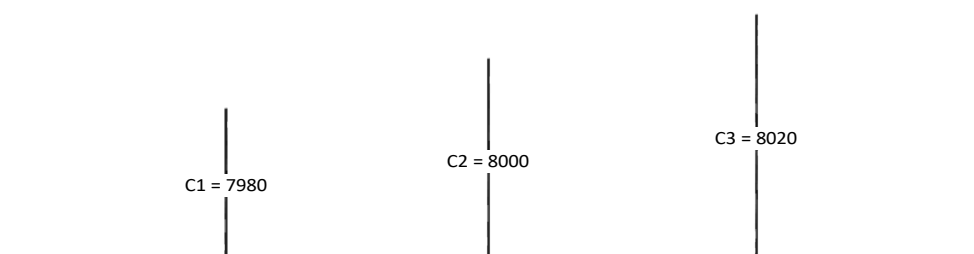


Figure 254. Cross-return algorithm

From the above figure, auto calibration function will adjust the `HICKCAL` or `HICKTRIM` according to the specified step as soon as the condition for triggering auto calibration is reached.

### **Cross:**

If the auto calibration condition is met, the actual sampling data in the first 1ms period will be either less than C2, or greater than C2.

When this value is less than C2, the auto calibration module will start increasing either the HICKCAL or HICKTRIM according to the step definition until the actual sampling value is greater than C2. In this way, the actual value will cross over C2 from small to large.

When this value is greater than C2, the auto calibration module will start decrease either the HICKCAL or HICKTRIM according to the step definition until the actual sampling value become less than C1. In this way, the actual value will cross over C2 from large to small.

### **Return:**

After cross operation is completed, the actual value closest to C2 can be obtained by comparing the difference (calculated as absolute value) between the actual sampling value and C2 before and after crossing C2 so as to get the best calibration value HICKCAL or HICKTRIM.

If the difference after crossing is less than the one before crossing C2, the calibration value after crossing prevails, and stops the calibration process until the next condition for auto calibration appears.

If the difference after crossing is greater than the one before crossing C2, the calibration value before crossing prevails, and it will return by one step to the one before crossing, and stops the calibration process until the next condition for auto calibration appears.

According to the cross-return strategy, in theory, it is possible to get the frequency accuracy that is 0.5 steps away from the center frequency.

Four conditions for enabling auto calibration function are as follows:

1. The rising edge of the CANLON (from 0 to 1)
2. When CALON = 1, reference signal is lost and restored
3. When the sample counter is less than C1
4. When the sample counter is greater than C3

Even though the sampling counter is between C1 and C3, at the rising edge the CANLON, the auto calibration module can also be activated so that the HICK frequency can be adjusted to be within a range of 0.5 steps of the center frequency as soon as the CANLON is enabled.

Under one of the above-mentioned circumstances, the HICK frequency can be calibrated to be within 0.5 steps of the center frequency. To achieve the best calibration accuracy, it is recommended to remain step as 1 (default value). If the step is set to 0, either HICKCAL or HICKTRIM will not be able to be calibrated.

## Register description

Refer to the list of abbreviations used in register descriptions.

These peripheral registers must be accessed by words (32 bits).

### ACC register map

**Table 81. ACC register map and reset values**

| Register  | Offset | Reset value  |
|-----------|--------|--------------|
| ACC_STS   | 0x00   | 0x0000 000   |
| ACC_CTRL1 | 0x04   | 0x0000 0100  |
| ACC_CTRL2 | 0x08   | 0x0000 2080  |
| ACC_C1    | 0x0C   | 0x0000 1F2C  |
| ACC_C2    | 0x10   | 0x0000 1F40  |
| ACC_C3    | 0x14   | 0x00000 1F54 |

### Status register (ACC\_STS)

| Bit      | Name     | Reset value | Type | Description   |
|----------|----------|-------------|------|---|
| Bit 31:9 | Reserved | 0x0000000   | resd | Kept at its default value.  |
| Bit 1    | RSLOST   | 0x0         | ro   | Reference Signal Lost<br>0: Reference Signal is not lost<br>1: Reference Signal is lost<br><br>Note: During the calibration, when the sample counter of the calibration module is twice that of C2, if a SOF reference signal is not detected, it means that the reference signal is lost. The internal statue machine will move to the idle state unless another SOF signal is detected; otherwise, the internal clock sample counter remains 0. The RSLOST bit is immediately cleared after the CALON bit is cleared or when the RSLOST is written with 0. Reference signal detection occurs only when CALON = 1. |
| Bit 0    | CALRDY   | 0x0         | ro   | Internal high-speed clock calibration ready<br>0: Internal 8 MHz oscillator calibration is not ready<br>1: Internal 8 MHz oscillator calibration is ready<br><br>Note: This bit is set by hardware to indicate that internal 8MHz oscillator has been calibrated to the frequency closest to 8 MHz. The CALRDY is immediately cleared after the CALON bit is cleared or when the CALRDY is written with 0.  |

## Control register 1 (ACC\_CTRL1)

| Bit       | Name      | Reset value | Type | Description  |
|-----------|-----------|-------------|------|--|
| Bit 31:12 | Reserved  | 0x00000     | resd | Forced to 0 by hardware.   |
| Bit 11:8  | STEP      | 0x1         | rw   | <p>Calibrated step</p> <p>This field defines the value after each calibration.</p> <p>Note: It is recommended to set the step bit in order to get a more accurate calibration result. While ENTRIM = 0, only the HICKCAL is calibrated. If the step is incremented or decremented by one, the HICKCAL will be incremented or decremented by one accordingly, and the HICK frequency will increase or decrease by 40 KHz (design value). This is a positive relationship.</p> <p>While ENTRIM = 1, only the HICKTRIM is calibrated. If the step is incremented or decremented by one, the HICKTRIM will be incremented or decremented by one accordingly, and the HICK frequency will increase or decrease by 20 KHz (design value). This is a positive relationship.</p> |
| Bit 7:6   | Reserved  | 0x0         | rw   | Forced by hardware to 0  |
| Bit 5     | CALRDYIEN | 0x0         | rw   | <p>CALRDY interrupt enable</p> <p>This bit is set or cleared by software.</p> <p>0: Interrupt generation disabled</p> <p>1: ACC interrupt is generated when CALRDY = 1 in the ACC_STS register</p>   |
| Bit 4     | EIEN      | 0x0         | rw   | <p>RSLOST error interrupt enable</p> <p>This bit is set or cleared by software.</p> <p>0: Interrupt generation disabled</p> <p>1: ACC interrupt is generated when RSLOST = 1 in the ACC_STS register</p>   |
| Bit 3:2   | Reserved  | 0x0         | rw   | Forced by hardware to 0  |
| Bit 1     | ENTRIM    | 0x0         | rw   | <p>Enable trim</p> <p>This bit is set or cleared by software.</p> <p>0: HICKCAL is calibrated.</p> <p>1: HICKTRIM is calibrated.</p> <p>Note: It is recommended to set ENTRIM = 1 in order to get higher calibration accuracy.</p>   |
| Bit 0     | CALON     | 0x0         | rw   | <p>Calibration on</p> <p>This bit is set or cleared by software.</p> <p>0: Calibration disabled</p> <p>1: Calibration enabled, and starts searching for a pulse on the USB_SOF.</p> <p>Note: This module cannot be used without the USB_SOF reference signal. If there are no requirements on the accuracy of the HICK clock, it is unnecessary to enable this module.</p>   |



## Control register 2 (ACC\_CTRL2)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:14 | Reserved | 0x00000     | resd | Forced to 0 by hardware   |
| Bit 13:8  | HICKTRIM | 0x20        | ro   | Internal high-speed auto clock trimming<br>This field is read only, but not written.<br>Internal high-speed clock is adjusted by ACC module, which is added to the ACC_HICKCAL[7:0] bit. These bits allow the users to input a trimming value to adjust the frequency of the HICKRC oscillator according to the variations in voltage and temperature.<br>The default value is 32, which can trim the HICK to 8 MHz $\pm$ 0.25. The trimming value is 20 kHz (design value) between two consecutive ACC_HICKTRIM steps. |
| Bit 7:0   | HICKCAL  | 0x80        | ro   | Internal high-speed auto clock calibration<br>This field is read only, but not written.<br>Internal high-speed clock is adjusted by ACC module. These bits allow the users to input a trimming value to adjust the frequency of the HICKPC oscillator according to the variations in voltage and temperature.<br>The default value is 128, which can trim the HICK to 8 MHz $\pm$ 0.25. The trimming value is 40 kHz (design value) between two consecutive ACC_HICKCAL steps.  |

## Compare value 1 (ACC\_C1)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Forced to 0 by hardware   |
| Bit 15:0  | C1       | 0x1F2C      | rw   | Compare 1<br>This value is the lower boundary for triggering calibration, and its default value is 7980. When the number of clocks sampled by ACC in 1ms period is less than or equal to C1, auto calibration is triggered automatically.<br>When the actual sampling value (number of clocks in 1ms) is greater than C1 but less than C3, auto calibration is not enabled. |

## Compare value 2 (ACC\_C2)

| Bit       | Name     | Reset value | Type | Description   |
|-----------|----------|-------------|------|---|
| Bit 31:16 | Reserved | 0x0000      | resd | Forced to 0 by hardware   |
| Bit 15:0  | C2       | 0x1F40      | rw   | Compare 2<br>This value defines the number of clocks sampled for 8MHz (ideal frequency) clock in 1ms period , and its default value is 8000 (theoretical value)<br>As a center point of cross-return strategy, this value is used to calculate the calibration value closest to the theoretical value. In theory, the actual frequency after calibration can be trimmed to be within an accuracy of 0.5 steps from the target frequency (8 MHz) |

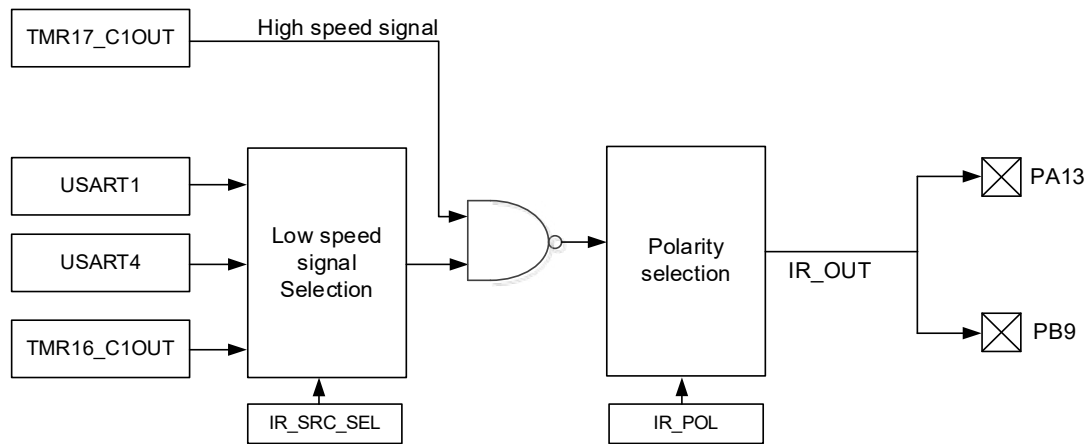
### Compare value 3 (ACC\_C3)

| Bit       | Name     | Reset value | Type | Description  |
|-----------|----------|-------------|------|--|
| Bit 31:16 | Reserved | 0x0000      | resd | Forced to 0 by hardware  |
| Bit 15:0  | C3       | 0x1F54      | rw   | <p>Compare 3</p> <p>This value is the upper boundary for triggering calibration. When the number of clock sampled by ACC in 1ms period is greater than or equal to C3, auto calibration is triggered automatically.</p> <p>When the actual sampling value (number of clocks in 1ms period) is greater than C1 but less than C3, auto calibration is not enabled.</p> |

## 22 Infrared timer (IRTMR)

The IRTMR (Infrared Timer) is used to generate the IR\_OUT signal that drives the infrared LED so as to achieve infrared control.

The IR\_OUT signals consists of a low-frequency modulation envelope and high-frequency carrier signals. The low-frequency modulation envelope signal selects from TMR10\_C1OUT, USART1 and USART through the IR\_SRC\_SEL[1:0] bit in the SCFG\_CFG1 register, while the high-frequency carrier signal is provided by the TMR11\_C1OUT register. The IR\_POL bit in the SCFG\_CFG1 register controls whether the IR\_OUT output is reversed or not. The IR\_OUT signal is output through multiplexed function via PB9 or PA13 (multiplexed mode needs to be configured in advance).



**Figure 255. IRTMR block diagram**

# 23 Debug (DEBUG)

## Debug introduction

Cortex®-M4 core provides powerful debugging features including halt and single step support, as well as trace function that is used for checking the details of the program execution. The debug features are implemented with a serial wire debug interface.

ARM Cortex®-M4 reference documentation:

- Cortex®-M4 Technical Reference Manual (TRM)
- ARM Debug Interface V5
- ARM CoreSight Design Kit revision r1p0 Technical Reference Manual

## Debug and Trace

It is possible to support debugging for different peripherals, and configure the status of peripherals during debugging. For timers and watchdogs, the user can select whether or not to stop or continue counting during debugging; For CAN, the user can select whether or not to stop or continue updating receive registers during debugging; For I2C, the user can select whether or not to stop or continue SMBUS timeout counting.

In addition, code debugging is supported in Low-power mode. In Sleep mode, the clock programmed by code remains active for HCLK and FCLK to continue to work. In Deepsleep mode, HICK oscillator is enabled to feed FCLK and HCLK.

There are several ID codes inside the MCU, which is accessible by the debugger using the DEBUG\_IDCODE at address 0xE0042000. It is part of the DEBUG and is mapped on the external PPB bus. These codes are accessible using the JTAG debug port or the SWD debug port or by the user software. They are even accessible while the MCU is under system reset.

Two trace interface modes supported: single-pin mode for serial wire view and multi-pin trace interface.

## I/O pin control

The HT32F49041 uses its two general-purpose I/O ports for SW-DP debugging. After reset, the SW-DP can be immediately used by the debugger by default.

When the debug ports are unused, these dedicated I/Os can be released for general-purpose I/Os through GPIO registers. After debug ports are released, they fall into the control of GPIO controller.

## DEGUB registers

Table 82 shows DEBUG register map and reset values.

These peripheral registers must be accessed by word (32 bits)

**Table 82. DEBUG register address and reset value**

| Register     | Offset      | Reset value |
|--------------|-------------|-------------|
| DEBUG_IDCODE | 0xE004 2000 | 0xFFFF XXXX |
| DEBUG_CTRL   | 0xE004 2004 | 0x0000 0000 |
| DEBUG_SER_ID | 0xE004 2020 | 0x0000 XX0X |

## DEBUG device ID (DEBUG\_IDCODE)

MCU integrates an ID code that is used to identify MCU's revision code. The DEBUG\_IDCODE register is mapped on the external PPB bus at address 0xE0042000. This code is accessible by the SW debug port or by the user code.

| Bit      | Name | Reset value | Type | Description     |
|----------|------|-------------|------|-----------------|
| Bit 31:0 | PID  | 0XXXXX XXXX | ro   | PID information |

| PID [31:0]  | HT32 part number | FLASH size | Packages       |
|-------------|------------------|------------|----------------|
| 0x5009_2103 | HT32F49041       | 64KB       | 64LQFP (7 × 7) |
| 0x5009_2106 | HT32F49041       | 64KB       | 48LQFP         |
| 0x5009_210F | HT32F49041       | 64KB       | 32QFN          |
| 0x5009_2112 | HT32F49041       | 64KB       | 20TSSOP        |

## DEBUG control register (DEBUG\_CTRL)

This register is asynchronously reset by POR Reset (not reset by system reset). It can be written by the debugger under reset.

| Bit       | Name               | Reset value | Type | Description   |
|-----------|--------------------|-------------|------|---|
| Bit 31:28 | Reserved           | 0x0000 0000 | resd | Always 0.   |
| Bit 27    | TMR14_PAUSE        | 0           | rw   | TMR14 debug control bit<br>0: TMR14 runs normally<br>1: TMR14 stops running   |
| Bit 26    | TMR13_PAUSE        | 0           | rw   | TMR13 debug control bit<br>0: TMR13 runs normally<br>1: TMR13 stops running   |
| Bit 25    | Reserved           | 0x0         | resd | Always 0.   |
| Bit 24    | TMR17_PAUSE        | 0           | rw   | TMR17 debug control bit<br>0: TMR17 runs normally<br>1: TMR17 stops running   |
| Bit 23    | TMR16_PAUSE        | 0           | rw   | TMR16 debug control bit<br>0: TMR16 runs normally<br>1: TMR16 stops running   |
| Bit 22    | TMR15_PAUSE        | 0           | rw   | TMR15 debug control bit<br>0: TMR15 runs normally<br>1: TMR15 stops running   |
| Bit 21    | ERTC_512_PAUSE     | 0           | rw   | ERTC 512Hz output clock pause control bit<br>0: ERTC 512 Hz output clock works normally<br>1: Froze 512 Hz output clock |
| Bit 20    | TMR7_PAUSE         | 0           | rw   | TMR7 debug control bit<br>0: TMR7 runs normally<br>1: TMR7 stops running  |
| Bit 19    | TMR6_PAUSE         | 0           | rw   | TMR6 debug control bit<br>0: TMR6 runs normally<br>1: TMR6 stops running  |
| Bit 18:17 | Reserved           | 0x0000 0000 | resd | Always 0.   |
| Bit 16    | I2C2_SMBUS_TIMEOUT | 0           | rw   | I2C2 pause control bit<br>0: I2C2 SMBUS timeout control works normally<br>1: I2C2 SMBUS timeout control stops running   |
| Bit 15    | I2C1_SMBUS_TIMEOUT | 0           | rw   | I2C1 pause control bit<br>0: I2C1 SMBUS timeout control works normally<br>1: I2C1 SMBUS timeout control stops running   |

| Bit     | Name            | Reset value | Type | Description   |
|---------|-----------------|-------------|------|---|
| Bit 14  | ERTC_PAUSE      | 0           | rw   | ERTC pause control bit<br>0: ERTC works normally<br>1: ERTC stops running   |
| Bit 13  | Reserved        | 0x0         | resd | Always 0.   |
| Bit 12  | TMR3_PAUSE      | 0           | rw   | TMR3 debug control bit<br>0: TMR3 runs normally<br>1: TMR3 stops running  |
| Bit 11  | TMR2_PAUSE      | 0           | rw   | TMR2 debug control bit<br>0: TMR2 runs normally<br>1: TMR2 stops running  |
| Bit 10  | TMR1_PAUSE      | 0           | rw   | TMR1 debug control bit<br>0: TMR1 runs normally<br>1: TMR1 stops running  |
| Bit 9   | WDT_PAUSE       | 0           | rw   | WDT pause control bit<br>0: WDT works normally<br>1: WDT stops running  |
| Bit 8   | WWDT_PAUSE      | 0           | rw   | WWDT pause control bit<br>0: WWDT works normally<br>1: WWDT stops running   |
| Bit 7:4 | Reserved        | 0x0         | resd | Always 0.   |
| Bit 3   | CAN_PAUSE       | 0           | rw   | CAN pause control bit<br>0: CAN1 works normally<br>1: CAN1 receive register pauses (does not receive data)  |
| Bit 2   | STANDBY_DEBUG   | 0           | rw   | Debug Standby mode control bit<br>0: The whole 1.2V digital circuit is unpowered in Standby mode<br>1: The whole 1.2V digital circuit is not unpowered in Standby mode, and the system clock is provided by the internal RC oscillator (HICK)   |
| Bit 1   | DEEPSLEEP_DEBUG | 0           | rw   | Debug Deepsleep mode control bit<br>0: In Deepsleep mode, all clocks in the 1.2V domain are disabled. When exiting from Deepsleep mode, the internal RC oscillator (HICK) is enabled, and HICK is used as the system clock source, and the software must reprogram the system clock according to application requirements.<br>1: In Deepsleep mode, system clock is provided by the internal RC oscillator (HICK). When exiting from Deepsleep mode, HICK is used as the system clock source, and the software must reprogram the system clock according to application requirements. |
| Bit 0   | SLEEP_DEBUG     | 0           | rw   | Debug Sleep mode control bit<br>0: When entering Sleep mode, CPU HCLK clock is disabled, but other clocks remain active. When exiting from Sleep mode, it is not necessary to reprogram the clock system.<br>1: When entering Sleep mode, all clocks keep running.  |

## DEBUG SERIES ID register (DEBUG\_SER\_ID)

This register is used to identify the MCU series ID and revision ID. It is mapped to external PPB bus. This register is asynchronously reset by POR Reset (not reset by system reset). The MCU series ID is accessible through SW debug port or user code.

| Bit       | Name     | Reset value | Type | Description                       |
|-----------|----------|-------------|------|-----------------------------------|
| Bit 31:16 | Reserved | 0x0000      | resd | Always 0.                         |
| Bit 15:8  | SER_ID   | 0xXX        | ro   | MCU series ID<br>HT32F49041: 0x0F |
| Bit 7:3   | Reserved | 0x0         | resd | Always 0.                         |
| Bit 2:0   | REV_ID   | 0xX         | ro   | Revision ID<br>0x0: Revision A    |

Copyright© 2025 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorize the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.