

# User Manual

**MM32F013x**

**32-Bit Microcontroller Based on ARM® Cortex® M0**

---

**Version: 1.00**

MindMotion reserves the right to change the relevant information without prior notification.

# Contents

<b>1 Memory and Bus Architecture .....</b>	1
1.1 System architecture .....	1
1.2 Memory organization .....	3
1.2.1 Introduction.....	3
1.2.2 Memory mapping and register addressing .....	3
1.3 Embedded SRAM .....	8
1.4 Flash memory overview .....	8
1.5 Boot configuration.....	8
<b>2 Embedded Flash Memory (FLASH) .....</b>	10
2.1 Key Flash features .....	10
2.2 Flash memory function description.....	10
2.2.1 Flash memory organization.....	10
2.2.2 Read operation of the Flash memory .....	9
2.2.3 Flash write and erase operations .....	12
2.3 Memory protection .....	19
2.3.1 Read protection .....	19
2.3.2 Write protection of main memory .....	20
2.3.3 Option byte write protection .....	20
2.4 Flash interrupt.....	20
2.5 Option byte description .....	21
2.6 Flash register descriptions .....	23
2.6.1 Flash access control register (FLASH_ACR) .....	23
2.6.2 Flash key register (FLASH_KEYR) .....	24
2.6.3 Flash OPTKEY register (FLASH_OPTKEYR) .....	24
2.6.4 Flash status register (FLASH_SR).....	25
2.6.5 Flash control register (FLASH_CR).....	25
2.6.6 Flash address register (FLASH_AR).....	27
2.6.7 Option byte register (FLASH_OBR) .....	27
2.6.8 Write protection register (FLASH_WRPR) .....	28

<b>3 Cyclic Redundancy Check (CRC) Calculation Unit .....</b>	29
3.1 CRC introduction .....	29
3.2 CRC main features .....	29
3.3 CRC function description .....	30
3.4 CRC registers .....	30
3.4.1 CRC data register (CRC_DR) .....	30
3.4.2 CRC independent data register (CRC_IDR).....	31
3.4.3 CRC control register (CRC_CTRL) .....	31
3.4.4 CRC data reverse register (CRC_REVERSE).....	32
<b>4 Hardware Division (HWDIV) .....</b>	33
4.1 Hardware division introduction .....	33
4.2 Main features of hardware division.....	33
4.3 Hardware division functions .....	33
4.4 Hardware division registers.....	33
4.4.1 Dividend register (HWDIV_DVDR).....	34
4.4.2 Divisor register (HWDIV_DVSR).....	34
4.4.3 Quotient register (HWDIV_QUOTR) .....	34
4.4.4 Remainder register (HWDIV_RMDR).....	35
4.4.5 HWDIV status register (HWDIV_SR).....	35
4.4.6 HWDIV control register (HWDIV_CR) .....	36
<b>5 Serial Transceiver Module (CSM).....</b>	37
5.1 Introduction .....	37
5.2 Main features .....	37
5.3 Function description.....	37
5.3.1 Block diagram .....	37
5.3.2 External port .....	39
5.3.3 Data reception .....	39
5.3.4 Data transmission .....	39
5.3.5 Status flag.....	39
5.3.6 Baud rate configuration.....	39
5.3.7 DMA communication .....	40
5.4 Register file and memory mapping description.....	40

5.4.1 CSM transmit data register 1 (CSM_TXREG1) .....	40
5.4.2 CSM transmit data register 2 (CSM_TXREG2) .....	41
5.4.3 CSM receive data register 1 (CSM_RXREG1) .....	41
5.4.4 CSM receive data register 2 (CSM_RXREG2) .....	41
5.4.5 CSM interrupt status register (CSM_INTSTAT) .....	42
5.4.6 CSM interrupt enable register (CSM_INTEN).....	43
5.4.7 CSM control register 1 (CSM_CTL1) .....	43
5.4.8 CSM control register 2 (CSM_CTL2) .....	44
5.4.9 CSM configuration register (CSM_CFG) .....	45
5.4.10 CSM baud rate generator (CSM_SPBRG) .....	46
5.4.11 CSM data bit count (CSM_BCNT).....	46
<b>6 Power Control (PWR)</b> .....	48
6.1 Power supplies .....	48
6.1.1 Independent A/D converter supply and reference voltage .....	48
6.1.2 Battery backup domain .....	49
6.1.3 Voltage regulator.....	49
6.2 Power supply supervisor.....	49
6.2.1 Power-on reset (POR)/power-down reset (PDR).....	49
6.2.2 Programmable voltage detector (PWD) .....	50
6.3 Low-power modes .....	50
6.3.1 Slowing down system clocks.....	51
6.3.2 Peripheral clock gating.....	51
6.3.3 Sleep mode .....	51
6.3.4 Stop mode .....	52
6.3.5 Standby mode.....	53
6.4 Power control registers .....	54
6.4.1 Power control register (PWR_CR) .....	54
6.4.2 Power control/status register (PWR_CSR).....	56
<b>7 Backup Registers (BKP)</b> .....	58
7.1 BKP introduction .....	58
7.2 BKP features.....	58
7.3 BKP functional description .....	58

7.3.1 Tamper detection .....	58
7.3.2 RTC calibration.....	59
7.4 BKP register description.....	59
7.4.1 Backup data register n(BKP_DRn)(n = 1...20) .....	59
7.4.2 RTC clock calibration register (BKP_RTCCR).....	60
7.4.3 Backup control register (BKP_CR).....	61
7.4.4 Backup control/status register (BKP_CSR) .....	62
<b>8 Real-Time Clock (RTC) .....</b>	<b>64</b>
8.1 RTC introduction.....	64
8.2 Main features.....	64
8.3 Functional description .....	64
8.3.1 Overview.....	65
8.3.2 Resetting RTC registers.....	65
8.3.3 Reading RTC registers .....	62
8.3.4 Configuring RTC registers.....	66
8.3.5 RTC flag assertion.....	66
8.4 RTC alarm description .....	67
8.4.1 RTC control register (RTC_CRH).....	68
8.4.2 RTC control register low (RTC_CRL) .....	69
8.4.3 RTC prescaler load register (RTC_PRLH/RTC_PRLL).....	71
8.4.4 RTC prescaler divider register (RTC_DIVH/RTC_DIVL).....	71
8.4.5 RTC counter register (RTC_CNTH/RTC_CNTL) .....	72
8.4.6 RTC alarm register (RTC_ALRH/RTC_ALRL).....	73
8.4.7 RTC millisecond register (RTC_MSRH/RTC_MSRL) .....	74
<b>9 Reset and Clock Control (RCC) .....</b>	<b>75</b>
9.1 Reset .....	75
9.1.1 System reset.....	75
9.1.2 Power reset.....	75
9.1.3 Backup domain reset .....	76
9.2 Clocks.....	76
9.2.1 HSE clock .....	78
9.2.2 HSI clock.....	79

9.2.3 PLL .....	79
9.2.4 LSE clock.....	79
9.2.5 LSI clock .....	80
9.2.6 System clock (SYSCLK) selection.....	80
9.2.7 Clock security system (CSS).....	80
9.2.8 RTC clock .....	80
9.2.9 Watchdog clock.....	81
9.2.10 Clock-out capability.....	81
<b>9.3 RCC register file and memory mapping description .....</b>	<b>81</b>
9.3.1 Clock control register (RCC_CR) .....	82
9.3.2 Clock configuration register (RCC_CFGR).....	84
9.3.3 Clock interrupt register (RCC_CIR) .....	87
9.3.4 APB2 peripheral reset register (RCC_APB2RSTR).....	90
9.3.5 APB1 peripheral clock enable register (RCC_APB1RSTR).....	91
9.3.6 AHB peripheral clock enable register (RCC_AHBENR).....	93
9.3.7 APB2 peripheral clock enable register (RCC_APB2ENR) .....	94
9.3.8 APB1 peripheral clock enable register (RCC_APB1ENR) .....	96
9.3.9 Backup domain control register (RCC_BDCR).....	98
9.3.10 Control/status register (RCC_CSR).....	100
9.3.11 AHB peripheral reset register (RCC_AHBRSTR).....	102
9.3.12 Clock configuration register 2(RCC_CFGR2) .....	103
9.3.13 RNG register (RCC_RNG) .....	104
9.3.14 System configuration register (RCC_SYSCFG) .....	105
<b>10 General-Purpose I/Os (GPIOs) .....</b>	<b>106</b>
10.1 GPIO functional description .....	106
10.1.1 General-purpose I/O (GPIO) .....	108
10.1.2 Atomic bit set or reset .....	108
10.1.3 External interrupt/wakeup lines .....	108
10.1.4 Alternate functions .....	108
10.1.5 Software remapping of I/O alternate functions.....	109
10.1.6 GPIO locking mechanism.....	109
10.1.7 Input configuration .....	109

10.1.8 Output configuration.....	110
10.1.9 Alternate function configuration.....	110
10.1.10 Analog input configuration.....	111
10.1.11 GPIO configurations for device peripherals .....	112
10.2. Alternate function I/O and debug configuration.....	115
10.2.1 OSC_IN/OSC_OUT used as GPIO port PD0/PD1.....	115
10.2.2 SWD alternate function remapping.....	115
10.3 GPIO register description.....	115
10.3.1 Port configuration register low (GPIOx_CRL)(x = A..D).....	116
10.3.2 Port configuration register high (GPIOx_CRH)(x = A..D) .....	116
10.3.3 Port input data register (GPIOx_IDR)(x = A..D).....	117
10.3.4 Port output data register (GPIOx_ODR)(x = A..D).....	118
10.3.5 Port bit set/reset register (GPIOx_BSRR)(x = A..D).....	118
10.3.6 Port bit reset register (GPIOx_BRR)(x = A..D).....	119
10.3.7 Port configuration lock register (GPIOx_LCKR)(x = A..D).....	119
10.3.8 Port output open drain control register (GPIOx_DCR)(x = A..D).....	120
10.3.9 Port alternate function register low (GPIOx_AFRL)(x = A..D) .....	120
10.3.10 Port alternate function register high (GPIOx_AFRH)(x = A..D) .....	121
<b>11 Interrupt and Events (EXTI) .....</b>	<b>123</b>
11.1 Nested vectored interrupt controller.....	123
11.1.1 SysTick calibration value register .....	123
11.1.2 Interrupt and exception vectors .....	123
11.2 External interrupt/event controller (EXTI).....	125
11.2.1 Main features .....	125
11.2.2 Block diagram .....	126
11.2.3 Wakeup event management.....	126
11.2.4 Functional description .....	126
11.2.5 External interrupt/event line mapping .....	127
11.3 EXTI register description .....	129
11.3.1 Interrupt mask register (EXTI_IMR).....	129
11.3.2 Event mask register (EXTI_EMR).....	129
11.3.3 Rising trigger selection register (EXTI_RTSR).....	130

11.3.4 Falling trigger selection register (EXTI_FTSR) .....	131
11.3.5 Software interrupt event register (EXTI_SWIER).....	131
11.3.6 Pending register (EXTI_PR).....	132
<b>12 Direct memory access controller (DMA) .....</b>	<b>133</b>
12.1 DMA introduction .....	133
12.2 DMA main features .....	133
12.3 DMA functional description.....	134
12.3.1 DMA transactions.....	134
12.3.2 Arbiter .....	135
12.3.3 DMA channels.....	135
12.3.4 Programmable data transfer width, data alignment and endians .....	136
12.3.5 Error management.....	139
12.3.6 Interrupts .....	139
12.3.7 DMA request mapping.....	139
12.4 DMA register description.....	140
12.4.1 DMA interrupt status register (DMA_ISR).....	140
12.4.2 DMA interrupt flag clear register (DMA_IFCR).....	141
12.4.3 DMA channel x configuration register (DMA_CCRx) (x = 1...5).....	142
12.4.4 DMA channel x number of data register (DMA_CNDTRx) (x = 1...5).....	144
12.4.5 DMA channel x peripheral address register (DMA_CPARx) (x = 1...5).....	145
12.4.6 DMA channel x memory address register (DMA_CMARx) (x = 1...5).....	145
<b>13 Analog-to-digital converter (ADC) .....</b>	<b>147</b>
13.1 ADC introduction .....	147
13.2 ADC main features.....	147
13.3 System block diagram.....	148
13.4 ADC functional description .....	148
13.4.1 ADC on-off control.....	149
13.4.2 Channel selection .....	150
13.5 Normal operation mode .....	150
13.5.1 Single conversion mode.....	150
13.5.2 One-cycle scan mode .....	150
13.5.3 Continuous scan mode .....	152

13.6 Arbitrary channel operation mode .....	153
13.6.1 Single conversion mode.....	154
13.6.2 One-cycle scan mode .....	154
13.6.3 Continuous scan mode .....	155
13.6.4 DMA request .....	157
13.7 Data alignment.....	157
13.7.1 Programmable resolution .....	157
13.7.2 Programmable sampling time .....	157
13.8 Conversion on external trigger .....	158
13.9 Temperature sensor .....	158
13.10 Internal reference voltage .....	159
13.11 Monitoring AD conversion result in the window comparator mode .....	159
13.12 ADC register description .....	159
13.12.1 A/D data register (ADC_ADDATA).....	160
13.12.2 A/D configuration register (ADC_ADCFG).....	161
13.12.3 A/D control register (ADC_ADCR).....	162
13.12.4 A/D channel selection register (ADC_ADCHS).....	165
13.12.5 A/D window compare register (ADC_ADCMPR).....	166
13.12.6 A/D status register (ADC_ADSTA) .....	167
13.12.7 A/D data register (ADC_ADDR0 ~ 914 ~ 15).....	168
13.12.8 A/D extended state register (ADC_ADSTA_EXT).....	169
13.12.9 A/D arbitrary channel selection register 0 (ADC_CHANY0).....	169
13.12.10 A/D arbitrary channel selection register 1 (ADC_CHANY1) .....	170
13.12.11 A/D arbitrary channel configuration register (ADC_ANY_CFG) .....	171
13.12.12 A/D arbitrary channel control register (ADC_ANY_CR) .....	171
<b>14 Comparator (COMP) .....</b>	<b>173</b>
14.1 COMP introduction.....	173
14.2 COMP main features.....	173
14.3 COMP functional description.....	173
14.3.1 Introduction .....	173
14.3.2 Clock.....	174
14.3.3 COMP switch .....	174

14.3.4 COMP inputs and outputs .....	174
14.3.5 COMP channel selection.....	174
14.3.6 Interrupt and wakeup .....	175
14.3.7 Power mode.....	175
14.3.8 Comparator LOCK mechanism .....	175
14.3.9 Hysteresis.....	176
14.4 COMP register description .....	176
14.4.1 COMP control and status register (COMP <sub>x</sub> _CSR)(x=1, 2) .....	176
14.4.2 COMP external reference voltage register (COMP_CRV) .....	179
14.4.3 COMP polling register (COMP <sub>x</sub> _POLL)(x=1, 2) .....	180
<b>15 Advanced-Control Timer (TIM1)</b> .....	<b>182</b>
15.1 TIM1 introduction .....	182
15.2 Main features .....	182
15.3 Functional description .....	183
15.3.1 Time-base unit.....	183
15.3.2 Counter modes .....	185
15.3.3 Repetition counter.....	194
15.3.4 Clock selection.....	195
15.3.5 Capture/compare channels .....	198
15.3.6 Input capture mode .....	200
15.3.7 PWM input mode .....	201
15.3.8 Forced output mode .....	202
15.3.9 Output compare mode .....	202
15.3.10 PWM mode .....	204
15.3.11 Complementary outputs and dead-time insertion .....	207
15.3.12 Using the break function .....	209
15.3.13 Clearing the OC <sub>x</sub> REF signal on an external event .....	211
15.3.14 6-step PWM generation .....	212
15.3.15 One-pulse mode .....	213
15.3.16 Encoder interface mode .....	215
15.3.17 Timer input XOR function .....	217
15.3.18 Interfacing with Hall sensors .....	217

15.3.19 TIMx and external trigger synchronization.....	218
15.3.20 Timer synchronization .....	221
15.3.21 Debug mode .....	221
<b>15.4 Register description .....</b>	<b>221</b>
15.4.1 Control register 1 (TIMx_CR1).....	223
15.4.2 Control register 2 (TIMx_CR2).....	225
15.4.3 Slave mode control register (TIMx_SMCR).....	228
15.4.4 DMA/Interrupt enable register (TIMX_DIER).....	230
15.4.5 Status register (TIMx_SR).....	232
15.4.6 Event generate register (TIMx_EGR).....	234
15.4.7 Capture/compare mode register 1 (TIMx_CCMR1).....	236
15.4.8 Capture/compare mode register 2 (TIMx_CCMR2) .....	240
15.4.9 Capture/compare enable register (TIMx_CCER).....	242
15.4.10 Counter (TIMx_CNT).....	246
15.4.11 Prescaler (TIMx_PSC).....	246
15.4.12 Auto-reload register (TIMx_ARR) .....	246
15.4.13 Repetition counter register (TIMx_RCR) .....	246
15.4.14 Capture/Compare register 1 (TIMx_CCR1) .....	247
15.4.15 Capture/Compare register 2 (TIMx_CCR2) .....	248
15.4.16 Capture/compare register 3 (TIMx_CCR3) .....	248
15.4.17 Capture/compare register 4 (TIMx_CCR4) .....	249
15.4.18 Brake and dead-time register (TIMx_BDTR) .....	249
15.4.19 DMA control register (TIMx_DCR) .....	252
15.4.20 DMA address for full transfer (TIMx_DMAR) .....	254
15.4.21 Capture/compare mode register 3 (TIMx_CCMR3) .....	254
15.4.23 PWM phase-shift/DMA repeat update request enable register (TIMx_PDER) ....	255
15.4.24 PWM phase-shift downcounting capture/compare register (CCRxFALL).....	256
<b>16 16-Bit General-Purpose Timer (TIM3) .....</b>	<b>257</b>
16.1 TIMx introduction .....	257
16.2 TIMx main features .....	257
16.3 TIMx functional description .....	258
16.3.1 Time-base unit .....	258

16.3.2 Counter modes .....	260
16.3.3 Clock selection.....	269
16.3.4 Capture/compare channel .....	272
16.3.5 Input capture mode .....	274
16.3.6 PWM input mode .....	275
16.3.7 Forced output mode.....	276
16.3.8 Output compare mode .....	276
16.3.9 PWM mode .....	277
16.3.10 One-pulse mode .....	280
16.3.11 Clearing the OC <sub>x</sub> REF signal on an external event.....	281
16.3.12 Encoder interface mode.....	282
16.3.13 Timer input XOR function .....	285
16.3.14 TIMx and external trigger synchronization.....	285
16.3.15 Timer synchronization .....	288
16.3.16 Debug mode .....	292
16.4 TIMx register description.....	293
16.4.1 Control register 1 (TIMx_CR1) .....	294
16.4.2 Control register 2 (TIMx_CR2) .....	296
16.4.3 Slave mode control register (TIMx_SMCR) .....	298
16.4.4 DMA/Interrupt enable register (TIMx_DIER).....	301
16.4.5 Status register (TIMx_SR).....	303
16.4.6 Event generate register (TIMx_EGR) .....	305
16.4.7 Capture/compare mode register 1 (TIMx_CCMR1) .....	306
16.4.8 Capture/compare mode register 2 (TIMx_CCMR2) .....	314
16.4.9 Capture/compare enable register (TIMx_CCER).....	315
16.4.10 Counter (TIMx_CNT).....	317
16.4.11 Prescaler (TIMx_PSC).....	317
16.4.12 Auto-reload register (TIMx_ARR) .....	318
16.4.13 Capture/compare register 1 (TIMx_CCR1).....	318
16.4.14 Capture/compare register 2 (TIMx_CCR2) .....	319
16.4.15 Capture/compare register 3 (TIMx_CCR3).....	319
16.4.16 Capture/compare register 4 (TIMx_CCR4).....	320

16.4.17 DMA control register (TIMx_DCR).....	320
16.4.18 DMA address for full transfer (TIMx_DMAR) .....	322
<b>17 32-Bit General-Purpose Timer (TIM2) .....</b>	<b>323</b>
17.1 TIMx introduction .....	323
17.2 TIMx main features .....	323
17.3 TIMx functional description .....	324
17.3.1 Time-base unit .....	324
17.3.2 Counter modes .....	326
17.3.3 Clock selection.....	335
17.3.4 Capture/compare channel .....	338
17.3.5 Input capture mode .....	340
17.3.6 PWM input mode .....	341
17.3.7 Forced output mode.....	342
17.3.8 Output compare mode .....	342
17.3.9 PWM mode .....	343
17.3.10 One-pulse mode .....	346
17.3.11 Clearing the OCxREF signal on an external event.....	347
17.3.12 Encoder Interface mode.....	348
17.3.13 Timer input XOR function .....	351
17.3.14 Timers and external trigger synchronization .....	351
17.3.15 Timer synchronization .....	354
17.3.16 Debug mode .....	359
17.4 TIMx register description.....	359
17.4.1 Control register 1 (TIMx_CR1) .....	360
17.4.2 Control register 2 (TIMx_CR2) .....	362
17.4.3 Slave mode control register (TIMx_SMCR) .....	364
17.4.4 DMA/Interrupt enable register (TIMx_DIER).....	367
17.4.5 Status register (TIMx_SR).....	369
17.4.6 Event generate register (TIMx_EGR) .....	370
17.4.7 Capture/compare mode register 1 (TIMx_CCMR1) .....	372
17.4.8 Capture/compare mode register 2 (TIMx_CCMR2) .....	376
17.4.9 Capture/compare enable register (TIMx_CCER).....	378

17.4.10 Counter (TIMx_CNT).....	379
17.4.11 Prescaler (TIMx_PSC).....	380
17.4.12 Auto-reload register (TIMx_ARR) .....	380
17.4.13 Capture/compare register 1 (TIMx_CCR1).....	381
17.4.14 Capture/compare register 2 (TIMx_CCR2).....	381
17.4.15 Capture/compare register 3 (TIMx_CCR3).....	382
17.4.16 Capture/compare register 4 (TIMx_CCR4).....	383
17.4.17 DMA control register (TIMx_DCR).....	383
17.4.18 DMA address for full transfer (TIMx_DMAR) .....	384
<b>18 Basic Timer (TIM14) .....</b>	<b>386</b>
18.1 TIM14 introduction .....	386
18.2 TIM14 main features .....	386
18.3 TIM14 functional description .....	387
18.3.1 Time-base unit .....	387
18.3.2 Counting mode .....	389
18.3.3 Clock source .....	391
18.3.4 Capture/compare channel .....	392
18.3.5 Input capture mode .....	394
18.3.6 Forced output mode .....	394
18.3.7 Output compare mode .....	395
18.3.8 PWM mode .....	396
18.3.9 Debug mode .....	397
18.4 TIM14 register description.....	397
18.4.1 Control register 1 (TIM14_CR1).....	398
18.4.2 Interrupt enable register (TIM14_DIER) .....	399
18.4.3 Status register (TIM14_SR).....	399
18.4.4 Event generation register (TIM14_EGR) .....	400
18.4.5 Capture/compare mode register 1 (TIM14_CCMR1).....	401
18.4.6 Capture/compare enable register (TIM14_CCER).....	406
18.4.7 Counter (TIM14_CNT) .....	407
18.4.8 Prescaler (TIM14_PSC) .....	407
18.4.9 Auto-reload register (TIM14_ARR) .....	408

18.4.10 Capture/compare register 1 (TIM14_CCR1).....	408
18.4.11 Break and dead-time register (TIM14_BDTR) .....	409
<b>19 Basic Timer (TIM16/17) .....</b>	<b>410</b>
19.1 TIM16/17 introduction .....	410
19.2 Main features .....	410
19.3 Functional description .....	411
19.3.1 Time-base unit .....	411
19.3.2 Counting mode .....	413
19.3.3 Clock source .....	415
19.3.4 Capture/compare channel .....	416
19.3.5 Input capture mode .....	418
19.3.6 Forced output mode.....	418
19.3.7 Output compare mode .....	419
19.3.8 PWM mode .....	420
19.3.9 Complementary outputs and dead-time insertion .....	421
19.3.10 Using the break function .....	423
19.3.11 One-pulse mode.....	425
19.3.12 Debug mode .....	426
19.4 Register description .....	426
19.4.1 TIM16/17 control register 1 (TIM16/17_CR1) .....	427
19.4.2 TIM16/17 control register 2 (TIM16/17_CR2) .....	428
19.4.3 TIM16/17 interrupt enable register (TIM16/17_DIER).....	429
19.4.4 TIM16/17 status register (TIM16/17_SR) .....	430
19.4.5 TIM16/17 event generation register 1 (TIM16/17_EGR).....	431
19.4.6 TIM16/17 capture/compare mode register 1 (TIM16/17_CCMR1) .....	432
19.4.7 TIM16/17 capture/compare enable register (TIM16/17_CCER).....	437
19.4.8 TIM16/17 counter (TIM16/17_CNT) .....	439
19.4.9 TIM16/17 prescaler (TIM16/17_PSC).....	439
19.4.10 TIM16/17 auto-reload register (TIM16/17_ARR).....	440
19.4.11 TIM16/17 capture/compare register 1 (TIM16/17_CCR1).....	440
19.4.12 TIM16/17 break and dead-time register (TIM16/17_BDTR).....	440
19.4.13 TIM16/17 DMA control register (TIM16/17_DCR).....	443

19.4.14 TIM16/17 address for full transfer (TIM16/17_DMAR) .....	444
<b>20 Independent Watchdog (IWDG).....</b>	<b>446</b>
20.1 Independent watchdog (IWDG) introduction.....	446
20.2 IWDG main features .....	446
20.3 IWDG functional description.....	446
20.3.1 Hardware watchdog .....	447
20.3.2 Register access protection.....	447
20.3.3 Debug mode .....	448
20.4 IWDG register description .....	448
20.4.1 Key register (IWDG_KR).....	448
20.4.2 Prescaler register (IWDG_PR).....	449
20.4.3 Reload register (IWDG_RLR).....	449
20.4.4 Status register (IWDG_SR).....	450
20.4.5 IWDG control (IWDG_CR) register .....	451
20.4.6 IWDG interrupt generate register (IWDG_IGEN).....	452
20.4.7 IWDG counter register (IWDG_CNT) .....	452
<b>21 Window Watchdog (WWDG).....</b>	<b>454</b>
21.1 WWDG introduction .....	454
21.2 WWDG main features .....	454
21.3 WWDG functional description .....	454
21.4 How to program the watchdog timeout.....	456
21.5 Debug mode .....	457
21.6 WWDG register description.....	457
21.6.1 Control register (WWDG_CR) .....	457
21.6.2 Configuration register (WWDG_CFGR).....	458
21.6.3 Status register (WWDG_SR).....	458
<b>22 Serial Peripheral Interface (SPI).....</b>	<b>460</b>
22.1 SPI introduction .....	460
22.2 Main features .....	460
22.3 SPI functional description.....	461
22.3.1 General description.....	461
22.3.2 SPI in slave mode .....	465

22.3.3 SPI in master mode.....	465
22.3.4 Status flags.....	466
22.3.5 Baud rate configuration.....	467
22.3.6 SPI communication using DMA.....	467
22.4 Register file and memory mapping description.....	468
22.4.1 Transmit data register (SPI_TXREG) .....	469
22.4.2 Receive data register (SPI_RXREG).....	469
22.4.3 Current status register (SPI_CSTAT).....	469
22.4.4 Interrupt status register (SPI_INTSTAT).....	470
22.4.5 Interrup enable register (SPI_INTEN) .....	472
22.4.6 Interrupt clear register (SPI_INTCLR) .....	473
22.4.7 Global control register (SPI_GCTL).....	474
22.4.8 Common control register (SPI_CCTL).....	476
22.4.9 Baud rate generator (SPI_SPBRG).....	477
22.4.10 Receive data number register (SPI_RXDNR).....	477
22.4.11 Slave chip select register (SPI_NSSR).....	478
22.4.12 Data control register (SPI_EXTCTL) .....	478
<b>23 Inter-Integrated Circuit (I2C) Interface .....</b>	<b>480</b>
23.1 I2C introduction.....	480
23.2 I2C main features.....	480
23.3 I2C protocol .....	480
23.3.1 Start and Stop conditions .....	481
23.3.2 Slave addressing protocol.....	481
23.3.3 Transmitting and receiving protocol.....	482
23.3.4 Tx buffer management and Start, Stop, and Restart generation.....	484
23.3.5 Multiple master arbitration.....	485
23.3.6 Clock synchronization .....	486
23.3.7 SCL configuration.....	487
23.4 I2C operation mode .....	488
23.4.1 Slave mode.....	489
23.4.2 Master mode.....	491
23.4.3 I2C abort transfer .....	492

23.5 Communication using DMA.....	493
23.6 I2C interrupts .....	493
23.7 I2C register description .....	494
23.7.1 I2C control register (I2C_CR).....	495
23.7.2 I2C target address register (I2C_TAR) .....	498
23.7.3 I2C Slave address register (I2C_SAR).....	498
23.7.4 I2C Data command register (I2C_DR) .....	499
23.7.5 Standard mode I2C clock SCL high count register (I2C_SSHR).....	499
23.7.6 Standard mode I2C clock SCL low count register (I2C_SSLR).....	499
23.7.7 Fast mode I2C clock SCL high count register (I2C_FSHR) .....	500
23.7.8 Fast mode I2C clock SCL low count register (I2C_FSLR) .....	500
23.7.9 I2C interrupt status register (I2C_ISR) .....	500
23.7.10 I2C interrupt mask register (I2C_IMR).....	501
23.7.11 I2C RAW interrupt status register (I2C_RAWISR) .....	501
23.7.12 I2C receive threshold register (I2C_RXTLR) .....	503
23.7.13 I2C transmit threshold register (I2C_TXTLR) .....	504
23.7.14 I2C combined and independent interrupt clear register (I2C_ICR) .....	504
23.7.15 I2C RX_UNDER interrupt clear register (I2C_RX_UNDER) .....	504
23.7.16 I2C RX_OVER interrupt clear register (I2C_RX_OVER) .....	505
23.7.17 I2C TX_OVER interrupt clear register (I2C_TX_OVER) .....	505
23.7.18 I2C RD_REQ interrupt clear register (I2C_RD_REQ) .....	505
23.7.19 I2C TX_ABRT interrupt clear register (I2C_TX_ABRT) .....	505
23.7.20 I2C RX_DONE interrupt clear register (I2C_RX_DONE).....	506
23.7.21 I2C ACTIVITY interrupt clear register (I2C_ACTIV).....	506
23.7.22 I2C STOP_DET interrupt clear register (I2C_STOP).....	506
23.7.23 I2C START_DET interrupt clear register (I2C_START) .....	507
23.7.24 I2C GEN_CALL interrupt clear register (I2C_GC) .....	507
23.7.25 I2C enable register (I2C_ENR) .....	507
23.7.26 I2C status register (I2C_SR) .....	508
23.7.27 I2C transmit FIFO level register (I2C_TXFLR) .....	509
23.7.28 I2C receive FIFO level register (I2C_RXFLR) .....	509
23.7.29 I2C SDA hold time register (I2C_HOLD) .....	510

23.7.30 I2C DMA controller (I2C_DMA) .....	510
23.7.31 I2C SDA setup time register (I2C_SETUP) .....	510
23.7.32 I2C general call ACK register (I2C_GCR) .....	511
23.7.33 I2C slave address mask register (I2C_SLVMASK) .....	511
23.7.34 I2C slave receive address register (I2C_SLVRCVADDR).....	511
<b>24 Universal Asynchronous Receiver Transmitter (UART)</b> .....	<b>513</b>
24.1 UART introduction.....	513
24.2 UART main features.....	513
24.3 UART functional description.....	514
24.3.1 UART character description .....	515
24.3.2 Transmitter.....	516
24.3.3 Receiver .....	518
24.3.4 9-bit data communication .....	519
24.3.5 Multiprocessor communication.....	519
24.3.6 Single-wire half-duplex communication.....	520
24.3.7 Smartcard .....	520
24.3.8 Fractional baud rate generator .....	522
24.3.9 Sampling.....	522
24.3.10 Parity control.....	523
24.3.11 Automatic baud rate detection .....	523
24.3.12 Communication using DMA.....	524
24.4 UART interrupt requests .....	524
24.5 UART register description .....	525
24.5.1 UART transmit data register (UART_TDR).....	526
24.5.2 UART receive data register (UART_RDR).....	526
24.5.3 UART current status register (UART_CSR).....	526
24.5.4 UART interrupt status register (UART_ISR) .....	527
24.5.5 UART interrupt enable register (UART_IER) .....	529
24.5.6 UART interrupt clear register (UART_ICR).....	530
24.5.7 UART global control register (UART_GCR).....	532
24.5.8 UART common control register (UART_CCR).....	533
24.5.9 UART baud rate register (UART_BRR) .....	535

24.5.10 UART fractional baud rate register (UART_FRA) .....	535
24.5.11 UART receive address register (UART_RXADDR).....	536
24.5.12 UART receive mask register (UART_RXMASK).....	536
24.5.13 UART SCR register (UART_SCR) .....	537
24.5.14 UART IDLE data length register (UART_IDLR) .....	538
24.5.15 UART automatic baud rate control register (UART_ABRCR) .....	538
<b>25 Controller Area Network (CAN).....</b>	<b>540</b>
25.1 CAN introduction.....	540
25.2 CAN main features.....	540
25.3 CAN controller general description.....	540
25.3.1 CAN 2.0B active core.....	541
25.3.2 CAN block diagram .....	541
25.3.3 Interface Management Logic (IML) .....	542
25.3.4 Transmit Buffer (TXB) .....	542
25.3.5 Receive Buffer (RXB, RXFIFO).....	542
25.3.6 Acceptance Filter (ACF) .....	542
25.3.7 Bit Stream Processor (BSP).....	543
25.3.8 Bit Timing Logic (BTL).....	543
25.3.9 Error Management Logic (EML) .....	543
25.4 CAN operating modes.....	543
25.4.1 Differences between Basic CAN and PeliCAN mode .....	543
25.5 CAN functional description.....	544
25.5.1 Basic CAN mode.....	544
25.5.2 Peli CAN mode .....	546
25.5.3 Transmission handling .....	549
25.5.4 Reception management.....	550
25.5.5 Identifier filtering .....	550
25.5.6 Message storage .....	557
25.5.7 Error management.....	560
25.5.8 Bit timing .....	564
25.5.9 Arbitration lost .....	564
25.5.10 CAN interrupt.....	565

25.6 CAN register description .....	566
25.6.1 CAN mode register (CAN_MOD).....	568
25.6.2 CAN control register (CAN_CR).....	568
25.6.3 CAN command register (CAN_CMR) .....	569
25.6.4 CAN status register (CAN_SR) .....	571
25.6.5 CAN interrupt register (CAN_IR) .....	573
25.6.6 CAN interrupt enable register (CAN_IER) .....	575
25.6.7 CAN acceptance code register group 0 (GROUP0_ACR) .....	576
25.6.8 CAN acceptance mask register group 0 (GROUP0_AMR).....	577
25.6.9 CAN bus timing register 0 (CAN_BTR0).....	578
25.6.10 CAN bus timing 1 (CAN_BTR1) .....	578
25.6.11 CAN transmit identifier register 0 (CAN_TXID0) .....	579
25.6.12 CAN transmit identifier register 1 (CAN_TXID1).....	580
25.6.13 CAN arbitration lost capture register (CAN_ALC).....	580
25.6.14 CAN error code capture register (CAN_ECC) .....	583
25.6.15 CAN error warning limit register (CAN_EWLR) .....	584
25.6.16 CAN RX error counter register (CAN_RXERR) .....	585
25.6.17 CAN TX error counter register (CAN_TXERR) .....	586
25.6.18 CAN standard frame format register (CAN_SFF) .....	587
25.6.19 CAN transmit identifier register 0 (CAN_TXID0).....	588
25.6.20 CAN transmit identifier register 1 (CAN_TXID1).....	589
25.6.21 CAN transmit data register 0 (CAN_TXDATA0).....	589
25.6.22 CAN transmit data register 1 (CAN_TXDATA1).....	590
25.6.23 CAN clock division register (CAN_CDR) .....	591
25.6.24 CAN filter mode register 0 (CAN_AFMO).....	591
25.6.25 CAN filter mode register 1 (CAN_AFMI).....	592
25.6.26 CAN filter mode register 2 (CAN_AFMO).....	592
25.6.27 CAN filter group enable register 0 (CAN_FGA0) .....	593
25.6.28 CAN filter group enable register 1 (CAN_FGA1) .....	593
25.6.29 CAN filter group enable register 2 (CAN_FGA2) .....	594
25.6.30 CAN acceptance code register group x (x = 1~19)(GROUPx_ACR) .....	594
25.6.31 CAN acceptance mask register group x (x = 1~19)(GROUPx_AMR) .....	595

<b>26 USB Full-Speed Device Interface (USB) .....</b>	597
26.1 USB introduction.....	597
26.2 USB main features.....	597
26.3 USB functional description .....	598
26.3.1 Description of USB functional modules .....	599
26.4 Programming considerations .....	599
26.4.1 General description of USB transfer.....	600
26.4.2 USB enumeration.....	602
26.4.3 USB transfer handling .....	604
26.4.4 IN token packets .....	605
26.4.5 OUT token packet .....	606
26.5 USB register description .....	606
26.5.1 USB TOP register (USB_TOP).....	607
26.5.2 USB interrupt state register (USB_INT_STATE).....	608
26.5.3 USB endpoint interrupt state register (EP_INT_STATE).....	609
26.5.4 USB endpoint 0 interrupt state register (EP0_INT_STATE).....	609
26.5.5 USB interrupt enable register (USB_INT_EN).....	611
26.5.6 USB endpoint interrupt enable register (EP_INT_EN).....	611
26.5.7 USB endpoint 0 interrupt enable register (EP0_INT_EN).....	612
26.5.8 USB endpoint X interrupt state register (EPX_INT_STATE)(X= 1~ 4) .....	613
26.5.9 USB endpoint X interrupt enable register (EPX_INT_EN) (X = 1 ~ 4) .....	615
26.5.10 USB address register (USB_ADDR) .....	615
26.5.11 USB endpoint enable register (EP_EN).....	616
26.5.12 USB endpoint DMA direction register (EP_DMA_DIR).....	616
26.5.13 USB endpoint type register (EP_TYPE).....	617
26.5.14 USB endpoint 12 index register (EP_INDEX1_2).....	618
26.5.15 USB endpoint 34 index register (EP_INDEX3_4).....	618
26.5.16 USB data toggle control register (TOG_CTRL1_4) .....	619
26.5.17 USB data toggle state register (TOG_STAT1_4) .....	619
26.5.18 USB setup data register (SETUPX) (X = 0 ~ 7) .....	620
26.5.19 USB packet size low register (PACKET_SIZEL).....	621
26.5.20 USB packet size high register (PACKET_SIZEH).....	622

26.5.21 USB endpoint X available data register (EPX_AVAIL) .....	622
26.5.22 USB endpoint 2 DMA address 0 register (DMA_ADDR0) .....	622
26.5.23 USB endpoint 2 DMA address 1 register (DMA_ADDR1) .....	623
26.5.24 USB endpoint 2 DMA address 2 register (DMA_ADDR2) .....	623
26.5.25 USB endpoint 2 DMA address 3 register (DMA_ADDR3) .....	624
26.5.26 USB endpoint 2 DMA number low register (DMA_NUML) .....	624
26.5.27 USB endpoint 2 DMA number high register (DMA_NUMH) .....	624
26.5.28 USB endpoint X control register (EPX_CTRL) .....	625
26.5.29 USB endpoint X FIFO register (EPX_FIFO) .....	625
26.5.30 USB endpoint data memory register (EP_MEM) .....	626
26.5.31 USB endpoint DMA enable register (EP_DMA) .....	626
26.5.32 USB endpoint halt register (EP_HALT) .....	627
26.5.33 USB power control register (USB_POWER) .....	627
26.5.34 USB AHB DMA register (USB_AHB_DMA) .....	628
26.5.35 USB AHB RST register (USB_AHB_RST) .....	629
<b>27 Clock Recovery System (CRS).....</b>	<b>630</b>
27.1 Introduction .....	630
27.2 CRS main features.....	630
27.3 CRS functional description.....	630
27.3.1 CRS block diagram .....	631
27.3.2 Synchronization input.....	631
27.3.3 Frequency error measurement.....	631
27.3.4 Frequency error evaluation and automatic trimming.....	632
27.3.5 CRS initialization and configuration.....	633
27.4 CRS low-power modes .....	633
27.5 CRS interrupts .....	634
27.6 CRS register description .....	634
27.6.1 CRS control register (CRS_CR).....	634
27.6.2 CRS configuration register (CRS_CFGR) .....	636
27.6.3 CRS interrupt status register (CRS_ISR) .....	637
27.6.4 CRS interrupt flag clear register (CRS_ICR) .....	640
<b>28 System Configuration Controller (SYSCFG) .....</b>	<b>642</b>

28.1 SYSCFG register description .....	642
28.1.1 SYSCFG configuration register (SYSCFG_CFGR) .....	642
28.1.2 External interrupt configuration register 1 (SYSCFG_EXTICR1) .....	645
28.1.3 External interrupt configuration register 2 (SYSCFG_EXTICR2) .....	645
28.1.4 External interrupt configuration register 3 (SYSCFG_EXTICR3) .....	646
28.1.5 External interrupt configuration register 4 (SYSCFG_EXTICR4) .....	646
28.1.6 PAD configuration register (SYSCFG_PADHYS) .....	647
<b>29 Device Electronic Signature (Device)</b> .....	648
29.1 Memory size registers .....	648
29.1.1 Unique device ID register (96 bits) .....	648
29.2 UID register description .....	648
29.2.1 Unique ID (UID1) .....	649
29.2.2 Unique ID (UID2) .....	649
29.2.3 Unique ID (UID3) .....	649
29.2.4 Unique ID (UID4) .....	650
<b>30 Debug Support (DBG)</b> .....	651
30.1 Overview .....	651
30.2 Pinout and debug port pins .....	652
30.2.1 SWD debug port pins .....	652
30.2.2 Internal pull-up and pull-down on SWD pins .....	652
30.3 ID codes and locking mechanism .....	652
30.3.1 MCU device ID code .....	652
30.3.2 Cortex JEDEC-106 ID code .....	653
30.4 SW debug port .....	653
30.4.1 SW protocol introduction .....	653
30.4.2 SW protocol sequence .....	654
30.4.3 SW-DP state machine (Reset, idle states, ID code) .....	655
30.4.4 DP and AP read/write accesses .....	655
30.4.5 SW-DP registers .....	655
30.4.6 SW-AP registers .....	656
30.5 MCU debug component (MCUDBG) .....	656
30.5.1 Debug support for low-power modes .....	656

30.5.2 Debug support for timers and watchdog.....	657
30.5.3 Debug MCU configuration register .....	657
30.6 DBG register description.....	657
30.6.1 DBG control register (DBG_CR).....	657
<b>31 Revision History.....</b>	<b>660</b>

## List of Figures

Figure 1. System architecture .....	2
Figure 2. Programming procedures .....	14
Figure 3. Flash register page erase procedures .....	15
Figure 4. Flash register mass erase procedures .....	16
Figure 5. Option byte programming procedures .....	17
Figure 6. Option byte erase procedure.....	18
Figure 7. CRC calculation unit block diagram .....	29
Figure 8. CSM block diagram .....	38
Figure 9. CSM block diagram .....	39
Figure 10. Power supply block diagram .....	48
Figure 11. Power-on reset/power-down reset waveform .....	49
Figure 12. PVD thresholds .....	50
Figure 13. RTC block diagram .....	65
Figure 14. RTC second and alarm waveform example with PR=0003, ALARM=00004 .....	67
Figure 15. RTC Overflow waveform example with PR=0003 .....	67
Figure 16. Reset circuit.....	76
Figure 17. Clock tree.....	77
Figure 18. Clock source.....	78
Figure 19. Basic structure of an I/O port bit.....	107
Figure 20. Input floating/pull up/pull down configurations .....	109
Figure 21. Output configuration .....	110
Figure 22. Alternate function configuration .....	111
Figure 23. High impedance-analog input configuration .....	112
Figure 24. External interrupt/event controller block diagram .....	126
Figure 25. External interrupt GPIO mapping .....	128
Figure 26. DMA block diagram .....	134
Figure 27. ADC system block diagram .....	148
Figure 28. ADC block diagram.....	149
Figure 29. Single conversion mode timing diagram .....	150

Figure 30. Enable channel conversion timing diagram in One-cycle scan mode (channel direction: from low to high) .....	151
Figure 31. Enable channel conversion timing diagram in One-cycle scan mode (channel direction: from high to low) .....	152
Figure 32. Enable channel conversion timing diagram in continuous scan mode (channel direction: from low to high) .....	153
Figure 33. Enable channel conversion timing diagram in continuous scan mode (channel direction: from high to low) .....	153
Figure 34. Channel conversion timing diagram in Single conversion mode.....	154
Figure 35. Channel conversion timing diagram In One-cycle scan mode .....	155
Figure 36. Channel conversion timing diagram in Continuous scan mode .....	156
Figure 37. Timing diagram with dynamically updated configuration, in Continuous scan mode .....	156
Figure 38. Data alignment style .....	157
Figure 39. Comparator block diagrams .....	174
Figure 40. Comparator hysteresis .....	176
Figure 41. Advanced-control timer block diagram .....	183
Figure 42. Counter timing diagram with prescaler division change from 1 to 2 .....	184
Figure 43. Counter timing diagram with prescaler division change from 1 to 4 .....	185
Figure 44. Counter timing diagram, internal clock divided by 1 .....	186
Figure 45. Counter timing diagram, internal clock divided by 2.....	186
Figure 46. Counter timing diagram, internal clock divided by 4.....	186
Figure 47. Counter timing diagram, internal clock divided by N .....	187
Figure 48. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) .....	187
Figure 49. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) .....	188
Figure 50. Counter timing diagram, internal clock divided by 1 .....	189
Figure 51. Counter timing diagram, internal clock divided by 2 .....	189
Figure 52. Counter timing diagram, internal clock divided by 4.....	189
Figure 53. Counter timing diagram, internal clock divided by N .....	190
Figure 54. Counter timing diagram, update event when repetition counter is not used .....	190
Figure 55. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 .....	191
Figure 56. Counter timing diagram, internal clock divided by 2.....	192
Figure 57. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x03 .....	192

Figure 58. Counter timing diagram, internal clock divided by N .....	193
Figure 59. Counter timing diagram, update event with ARPE=1 (counter underflow) .....	193
Figure 60. Counter timing diagram, Update event with ARPE=1 (counter overflow) .....	194
Figure 61. Update rate examples depending on mode and TIMx_RCR register settings.....	195
Figure 62. Control circuit in normal mode, internal clock divided by 1 .....	196
Figure 63. TI2 external clock connection example .....	196
Figure 64. Control circuit in external clock mode 1 .....	197
Figure 65. External trigger input block.....	197
Figure 66. Control circuit in external clock mode 2 .....	198
Figure 67. Capture/compare channel (example: channel 1 input stage).....	199
Figure 68. Capture/compare channel 1 main circuit.....	199
Figure 69. Output stage of capture/compare channel (channel 1 to 3).....	200
Figure 70. Output stage of capture/compare channel (channel 4).....	200
Figure 71. PWM input mode timing.....	202
Figure 72. Output compare mode, toggle on OC1 .....	203
Figure 73. Edge-aligned PWM waveforms (ARR = 8) .....	205
Figure 74. Center-aligned PWM waveforms (ARR = 8).....	206
Figure 75. Phase shift diagram.....	207
Figure 76. Complementary output with dead-time insertion .....	208
Figure 77. Dead-time waveforms with delay greater than the negative pulse .....	208
Figure 78. Dead-time waveforms with delay greater than the positive pulse .....	209
Figure 79. Output behavior in response to a break.....	211
Figure 80. Clearing TIMx OCxREF .....	212
Figure 81. 6-step generation, COM example (OSSR=1).....	213
Figure 82. Example of one pulse mode.....	214
Figure 83. Example of counter operation in encoder interface mode.....	216
Figure 84. Example of encoder interface mode with IC1FP1 polarity inverted .....	217
Figure 85. Example of Hall sensor interface.....	218
Figure 86. Control circuit in reset mode .....	219
Figure 87. Control circuit in gated mode .....	220
Figure 88. Control circuit in trigger mode.....	220

Figure 89. Control circuit in external clock mode 2 + trigger mode.....	221
Figure 90. General-purpose timer block diagram .....	258
Figure 91. Counter timing diagram with prescaler division change from 1 to 2 .....	259
Figure 92. Counter timing diagram with prescaler division change from 1 to 4 .....	260
Figure 93. Counter timing diagram, internal clock divided by 1 .....	261
Figure 94. Counter timing diagram, internal clock divided by 2.....	261
Figure 95. Counter timing diagram, internal clock divided by 4.....	261
Figure 96. Counter timing diagram, internal clock divided by N .....	262
Figure 97. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) .....	262
Figure 98. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) .....	263
Figure 99. Counter timing diagram, internal clock divided by 1 .....	264
Figure 100. Counter timing diagram, internal clock divided by 2 .....	264
Figure 101. Counter timing diagram, internal clock divided by 4 .....	264
Figure 102. Counter timing diagram, internal clock divided by N.....	265
Figure 103. Counter timing diagram, update event when repetition counter is not used .....	265
Figure 104. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6.....	266
Figure 105. Counter timing diagram, internal clock divided by 2 .....	267
Figure 106. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x03.....	267
Figure 107. Counter timing diagram, internal clock divided by N.....	268
Figure 108. Counter timing diagram, update event with ARPE=1 (counter underflow) .....	268
Figure 109. Counter timing diagram, Update event with ARPE=1 (counter overflow) .....	269
Figure 110. Control circuit in normal mode, internal clock divided by 1 .....	270
Figure 111. TI2 external clock connection example.....	270
Figure 112. Control circuit in external clock mode 1.....	271
Figure 113. External trigger input block .....	271
Figure 114. Control circuit in external clock mode 2.....	272
Figure 115. Capture/compare channel (example: channel 1 input stage) .....	273
Figure 116. Capture/compare channel 1 main circuit.....	273
Figure 117. Output stage of capture/compare channel (channel 1).....	274
Figure 118. PWM input mode timing.....	276
Figure 119. Output compare mode, toggle on OC1.....	277

Figure 120. Edge-aligned PWM waveforms (ARR = 8) .....	278
Figure 121. Center-aligned PWM waveforms (ARR = 8) .....	279
Figure 122. Example of one pulse mode .....	280
Figure 123. Clearing TIMx OCxREF .....	282
Figure 124. Example of counter operation in encoder interface mode .....	284
Figure 125. Example of encoder interface mode with IC1FP1 polarity inverted.....	284
Figure 126. Control circuit in reset mode .....	285
Figure 127. Control circuit in gated mode .....	286
Figure 128. Control circuit in trigger mode.....	287
Figure 129. Control circuit in external clock mode 2 + trigger mode .....	288
Figure 130. Master/Slave timer example .....	288
Figure 131. Gating timer 2 with OC1REF of timer 1.....	289
Figure 132. Gating timer 2 with enable of timer 1.....	290
Figure 133. Triggering timer 2 with update of timer 1.....	291
Figure 134. Triggering timer 2 with enable of timer 1.....	291
Figure 135. Triggering timer 1 and 2 with timer 1 TI1 input.....	292
Figure 136. General-purpose timer block diagram.....	324
Figure 137. Counter timing diagram with prescaler division change from 1 to 2.....	325
Figure 138. Counter timing diagram with prescaler division change from 1 to 4.....	326
Figure 139. Counter timing diagram, internal clock divided by 1 .....	327
Figure 140. Counter timing diagram, internal clock divided by 2 .....	327
Figure 141. Counter timing diagram, internal clock divided by 4 .....	327
Figure 142. Counter timing diagram, internal clock divided by N.....	328
Figure 143. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) .....	328
Figure 144. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) ....	329
Figure 145. Counter timing diagram, internal clock divided by 1 .....	330
Figure 146. Counter timing diagram, internal clock divided by 2 .....	330
Figure 147. Counter timing diagram, internal clock divided by 4 .....	330
Figure 148. Counter timing diagram, internal clock divided by N.....	331
Figure 149. Counter timing diagram, update event when repetition counter is not used .....	331

Figure 150. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6.....	332
Figure 151. Counter timing diagram, internal clock divided by 2 .....	333
Figure 152. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36.....	333
Figure 153. Counter timing diagram, internal clock divided by N.....	334
Figure 154. Counter timing diagram, update event with ARPE=1 (counter underflow) .....	334
Figure 155. Counter timing diagram, Update event with ARPE=1 (counter overflow) .....	335
Figure 156. Control circuit in normal mode, internal clock divided by 1 .....	336
Figure 157. TI2 external clock connection example .....	336
Figure 158. Control circuit in external clock mode 1 .....	337
Figure 159. External trigger input block.....	337
Figure 160. Control circuit in external clock mode 2 .....	338
Figure 161. Capture/compare channel (example: channel 1 input stage).....	339
Figure 162. Capture/compare channel 1 main circuit .....	339
Figure 163. Output stage of capture/compare channel (channel 1) .....	340
Figure 164. Output stage of capture/compare channel (channel 1) .....	342
Figure 165. Output compare mode, toggle on OC1 .....	343
Figure 166. Edge-aligned PWM waveforms (ARR = 8) .....	344
Figure 167. Center-aligned PWM waveforms (ARR = 8) .....	345
Figure 168. Example of one pulse mode .....	346
Figure 169. Clearing TIMx OCxREF .....	348
Figure 170. Example of counter operation in encoder interface mode.....	350
Figure 171. Example of encoder interface mode with IC1FP1 polarity inverted.....	350
Figure 172. Control circuit in reset mode .....	351
Figure 173. Control circuit in gated mode .....	352
Figure 174. Control circuit in trigger mode.....	353
Figure 175. Control circuit in external clock mode 2 + trigger mode.....	354
Figure 176. Master/Slave timer example .....	354
Figure 177. Gating timer 2 with OC1REF of timer 1.....	355
Figure 178. Gating timer 2 with enable of timer 1.....	356
Figure 179. Triggering timer 2 with update of timer 1.....	357
Figure 180. Triggering timer 2 with enable of timer 1.....	357

Figure 181. Triggering timer 1 and 2 with timer 1 TI1 input.....	358
Figure 182. Basic timer block diagram .....	387
Figure 183. Counter timing diagram with prescaler division change from 1 to 2.....	388
Figure 184. Counter timing diagram with prescaler division change from 1 to 4.....	388
Figure 185. Counter timing diagram, internal clock divided by 1 .....	389
Figure 186. Counter timing diagram, internal clock divided by 2 .....	390
Figure 187. Counter timing diagram, internal clock divided by 4 .....	390
Figure 188. Counter timing diagram, internal clock divided by N.....	390
Figure 189. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) .....	391
Figure 190. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) ....	391
Figure 191. Control circuit in normal mode, internal clock divided by 1.....	392
Figure 192. Capture/compare channel (example: channel 1 input stage).....	392
Figure 193. Capture/compare channel 1 main circuit .....	393
Figure 194. Output stage of capture/compare channel (channel 1) .....	393
Figure 195. Output compare mode, toggle on OC1 .....	396
Figure 196. Edge-aligned PWM waveforms (ARR = 8) .....	397
Figure 197. Basic timer TiM16 and TIM17 block diagram .....	411
Figure 198. Counter timing diagram with prescaler division change from 1 to 2.....	412
Figure 199. Counter timing diagram with prescaler division change from 1 to 4.....	413
Figure 200. Counter timing diagram, internal clock divided by 1 .....	413
Figure 201. Counter timing diagram, internal clock divided by 2 .....	414
Figure 202. Counter timing diagram, internal clock divided by 4 .....	414
Figure 203. Counter timing diagram, internal clock divided by N.....	414
Figure 204. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) .....	415
Figure 205. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) ....	415
Figure 206. Control circuit in normal mode, internal clock divided by 1.....	416
Figure 207. Capture/compare channel (example: channel 1 input stage).....	416
Figure 208. Capture/compare channel 1 main circuitFigure 209. Output stage of capture/compare channel (channel 1).....	417
Figure 210. Output compare mode, toggle on OC1 .....	420

Figure 211. Edge-aligned PWM waveforms (ARR = 8) .....	421
Figure 212. Complementary output with dead-time insertion .....	422
Figure 213. Dead-time waveforms with delay greater than the negative pulse .....	422
Figure 214. Dead-time waveforms with delay greater than the positive pulse.....	422
Figure 215. Output behavior in response to a break.....	424
Figure 216. Example of one pulse mode .....	425
Figure 217. Independent watchdog block diagram.....	447
Figure 218. Watchdog block diagram .....	455
Figure 219. Window watchdog timing diagram .....	456
Figure 220. SPI block diagram .....	461
Figure 221. Single master/single slave application .....	462
Figure 222. Data clock timing diagram.....	464
Figure 223. Start and Stop conditions .....	481
Figure 224. 7-bit address format .....	481
Figure 225. 10-bit address format .....	482
Figure 226. Master transmit protocol .....	483
Figure 227. Master receive protocol .....	483
Figure 228. Start Byte transfer.....	484
Figure 229. DR register.....	485
Figure 230. Master transmit - empty Tx FIFO .....	485
Figure 231. Master receive - empty Tx FIFO.....	485
Figure 232. Multiple master arbitration.....	486
Figure 233. Multiple master clock synchronization.....	487
Figure 234. SCL master clock generation.....	487
Figure 235. SCL master clock synchronization.....	488
Figure 236. I2C function block diagram.....	489
Figure 237. I2C interface master flow chart.....	492
Figure 238. Interrupt mechanism .....	494
Figure 239. UART block diagram .....	515
Figure 240. UART timing .....	516
Figure 241. Status bit change during transmission .....	517

Figure 242. UART block diagram .....	521
Figure 243. UART block diagram .....	522
Figure 244. RX pin sampling scheme.....	523
Figure 245. CAN network topology.....	541
Figure 246. CAN structure block diagram.....	542
Figure 247. Example of CAN identifier acceptance.....	551
Figure 248. Single filter configuration, receiving standard frame messages.....	552
Figure 249. Single filter configuration, receiving extended frame messages.....	553
Figure 250. Dual filter configuration, receiving standard frame messages.....	554
Figure 251. Dual filter configuration, receiving extended frame messages.....	555
Figure 252. Transmit buffer layout for standard and extended frame format configurations ....	559
Figure 253. Transmit buffer layout for standard and extended frame format configurations ....	560
Figure 254. Example for the Error Code Capture Function.....	561
Figure 255. Example of arbitration lost bit number interpretation .....	565
Figure 256. USB peripheral block diagram.....	598
Figure 257. Basic format of a packet.....	600
Figure 258. USB transaction.....	601
Figure 259. USB transfer.....	601
Figure 260. Enumeration process .....	602
Figure 261. Flow chart of USB transfer .....	605
Figure 262. CRS block diagram .....	631
Figure 263. CRS counter behavior.....	632
Figure 264. Block diagram of MM32 series and CPU level debug support.....	651

## List of Tables

Table 1. Memory mapping.....	3
Table 2. Boot modes .....	8
Table 3. Flash module organization.....	10
Table 4. Flash memory read protection status.....	19
Table 5. Flash memory read unprotection status.....	20
Table 6. Flash interrupt request.....	20
Table 7. Option byte format.....	21
Table 8. Option byte organization.....	21
Table 9. Option bytes description .....	22
Table 10. Flash register overview.....	23
Table 11. Overview of CRC registers .....	30
Table 12. Overview of hardware division registers .....	33
Table 13. Baud rate formula.....	39
Table 14. Overview of CSM register description.....	40
Table 15. Low-power mode summary .....	51
Table 16. SLEEP NOW mode .....	52
Table 17. SLEEP ON EXIT mode.....	52
Table 18. Stop mode .....	53
Table 19. Standby mode .....	54
Table 20. Overview of power control registers.....	54
Table 21. Overview of BKP registers.....	59
Table 22. Overview of RTC registers.....	68
Table 23. Overview of RCC registers .....	81
Table 24. Port bit configuration table.....	107
Table 25. Output MODE bits .....	107
Table 26. Advanced timer TIM1.....	112
Table 27. General-purpose timers TIM2/3/14/16/17 .....	112
Table 28. UART .....	112
Table 29. SPI .....	113
Table 30. I2C .....	113
Table 31. CAN .....	113

Table 32. ADC.....	113
Table 33. Other IOs.....	114
Table 34. Debug port signal .....	115
Table 35. Overview of GPIO registers .....	115
Table 36. Vector table for this product series.....	123
Table 37. Overview of EXTI registers .....	129
Table 38. Programmable data transfer width & endian behavior (when bits PINC = MINC = 1) .....	136
Table 39. DMA interrupt requests.....	139
Table 40. Summary of the DMA requests for each channel.....	139
Table 41. Overview of DMA registers .....	140
Table 42. Overview of ADC registers.....	159
Table 43. Overview of COMP registers .....	176
Table 44 Counting direction versus encoder signals .....	215
Table 45. Overview of TIM1 registers.....	221
Table 46. TIMx internal trigger connection.....	230
Table 47. Output control bitts for complementary OCx and OCxN channels with break feature .....	243
Table 48. Counting direction versus encoder signals .....	283
Table 49. Overview of TIMx registers .....	294
Table 50. TIMx internal trigger connection.....	301
Table 51. Output control bit for standard Ocx channels.....	317
Table 52. Counting direction versus encoder signals .....	349
Table 53. Overview of TIMx registers .....	360
Table 54. TIMx internal trigger connection.....	367
Table 55. Output control bit for standard Ocx channels.....	379
Table 56. Overview of TIM14 registers.....	397
Table 57. Output control bit for standard OCx channels .....	407
Table 58. Overview of TIM16/17 registers .....	426
Table 59. Output control bits for complementary OCx and OCxN channels with break feature .....	430
Table 60. IWDG timeout period (input clock at 40 KHz (LSI)).....	447
Table 61. Overview of IWDG registers .....	448

Table 62. Overview of WWDG registers.....	457
Table 63. SPI status.....	466
Table 64. Baud rate formula.....	467
Table 65. Overview of SPI registers .....	468
Table 66. First bytes for I2C .....	482
Table 67. Setting and clearing the interrupt bits .....	493
Table 68. Overview of I2C register description .....	494
Table 69. DISSLAVE(bit 6) and MASTER(bit 0) settings .....	498
Table 70. UART interrupt requests .....	524
Table 71. Overview of UART registers .....	525
Table 72. Basic CAN register permission allocation .....	544
Table 73. Peli CAN register permission allocation .....	547
Table 74. Rx- and Tx-Buffer in BasicCAN mode.....	558
Table 75. Possible errors during reception .....	562
Table 76. Possible errors during transmission .....	562
Table 77. Overview of CAN registers .....	566
Table 78. Function of bits 4 to 0 of the arbitration lost capture register.....	581
Table 79. Overview of USB registers.....	606
Table 80. Effect of low-power modes on CRS .....	633
Table 81. Interrupt control bits.....	634
Table 82. Overview of CRS registers .....	634
Table 83. Overview of SYSCFG registers .....	642
Table 84. Overview of memory size register description.....	648
Table 85. SWJ debug port pins .....	652
Table 87. ID codes .....	653
Table 88. Packet request (8-bits) .....	654
Table 89. ACK response (3 bits).....	654
Table 90. DATA transfer (33 bits).....	654
Table 92. Overview of DBG registers .....	657
Table 93. Revision history .....	660

# 1 | Memory and Bus Architecture

## Memory and Bus Architecture

### 1.1 System architecture

The main system consists of:

- Two masters:
  - CPU core system bus (S-bus)
  - General-purpose DMA
- Three slaves:
  - Internal SRAM
  - Internal Flash memory
  - AHB to APB bridges (APBx), which connect all the APB peripherals

These are interconnected using a multilayer AHB bus architecture as shown in Figure 1:

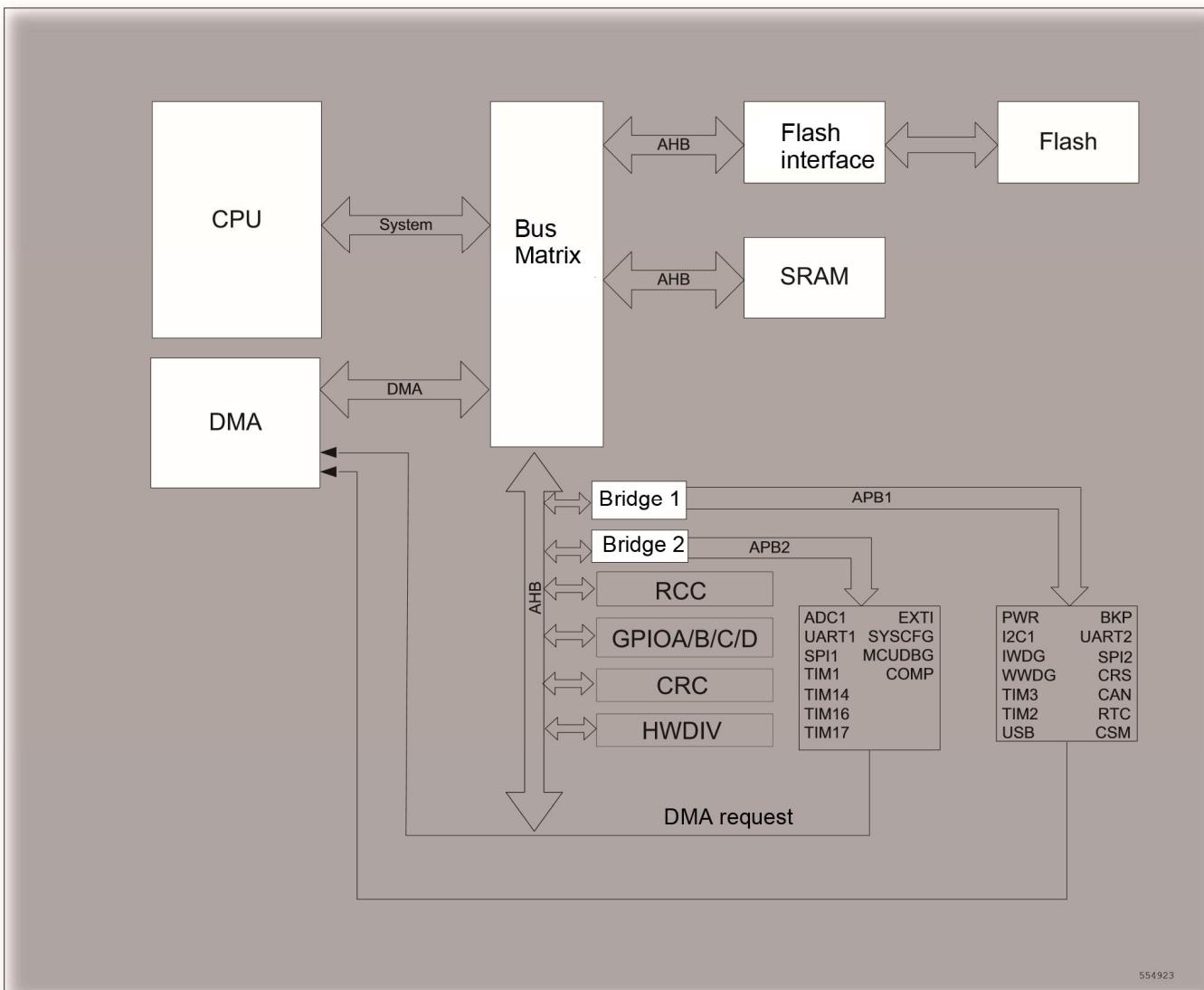


Figure 1. System architecture

### System bus

This bus connects the system bus of the CPU core (peripheral bus) to a BusMatrix which manages the arbitration between the core and the DMA.

### DMA bus

This bus connects the AHB master interface of the DMA to the BusMatrix which manages the access of CPU and DMA to SRAM, Flash memory and peripherals.

### BusMatrix

The BusMatrix manages the access arbitration between the core system bus and the DMA bus. The BusMatrix is composed of the master module bus and the slave module bus.

AHB peripherals are connected on system bus through a BusMatrix to allow DMA access.

### AHB2APB bridges - APB

The AHB to APB bridges provide full synchronous connections between the AHB and the APB buses. After each device reset, all peripheral clocks are disabled (except for the SRAM and Flash). Before using a peripheral you have to enable its clock in the RCC\_AHBENR, RCC\_APB2ENR or RCC\_APB1ENR register.

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

## 1.2 Memory organization

### 1.2.1 Introduction

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

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

The addressable memory space is divided into 8 main blocks, each of 512MB. All the other memory areas that are not allocated to on-chip memories and peripherals are considered "Reserved" address space. For details, refer to the memory mapping and register addressing and the peripheral sections.

### 1.2.2 Memory mapping and register addressing

See the datasheet corresponding to each peripheral section for a comprehensive diagram of the memory mapping.

The table below gives the start addresses of all built-in peripherals.

Table 1. Memory mapping

Bus	Boundary Address	Size	Peripheral	Remark
Flash	0x0000 0000–0x0000 FFFF	64KB	Main Flash memory, system memory or SRAM, depending on the BOOT configuration	
	0x0001 0000–0x07FF FFFF	~ 128 MB	Reserved	
	0x0800 0000–0x0800 FFFF	64 KB	Main Flash memory	
	0x0801 0000–0x1FFD FFFF	~ 383 MB	Reserved	
	0x1FFE 0000–0x1FFE 01FF	0.5 KB	Reserved	
	0x1FFE 0200–0x1FFE 0FFF	3 KB	Reserved	
	0x1FFE 1000–0x1FFE 11FF	0.5 KB	Reserved	
	0x1FFE 1200–0x1FFE 1BFF	2.5 KB	Confidential space	

<b>Bus</b>	<b>Boundary Address</b>	<b>Size</b>	<b>Peripheral</b>	<b>Remark</b>
SRAM	0x1FFE 1C00–0x1FFF F3FF	~ 256 MB	Reserved	
	0x1FFF F400–0x1FFF F7FF	1 KB	System memory	
	0x1FFF F800–0x1FFF F80F	16 B	Option bytes	
	0x1FFF F810–0x1FFF FFFF	~2 KB	Reserved	
APB1	0x2000 0000–0x2000 3FFF	16 KB	SRAM	
	0x2000 4000–0x2FFF FFFF	~ 255 MB	Reserved	
APB1	0x4000 0000–0x4000 03FF	1 KB	TIM2	
	0x4000 0400–0x4000 07FF	1 KB	TIM3	
	0x4000 0800–0x4000 0BFF	8 KB	Reserved	
	0x4000 2800–0x4000 2BFF	1 KB	RTC/BKP	
	0x4000 2C00–0x4000 2FFF	1 KB	WWDG	
	0x4000 3000–0x4000 33FF	1 KB	IWDG	
	0x4000 3400–0x4000 37FF	1 KB	Reserved	
	0x4000 3800–0x4000 3BFF	1 KB	SPI2	

Bus	Boundary Address	Size	Peripheral	Remark
APB1	0x4000 4000–0x4000 43FF	1 KB	Reserved	
	0x4000 4400–0x4000 47FF	1 KB	UART2	UM_MM32F013x_Ver1.00
	0x4000 4800–0x4000 4BFF	3 KB	Reserved	
	0x4000 5400–0x4000 57FF	1 KB	I2C	
	0x4000 5800–0x4000 5BFF	1 KB	Reserved	
	0x4000 5C00–0x4000 5FFF	1 KB	USB	
	0x4000 6000–0x4000 63FF	1 KB	Reserved	
	0x4000 6400–0x4000 67FF	1 KB	CAN	
	0x4000 6800–0x4000 6BFF	1 KB	CSM	
	0x4000 6C00–0x4000 6FFF	1 KB	Reserved	
APB2	0x4000 7000–0x4000 73FF	1 KB	PWR	
	0x4000 7400–0x4000 FFFF	35 KB	Reserved	
	0x4001 0000–0x4001 03FF	1 KB	SYSCFG	
	0x4001 0400–0x4001 07FF	1 KB	EXTI	
	0x4001 0800–0x4001 23FF	7 KB	Reserved	
	0x4001 2400–0x4001 27FF	1 KB	ADC	
	0x4001 2800–0x4001 2BFF	1 KB	Reserved	
	0x4001 2C00–0x4001 2FFF	1 KB	TIM1	
	0x4001 3000–0x4001 33FF	1 KB	SPI1	
	0x4001 3400–0x4001 37FF	1 KB	DBGMCU	
AHB	0x4001 3800–0x4001 3BFF	1 KB	UART1	
	0x4001 3C00–0x4001 3FFF	1 KB	COMP	
	0x4001 4000–0x4001 43FF	1 KB	TIM14	
	0x4001 4400–0x4001 47FF	1 KB	TIM16	
	0x4001 4800–0x4001 4BFF	1 KB	TIM17	
	0x4001 4C00–0x4001 7FFF	13 KB	Reserved	
AHB	0x4002 0000–0x4002 03FF	1 KB	DMA	
	0x4002 0400–0x4002 0FFF	3 KB	Reserved	

0x4002 1000–0x4002 13FF	1 KB	RCC	
0x4002 1400–0x4002 1FFF	3 KB	Reserved	
0x4002 2000–0x4002 23FF	1 KB	Flash Interface	
0x4002 2400–0x4002 2FFF	3 KB	Reserved	
0x4002 3000–0x4002 33FF	1 KB	CRC	
0x4002 3400–0x4002 FFFF	47 KB	Reserved	
0x4003 0000–0x4003 03FF	1 KB	HWDIV	
0x4003 0400–0x47FF FFFF	~ 127 MB	Reserved	
0x4800 0000–0x4800 03FF	1 KB	GPIOA	
0x4800 0400–0x4800 07FF	1 KB	GPIOB	
0x4800 0800–0x4800 0BFF	1 KB	GPIOC	
0x4800 0C00–0x4800 0FFF	1 KB	GPIOD	
0x4800 1000–0x5FFF FFFF	~ 384 MB	Reserved	



## 1.3 Embedded SRAM

The device features up to 16K bytes of a static SRAM. It can be accessed as bytes (8 bits), half-words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000.

- The SRAM on the data bus reaches up to 16K bytes. It can be accessed by the CPU or DMA through the fastest system clock without inserting any latency.

## 1.4 Flash memory overview

The Flash memory has two different storage areas:

- Main Flash memory block, which includes the application and the user data zones (if necessary)
- Information block, which includes four parts:
  - Option bytes — containing the hardware and storage protection user configuration options.
  - System memory — containing the boot loader code. Refer to the embedded Flash memory section.

The Flash memory interface executes commands and data access based on the AHB protocol. Its prefetch buffer function enables the acceleration of the code execution by the CPU.

## 1.5 Boot configuration

In the chip, three different boot modes can be selected through BOOT0 pin level status and nBOOT1 bit configuration, as shown in the following table.

Table 2. Boot modes

Boot mode selection		Boot mode	Description
nBOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

The value of BOOT0 pin is latched on the 4th rising edge of SYSCLK after a reset. It is up to the user to set nBOOT1 bit in conjunction with the BOOT0 pin values to select the boot mode.

The BOOT0 pin value and the nBOOT1 bit are also re-sampled when exiting from Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode.

After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory starting from 0x0000 0004.

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

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF F400).
- Boot from the embedded SRAM: SRAM is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x2000 0000).

## **Embedded boot loader**

The boot loader is located in the system memory. It is used to reprogram the Flash memory through UART1.

## 2 | Embedded Flash Memory (FLASH)

### Embedded Flash Memory (FLASH)

#### 2.1 Key Flash features

- The Flash memory is up to 64K bytes
- Memory organization:
  - Main Flash memory block: Up to 16K bytes (16K × 32 bits)
  - Information block:
    - \* System memory: Up to 1K bytes (1K × 8 bits)
    - \* Option bytes: Up to 512 bits
    - \* Confidential space: Up to 2.5K bytes (2.5K × 8 bits)
    - \* Guard bytes: Up to 512 bytes (512 × 8 bits)

The Flash memory interface features:

- Read interface with prefetch buffer (2 x 64-bit words)
- Option byte loader
- Flash Program/Erase operation
- Read/Write protection
- Low-power mode

#### 2.2 Flash memory function description

##### 2.2.1 Flash memory organization

The Flash memory is organized as 64-bit wide memory cells that can be used for storing both code and data. The main Flash memory block is partitioned by 64 pages (1K bytes per page) or 16 sectors (4K bytes per sector), with write protection on a sector-by-sector basis (refer to the memory protection part).

Table 3. Flash module organization

Block	Name	Address	Size (bytes)
Main memory block	Page 0	0x0800 0000 - 0x0800 03FF	1K
	Page 1	0x0800 0400 - 0x0800 07FF	1K
	Page 2	0x0800 0800 - 0x0800 0BFF	1K
	Page 3	0x0800 0C00 - 0x0800 0FFF	1K
	...	...	...

Block	Name	Address	Size (bytes)
Main memory block	...	...	...
	Page 28	0x0800 7000 - 0x0800 73FF	1K
	Page 29	0x0800 7400 - 0x0800 77FF	1K
	Page 30	0x0800 7800 - 0x0800 7BFF	1K
	Page 31	0x0800 7C00 - 0x0800 7FFF	1K
	...	...	...
	...	...	...
	Page 60	0x0800 F000 - 0x0800 F3FF	1K
	Page 61	0x0800 F400 - 0x0800 F7FF	1K
	Page 62	0x0800 F800 - 0x0800 FBFF	1K
	Page 63	0x0800 FC00 - 0x0800 FFFF	1K
Information block	Guard bytes	0x1FFE 0000 - 0x1FFE 01FF	0.5K
	Confidential space	0x1FFE 1200 - 0x1FFE 1BFF	2.5K
	System memory ISP	0x1FFF F400 - 0x1FFF F7FF	1K
	Option bytes	0x1FFF F800 - 0x1FFF F9FF	0.5K
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

## 2.2.2 Read operation of the Flash memory

The embedded Flash module can be addressed directly as a common memory space. Any read operation that accesses the content of the Flash module should go through a dedicated judgmental process.

Instruction and the data fetches are performed through the AHB bus in the way that is specified as the option in the Flash access control register (FLASH\_ACR).

- Instruction fetch: CPU operation speed can be accelerated after enabling the prefetch buffer.
- Latency: Number of wait bits for a correct read operation.

### Instruction fetch

The CPU fetches the instructions over the AHB bus. The prefetch block aims at increasing the efficiency of instruction fetch.

## Prefetch buffer

Prefetch buffer (2 x 64-bit blocks): The buffer is enabled automatically after each reset. A whole block can be updated with a single read from the Flash memory as the size of each block (64 bits) matches the bandwidth of the Flash memory. Thanks to the prefetch buffer, faster CPU execution is possible as the CPU fetches one word up to 32 bits at a time with the next word readily available in the prefetch buffer.

## Prefetch controller

The prefetch controller decides to access the Flash memory depending on the available space in the prefetch buffer. The controller initiates a read request when there is at least one block free in the prefetch buffer. After reset, the prefetch buffer is on by default. The prefetch buffer must be switched on/off only when SYSCLK is lower than 24MHz and no prescaler is applied on the AHB clock (SYSCLK must be equal to HCLK). The prefetch buffer is usually switched on/off during the initialization routine, while the MCU is running on the internal 8MHz oscillator.

Note: The latency for prefetch buffer must be kept on when using a prescaler different from 1 on the AHB clock.

## Access latency

In order to maintain the correct reading from the Flash memory, the speed ratio of the prefetch controller has to be specified in the LATENCY[2:0] of the Flash access control register. This value gives the number of wait cycles needed to insert between each access of the Flash memory and next access. After reset, the value is zero by default, ie. no wait state is inserted.

### 2.2.3 Flash write and erase operations

The embedded Flash memory supports the in-circuit programming as well as the in-application programming.

The in-circuit programming (ICP) method refers to updating the contents of the Flash memory in the circuit by loading the user codes into the microcontroller. ICP offers a quick and efficient method eliminating unnecessary socketing of devices during chip programming.

In contrast to the ICP method, in-application programming (IAP) can use any communication interface supported by the MCU (I/Os, USB, UART, I2C, SPI, etc.) to download programs or data. IAP allows user to re-program the application while the application is running, provided that part of the application has to be programmed in the Flash memory with ICP/ISP in advance.

The write and erase operations can be performed within the whole operating voltage range of the device. They are performed by the following 7 registers:

- Key register (FLASH\_KEYR)
- Option byte key register (FLASH\_OPRKEYR)
- Flash control register (FLASH\_CR)
- Flash status register (FLASH\_SR)
- Flash address register (FLASH\_AR)
- Option byte register (FLASH\_OBR)
- Write protection register (FLASH\_WRPR)

An ongoing Flash memory write operation will not block the running of CPU as long as the CPU does not access the Flash memory. That is to say, during a write/erase operation to the Flash memory, any access to the Flash memory will stall the bus and will continue to proceed correctly once the write/erase operation has completed. This means that instruction or data fetches cannot be made while a write/erase operation is ongoing.

For write/erase operations on the Flash memory, the internal oscillator (HSI) must be ON.

## Unlocking the Flash memory

After reset, the Flash memory is protected by default to prevent unexpected erase operation. The FLASH\_CR register is not allowed to be rewritten, unless an unlocking sequence is executed for the FLASH\_CR register to enable access permission to the FLASH\_CR. This sequence consists of two write operations:

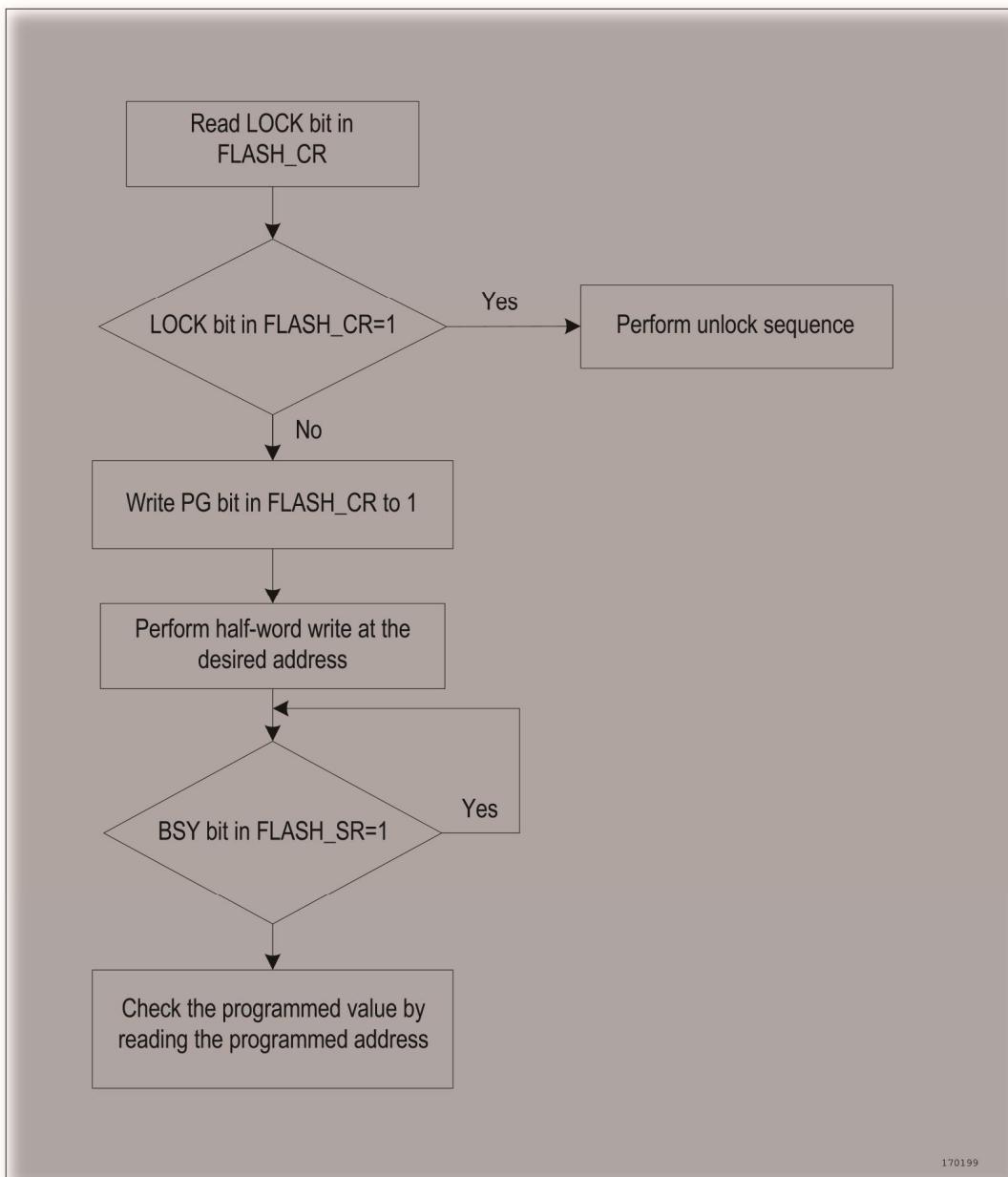
- Write Key 1 = 0x45670123
- Write Key 2 = 0xCDEF89AB

Any wrong sequence will lock up the FLASH\_CR register until the next reset.

Also a bus error is returned on a wrong key sequence, which will then trigger a hardware error interrupt. This is done immediately if KEY1 is wrong, or if KEY1 has been correctly written but KEY2 is wrong.

## Main Flash memory programming

The main Flash memory can be programmed 16 bits at a time. One programming operation is done when a half-word (16 bits) is written into the corresponding address with the PG bit of the FLASH\_CR to be 1. Any attempt to write word that are not half-word long will result in a hardware error interrupt.



170199

### Figure 2. Programming procedures

The Flash memory interface prefetches the bytes to be programmed and checks whether they are all set to 1. If not, the program operation is canceled and a programming error warning is issued by the PGERR bit in the FLASH\_SR register.

If the address to be programmed is write-protected by the FLASH\_WRPR register, the program operation will be canceled and a programming error warning will be issued. The end of the program operation is indicated by the EOP bit in the FLASH\_SR register.

The main Flash memory programming sequence in standard mode is as follows:

- Check the BSY bit in the FLASH\_SR register to confirm that the last operation has finished
- Set the PG bit in the FLASH\_CR register
- Perform the data write in a unit of half-word to desired address
- Wait for the BSY bit in the FLASH\_SR register to return to zero
- Read the data and verify

Note: These registers are not to be written when the BSY bit of the FLASH\_SR register is set to 1.

## Flash memory erase

The Flash memory erase operation can be performed at page level or on the whole Flash area (mass-erase).

### Page erase

To erase a page, the procedure below should be followed:

- Check the BSY bit in the FLASH\_SR register to confirm that the last operation has finished
- Set the PER bit to 1 in the FLASH\_CR register
- Write the FLASH\_AR register to select a page to erase
- Set the STRT bit to 1 in the FLASH\_CR register
- Wait for the BSY bit in the FLASH\_SR register to return to zero
- Read the erased page and verify

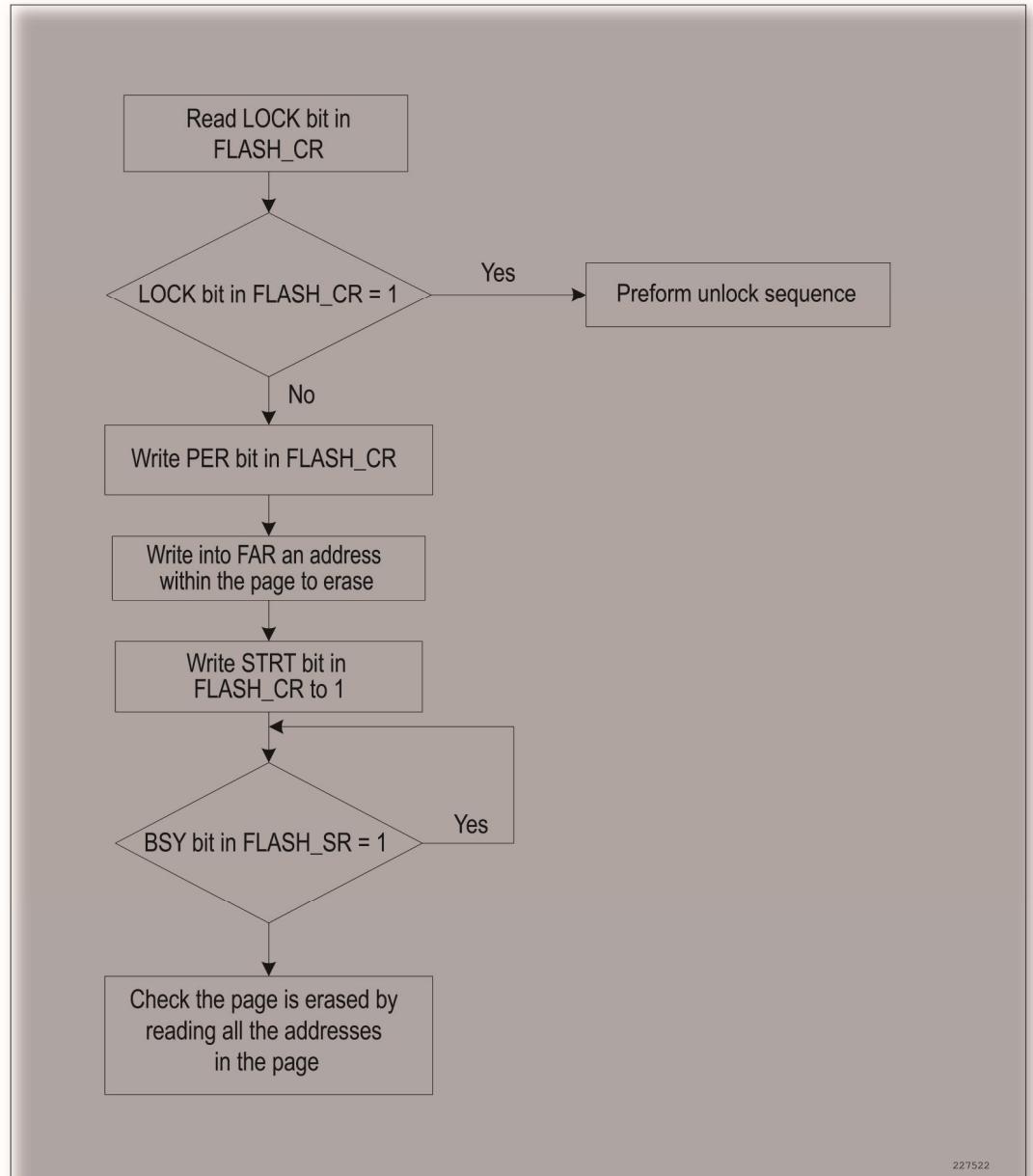


Figure 3. Flash register page erase procedures

## Mass Erase

The Mass Erase command can be used to erase the whole Flash user area. The information block is unaffected by this instruction. The specific procedure is as follows:

- Check the BSY bit in the FLASH\_SR register to confirm that the last operation has finished
- that last operation is finished by checking the BSY bit in the FLASH\_SR register
- Set the MER bit to 1 in the FLASH\_CR register
- Set the STRT bit to 1 in the FLASH\_CR register
- Wait for the BSY bit to return to zero
- Read all the pages and verify

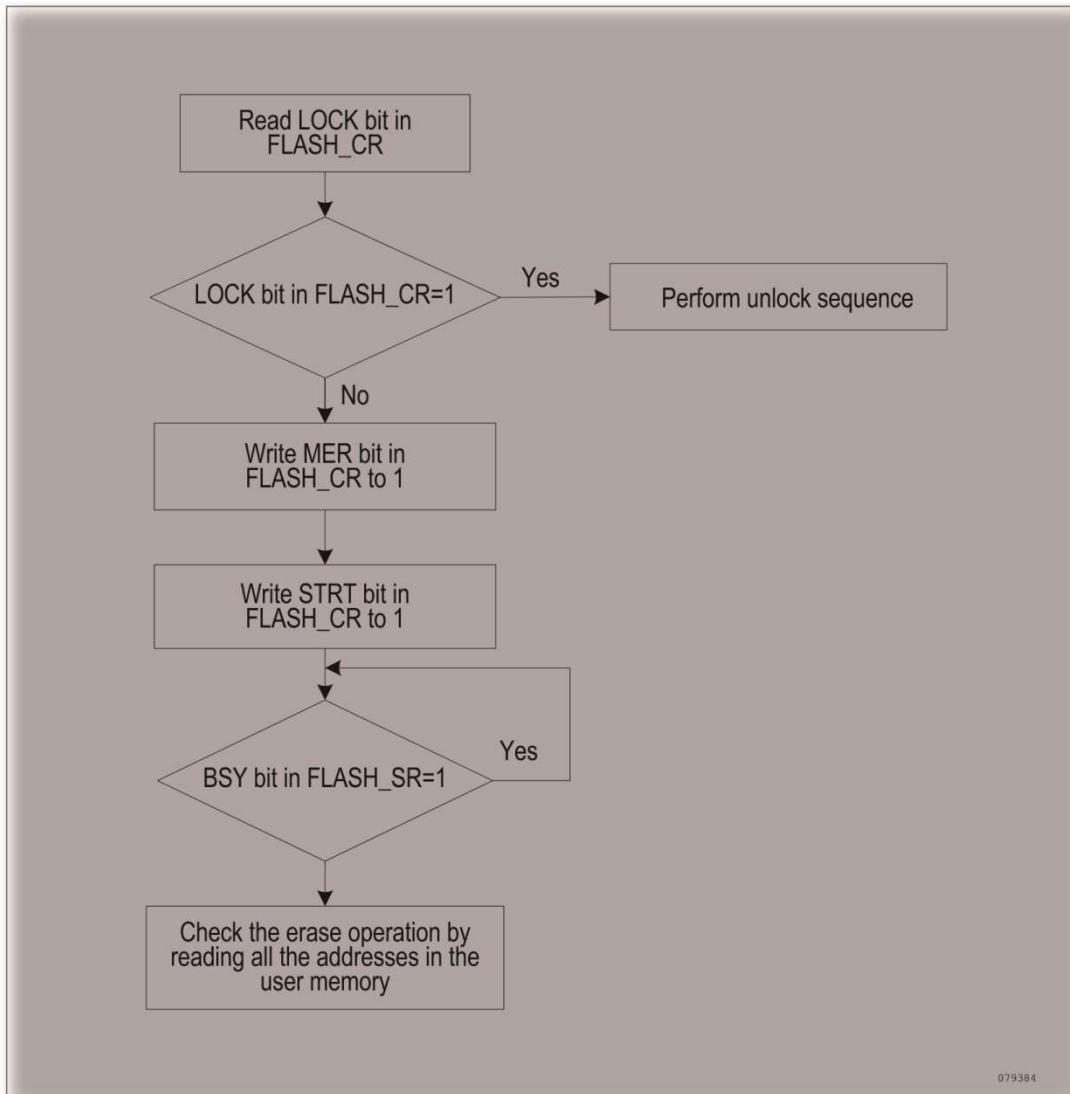


Figure 4. Flash register mass erase procedures

## Option byte programming

The option bytes are programmed differently from normal user addresses, including two write protection and one hardware configuration. After unlocking the access permission to the Flash, the writing of keys into the FLASH\_OPTKEYR register has to be performed. Once it is done, the OPTWRE bit in the FLASH\_CR register will be set to '1'. Then the user has to set the OPTPG bit in the FLASH\_CR register and perform a half-word write operation at the desired address. Similarly, the option bytes are checked to confirm they are set as 1. If not, the relevant operation will be canceled and an error warning will be issued by the PGERR bit in the FLASH\_SR register. The end of the program operation is indicated by the EOP bit in the FLASH\_SR register.

The option byte is 16-bit data. The significant data is the 8 lower bits, while the 8 higher bits are the complement of the 8 lower bits. The 8 higher bits are automatically set as the complement of 8 lower bits by hardware during the programming operation. This guarantees that the option byte value is always correct. The sequence is as follows:

- Check the BSY bit in the FLASH\_SR register to confirm that the last operation has finished
- Unlock the OPTWRE bit in the FLASH\_CR register
- Set the OPTPG bit to 1 in the FLASH\_CR register
- Write the data (half-word) to the desired address
- Wait for the BSY bit to return to zero

- Read the data and verify that a mass erase is automatically performed when the protection option byte is changed from protected to unprotected state.  
If the user wants to modify other bytes, then the mass erase is not performed. This mechanism is used to protect the content of the Flash memory.

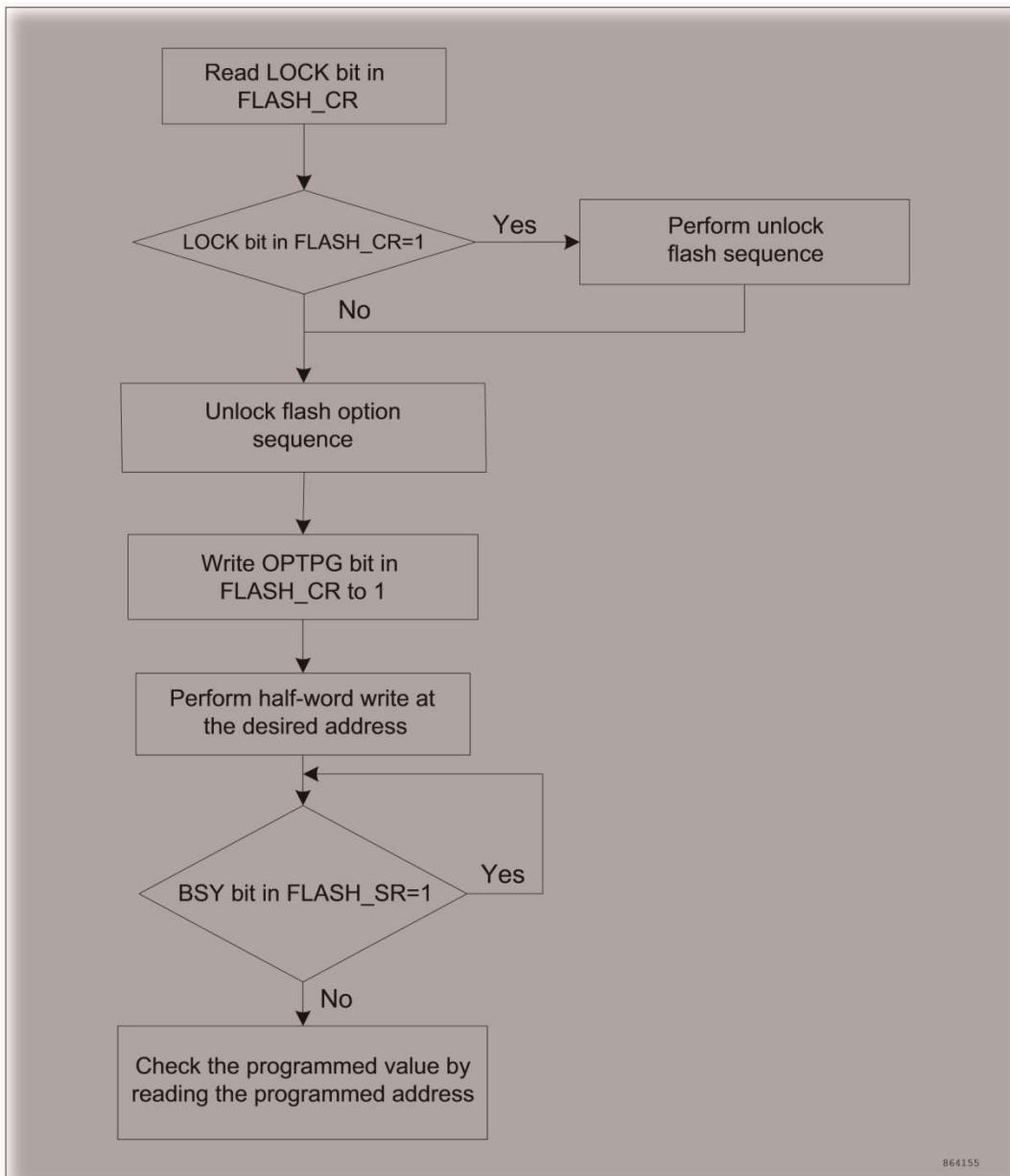


Figure 5. Option byte programming procedures

## Erase procedure

The option byte erase sequence is as follows:

- Check the BSY bit in the FLASH\_SR register to confirm that the last operation has finished
- Unlock the OPTWRE bit in the FLASH\_CR register
- Set the OPTER bit to 1 in the FLASH\_CR register
- Set the STRT bit to 1 in the FLASH\_CR register
- Wait for the BSY bit to be reset
- Read and verify

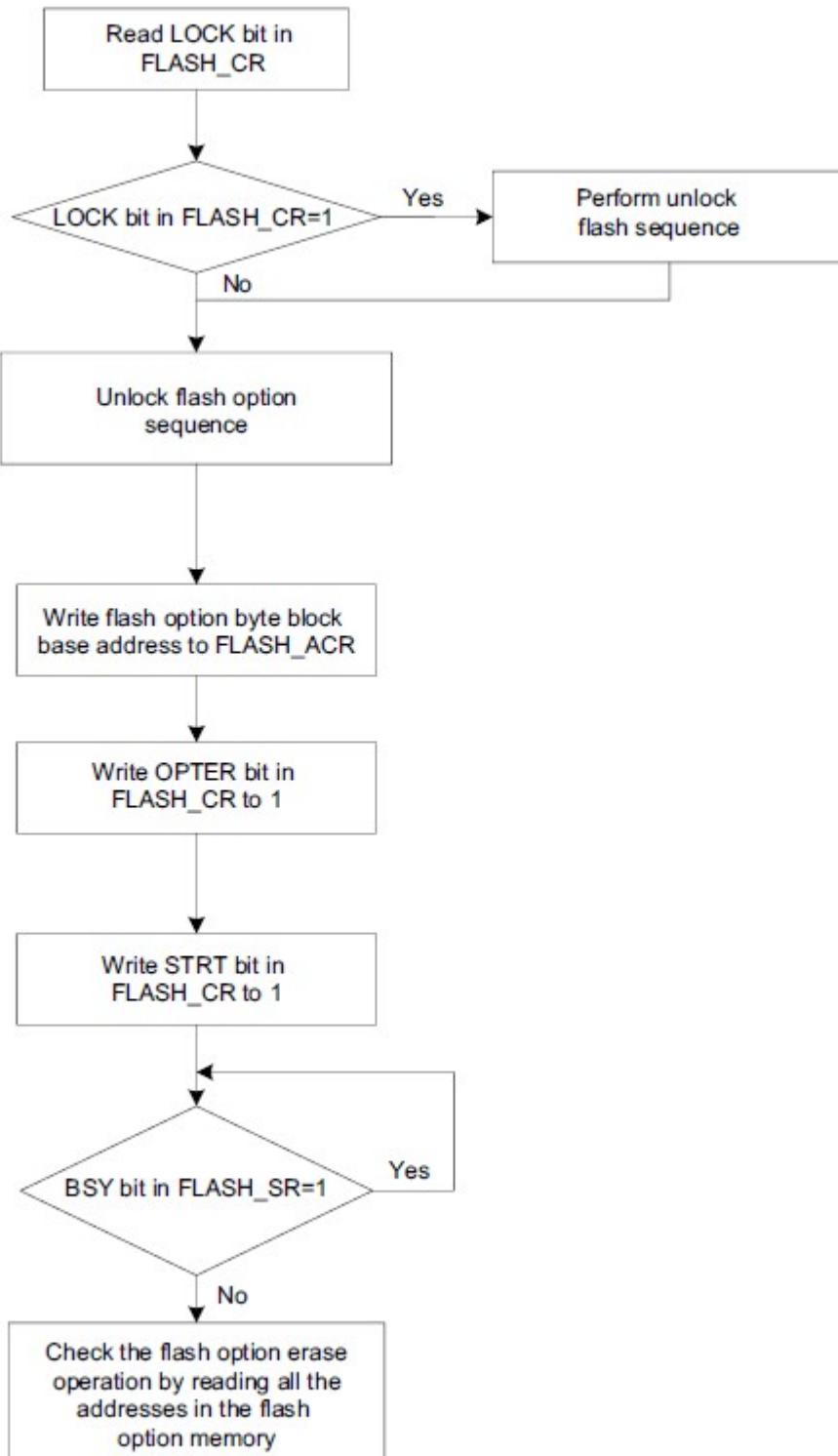


Figure 6. Option byte erase procedure

80/682.5

## 2.3 Memory protection

The user area of the Flash memory can be guarded against reading by untrusted code. The Flash memory can also be guarded against unwanted erase caused by program runaway. The write-protection granularity is then of one sector (four pages).

### 2.3.1 Read protection

The read protection is activated by setting the RDP2 and RDP3 options (two half words) and then, by re-applying a system POR (power-on reset) to reload new RDPs.

Note: If the read protection is set while the debugger is still connected to SWD/JTAG, apply a POR instead of a system reset (without debugger).

#### Set as read protection

According to the half-word programming method in the option byte area, and write two half words for the RDP2 and RDP3 options in sequence to the corresponding address.

1. Write the desired value of 0x7F80 to 0x1FFE0000
2. Write the desired value of 0xFF00 to 0x1FFE0002

Once corresponding values have been written to the protection byte:

- Main Flash memory read access is not allowed except for the user code (when booting from main Flash memory itself in the non-debug mode).
- After the read protection is activated, the Flash is inaccessible in the debug mode (sram boot and debug modes). 4KB space below is not allowed to be written in Flash's own programs .
- Pages 0-3 are automatically write-protected. The rest of the memory can be programmed by the code executed from the main Flash memory (for IAP or data storage, etc.), but it is protected against write/erase (but not against mass erase) in debug mode or when booting from the embedded SRAM.
- All features for loading code into and executing code from the embedded SRAM via SWD are still active. Booting from embedded SRAM via SWD is also allowed and this can be used to disable the read protection.
- When executing code from the embedded SRAM to access the main Flash memory, Flash memory accesses through the DMA, SWV (serial wire viewer), SWD (serial wire debug), ETM () and boundary scan are not allowed.

The Flash memory is protected when the RDPs contain the following values:

Table 4. Flash memory read protection status

RDP Byte Value	Read Protection Status
RDP2=0x7F80 @0x1FFE0000	Protected
RDP3=0xFF00 @0x1FFE0002	

#### Disable read protection

To disable the read protection from the embedded SRAM:

According to the half-word programming method in the option byte area, and write two half words for the RDP2 and RDP3 options in sequence to the corresponding address:

1. Set the Flash AR address value to 0xFFFF800 and execute the block erase
2. Set the Flash AR address value to 0x08000000 and execute the main Flash mass erase
3. Set the Flash AR address value to 0x1FFE0000 and execute the block erase. It is equivalent to writing 0xFFFF to RDP2 and RDP3

Apply a POR reset to reload the option bytes. At this time, the read protection is disabled.

Table 5. Disabled protection status of Flash memory

RDPs Byte Value	Read Protection Status
Erase the 0xFFFF800 option block	
Trigger the main Flash mass erase for 0x08000000	
Erase the 0xFFE0000 option block, to ensure RDP2 is 0xFFFF @0xFFE0000 and RDP3 is 0xFFFF @0xFFE0002	Protection is disabled

Note: If the corresponding address value for the option byte block is not 0xFFFF, the erase of option byte block should be executed first. Erasing the option byte block will not trigger an automatic mass erase or change the read protection status. At the same time, the main Flash mass erase for 0x08000000 is required.

### 2.3.2 Write protection of main memory

The write protection is controlled in a unit of sector (4 pages). The WRP bit in the option bytes should be configured, and then the system reset will load the new option bytes to enable the protection. If a write or an erase operation is performed on a protected sector, the WRPRTER flag bit in the FLASH\_SR will be set .

#### Disable write protection

To disable the write protection, two application cases are provided:

- Case 1: Disable write protection and read protection:
  - Erase the entire option byte area by using the OPTER bit in the Flash memory control register (FLASH\_CR);
  - Perform a Mass Erase to the main Flash memory for 0x08000000;
  - Reset the system to reload the option byte (including the new WRP byte), and to disable the write protection.

This operation will disable the write protection of the entire main Flash memory module.

- Case 2: Disable write protection while keeping the read protection active. This is useful for in-application programming with a user boot loader:
  - Erase the entire option byte area by using the OPTER bit in the Flash memory control register (FLASH\_CR);
  - Reset the system to reload the option byte (and the new WRP byte); disable the write protection.

This operation will disable the write protection of the entire main Flash memory module except for page 0 to page 3 where the write protection stays active.

### 2.3.3 Option byte write protection

The option byte blocks are always read-accessible and write-protected by default. To gain write access (program/erase) to the option byte blocks, a sequence of keys (same as for lock) has to be written into the OPTKEYR. It then gives write access to the option byte blocks and this is indicated by OPTWRE bit in the FLASH\_CR register. Write access can be disabled by clearing the bit.

## 2.4. Flash interrupt

Table 6. Flash interrupt request

Interrupt Event	Event Flag	Enable Control Bit
End of operation	EOP	EOPIE
Write protection error	WRPRTERR	ERRIE
Programming error	PGERR	ERRIE

## 2.5 Option byte description

Option bytes are configured by the user based on the application requirements. For example, the watchdog may be selected in hardware or software mode.

A 32-bit word is split up in the following option bytes:

Table 7. Option byte format

Bits 31 ~ 24	Bits 23 ~ 16	Bits 15 ~ 8	Bits 7 ~ 0
Complement of option byte 1	Option byte 1	Complement of option byte 0	Option byte 0

Note: Complements are automatically written by hardware. Software programming is invalid.

The organization of these option bytes in the option byte block is as below.

The option bytes can be read from the memory addresses listed in the table below or from the option byte register (FLASH\_OBR).

Note: The new programmed option bytes (user, read/write protection) take effect after a system reset.

Table 8. Option byte organization

Address	[31:24]	[23:16]	[15:8]	[7:0]
0x1FFF F800	nUSER	USER	nReserved	Reserved
0x1FFF F804	nData1	Data1	nData0	Data0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF F80C				

Table 9. Option bytes description

Flash memory address	Option bytes
0x1FFF F800	<p>Bits [31:24] nUSER</p> <p>Bits [23:16] USER: User option byte (stored in FLASH_OBR[9:2]) This byte is used to configure the following features:</p> <ul style="list-style-type: none"> <li>Select the watchdog event: hardware or software</li> </ul> <p>Note: Only bits [20] and [16] are used; bits [23:17] and [19:17] are not used.</p> <p>Bit 20: nBOOT1</p> <p>Bit 16: WDG_SW</p> <p>0: hardware watchdog 1: software watchdog</p> <p>Bits [15:0]: reserved, operation disabled</p>
0x1FFF F804	<p>Datax: two bytes for user data</p> <p>The address can be programmed with the option byte programming procedure.</p> <p>Bits [31:24]: nData1</p> <p>Bits [23:16]: Data1 (stored in FLASH_OBR[25:18])</p> <p>Bits [15:8]: nData0</p> <p>Bits [7:0]: Data0 (stored in FLASH_OBR[17:10])</p>
0x1FFF F808	<p>WRPx: Flash memory write protection option bytes</p> <p>Bits [31:24]: nWRP1</p> <p>Bits [23:16]: WRP1 (stored in FLASH_WRPR[15:8])</p> <p>Bits [15:8]: nWRP0</p> <p>Bits [7:0]: WRPO (stored in FLASH_WRPR[7:0])</p> <p>Each bit of the option byte WRPx is used to protect 4 pages in the main memory.</p> <p>0: write protection active 1: write protection not active</p> <p>In total, two user option bytes are used to protect the 64-Kbyte main Flash memory.</p> <p>WRP0: write-protects for pages 0 to 31</p> <p>WRP1: write-protects for pages 32 to 63</p>

On every system reset, the option byte loader (OBL) reads the information block and stores the data into the option byte register (FLASH\_OBR). Each option bit has its complement in the information block. When option bit is loaded, its complement is used to verify that it has correctly taken place. If this is not the case, an option byte error flag(OPTERR) is generated. When an error occurs, the corresponding option byte is forced to be set as 0xFF. The verification feature is disabled when the option byte and its complement are both 0xFF (after-erase state).

All option bits (excluding their complements) are used to configure the microcontroller. The option byte registers can be read by the CPU.

## 2.6. Flash register descriptions

Table 10. Flash register overview

Offset	Acronym	Register Name	Reset	Section
0x00	FLASH_ACR	Flash access control register	0x00000038	Section 2.6.1
0x04	FLASH_KEYR	FPEC key register	0x00000000	Section 2.6.2
0x08	FLASH_OPTKEYR	Flash OPTKEY register	0x00000000	Section 2.6.3
0x0C	FLASH_SR	Flash status register	0x00000000	Section 2.6.4
0x10	FLASH_CR	Flash control register	0x00000080	Section 2.6.5
0x14	FLASH_AR	Flash address register	0x00000000	Section 2.6.6
0x1C	FLASH_OBR	Option byte register	0x03FFFC1C	Section 2.6.7
0x20	FLASH_WRPR	Write protection register	0xFFFFFFFF	Section 2.6.8

### 2.6.1 Flash access control register (FLASH\_ACR)

Address offset: 0x00

Reset value: 0x0000 0038

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

Bit	Field	Type	Reset	Description
31:6	Reserved			Reserved, always read as 0.
5	PRFTBS	r	0x01	Prefetch buffer status 0: prefetch buffer is disabled 1: prefetch buffer is enabled
4	PRFTBE	rw	0x01	Prefetch buffer enable 0: prefetch buffer is disabled 1: prefetch buffer is enabled
3	Reserved			Reserved and always read as 1.

Bit	Field	Type	Reset	Description
2:0	LATENCY	rw	0x00	<p>Latency</p> <p>These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time.</p> <p>000: zero wait state, if <math>0 &lt; \text{SYSCLK} \leq 24\text{MHz}</math></p> <p>001: one wait state, if <math>24\text{MHz} &lt; \text{SYSCLK} \leq 48\text{MHz}</math></p> <p>010: two wait states, if <math>48\text{MHz} &lt; \text{SYSCLK} \leq 72\text{MHz}</math></p>

## 2.6.2 Flash key register (FLASH\_KEYR)

Address offset: 0x04

Reset value: 0x0000 0000															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FKEYR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FKEYR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
31:0	FKEYR	w	0x0000 0000	<p>FPEC key (Flash key)</p> <p>These bits represent the keys to unlock the FPEC.</p>

Note: These bits are all write-only and will return a 0 when read.

## 2.6.3 Flash OPTKEY register (FLASH\_OPTKEYR)

Address offset: 0x08

Reset value: 0x0000 0000															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEYR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTKEYR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
31:0	OPTKEYR	w	0x0000 0000	<p>Option byte key</p> <p>These bits represent the keys to unlock the OPTWRE.</p>

Note: These bits are all write-only and will return a 0 when read.

## 2.6.4 Flash status register (FLASH\_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EOP	WRPRTE RR	Res.	PGERR	Res.	BSY	r	rc_w1

Bit	Field	Type	Reset	Description
31:6	Reserved			Reserved, always read as 0.
5	EOP	rc_w1	0x00	<p>End of operation</p> <p>Set by hardware to '1' when a Flash operation (programming/erase) is completed. Its status can be cleared by writing a '1' to it.</p> <p>Note: EOP status is set at the end of each successful program or erase operation.</p>
4	WRPRTERR	rc_w1	0x00	<p>Write protection error</p> <p>Set by hardware to '1' when a write-protected address of the Flash memory to be programmed. Its status can be cleared by writing a '1' to it.</p>
3	Reserved			Reserved and always read as 0.
2	PGERR	rc_w1	0x00	<p>Programming error</p> <p>Set by hardware to '1' when an address to be programmed contains a value different from '0xFFFF'. Its status can be cleared by writing a '1' to it.</p> <p>Note: The STRT bit in the FLASH_CR register should be cleared before starting a programming operation.</p>
1	Reserved			Reserved and always read as 0.
0	BSY	r	0x00	This bit indicates that a Flash operation is in progress. It is set to '1' on the beginning of a Flash operation and cleared (as '0') when the operation finishes or when an error occurs.

## 2.6.5 Flash control register (FLASH\_CR)

Address offset: 0x10

Reset value: 0x0000 0080

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

Reserved
----------

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	EOPIE	Res.	ERRIE	OPTWRE	Res.	LOCK	STRT	OPTER	OPTPG	Res.	MER	PER	PG		
	rw		rw	rc_w0		rw	rw	rw	rw		rw	rw	rw		

Bit	Field	Type	Reset	Description
31:13	Reserved			Reserved, always read as 0.
12	EOPIE	rw	0x00	<p>End of operation interrupt enable</p> <p>This bit enables the interrupt generation when the EOP bit in the FLASH_SR register goes to '1'.</p> <p>0: interrupt generation disabled 1: interrupt generation enabled</p>
11	Reserved			Reserved and always read as 0.
10	ERRIE	rw	0x00	<p>Error interrupt enable</p> <p>This bit enables the interrupt generation on an FPEC error (when PGERR/WRPRTERR is set to '1' in the FLASH_SR register).</p> <p>0: interrupt generation disabled 1: interrupt generation enabled</p>
9	OPTWRE	rc_w0	0x00	<p>Option byte write enable</p> <p>When set to '1', the option bytes can be programmed.</p> <p>This bit is set to '1' on writing the correct key sequence to the FLASH_OPTKEYR register.</p> <p>Writing '0' to it by software can clear this bit.</p>
8	Reserved			Reserved and always read as 0.
7	LOCK	rw	0x01	<p>Lock</p> <p>Write to '1' only. When it is set to '1', it indicates that the FPEC and FLASH_CR are locked. This bit is cleared to '0' by hardware after detecting the correct unlock sequence.</p> <p>In the event of unsuccessful unlock operation, this bit is not allowed to be changed until the next system reset.</p>
6	STRT	rw	0x00	<p>Start</p> <p>This bit triggers an ERASE operation when set to '1'. This bit can be set to '1' only by software and cleared to '0' automatically when the BSY bit is set to '1'.</p>
5	OPTER	rw	0x00	Option byte erase Option byte erase chosen.
4	OPTPG	rw	0x00	Option byte programming Option byte programming chosen.
3	Reserved			Reserved and always read as 0.

Bit	Field	Type	Reset	Description
2	MER	rw	0x00	Mass erase Erase of all user pages chosen.
1	PER	rw	0x00	Page erase Page erase chosen.
0	PG	rw	0x00	Programming Flash programming chosen.

### 2.6.6 Flash address register (FLASH\_AR)

Address offset: 0x014

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:0	FAR	w	0x0000 0000	Flash Address Chooses the address to program when programming is ongoing, or a page to erase when Page Erase is selected. Note: Write access to this register is blocked when the BSY bit in the FLASH_SR register is set to '1'.

Updated by hardware to the currently/last used address. For Page Erase operations, this register should be updated to indicate the page to erase.

### 2.6.7 Option byte register (FLASH\_OBR)

Address offset: 0x1C

Reset value: 0x03FF FC1C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					Data1					Data0					
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data0					Reserved					nBOOT1	Res.	nRST_STDBY	nRST_STOP	WDG_SW	Res. OPTERR
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31:26	Reserved			Reserved, always read as 0.
25:18	Data1	r	0xFF	Data1
17:10	Data0	r	0xFF	Data0

Bit	Field	Type	Reset	Description
9:7	Reserved			Reserved and always read as 0.
6	nBOOT1	r	0x01	nBOOT1
5	Reserved			Reserved and always read as 0.
4	nRST_STDBY	r	0x01	Reset event when entering the Standby mode 0: generate a reset event when entering the Standby mode 1: do not generate a reset event when entering the Standby mode
3	nRST_STOP	r	0x01	Reset event when entering the Stop mode 0: generate a reset event when entering the Stop mode 1: do not generate a reset event when entering the Stop mode
2	WDG_SW	r	0x01	Select the watchdog event 0: hardware watchdog 1: software watchdog
1	Reserved			Reserved and always read as 0.
0	OPTERR	r	0x00	Option byte error When set to '1', this bit indicates that the option byte and its complement do not match. Note: This bit is read-only.

The value used to reset this register is related to the value written to the option byte. The value used to reset the OPTERR bit is related to the comparison between the option byte and its complement when loading the option byte.

### 2.6.8 Write protection register (FLASH\_WRPR)

Address offset: 0x20

Reset value: 0xFFFF FFFF

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

Bit	Field	Type	Reset	Description
31:0	WRP	r	0xFFFF FFFF	Write protection This register contains the write-protection option bytes loaded by the OBL. 0: write protection active 1: write protection not active Note: These bits are read-only.

# 3

# Cyclic Redundancy Check (CRC) Calculation Unit

Cyclic Redundancy Check (CRC) Calculation Unit

## 3.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC calculation result from a 32-bit full word based on a fixed generator polynomial. Among other applications, CRC-based techniques are used to verify data transmission or storage correctness or storage integrity. In the scope of the EN/IEC60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at linktime and stored at a given memory location.

## 3.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$
- Single input/output 32-bit data register
- CRC computation done in 5 AHB clock cycles (HCLK)
- General-purpose 8-bit register (can be used for temporary storage)

The CRC calculation unit block diagram is shown below.

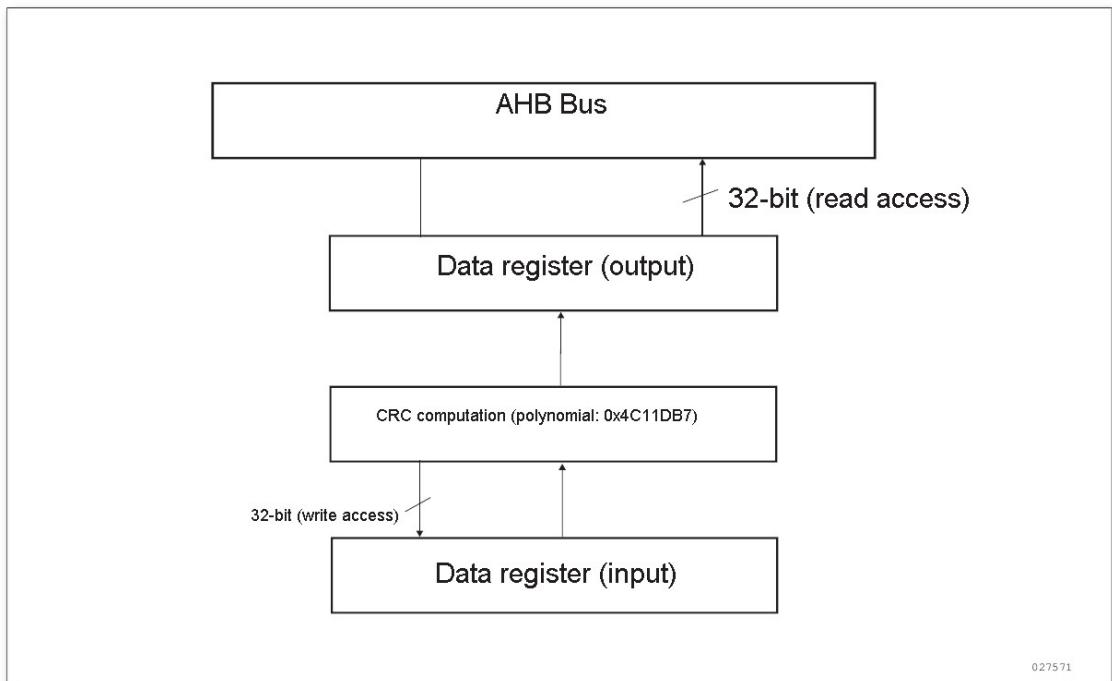


Figure 7. CRC calculation unit block diagram

027571

### 3.3 CRC function description

The CRC calculation unit mainly consists of a single 32-bit data register, which:

- is used as an input register to enter new data in the CRC calculator (when writing into the register).
- holds the result of the previous CRC calculation (when reading the register).

Each write operation into the data register creates a combination of the previous CRC calculation value and the new one (CRC computation is done on the whole 32-bit data word, and not byte per byte).

The write operation is stalled until the end of the CRC computation, thus allowing back-to-back write accesses or consecutive write and read accesses against the CRC\_DR register.

The CRC\_DR register can be reset to 0xFFFF FFFF with the RESET bit in the CRC\_CTRL register.

This operation does not affect the contents of the CRC\_IDR register.

### 3.4 CRC registers

The CRC calculation unit contains two data registers and a control register.

Table 11. Overview of CRC registers

Offset	Acronym	Register Name	Reset	Section
0x00	CRC_DR	CRC data register	0xFFFFFFFF	Section 3.4.1
0x04	CRC_IDR	CRC independent data register	0x00000000	Section 3.4.2
0x08	CRC_CTRL	CRC control register	0x00000000	Section 3.4.3
0x20	CRC_REVERSE	CRC data reverse register	0x00000000	Section 3.4.4

#### 3.4.1 CRC data register (CRC\_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bit	Field	Type	Reset	Description
31:0	DR	rw	0xFFFF FFFF	DR: data register bits Read as an input register to return the CRC calculation result when writing new data into the CRC calculator.

### 3.4.2 CRC independent data register (CRC\_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:8	Reserved			always read as 0.
7:0	IDR	rw	0x00	IDR: general-purpose 8-bit data register bits Can be used as a temporary storage location for one byte. This register is not affected by CRC resets generated by the RESET bit in the CRC_CTRL register.

Note: This register is not involved in the CRC calculation, thus it can be used as a storage for any data.

### 3.4.3 CRC control register (CRC\_CTRL)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:1	Reserved			always read as 0.

0	RESET	w	0x00	RESET: CRC reset calculation unit Sets the data register to 0xFFFF FFFF. This bit can only be set to '1', and it is automatically cleared to '0' by hardware.
---	-------	---	------	--

### 3.4.4 CRC data reverse register (CRC\_REVERSE)

Address offset: 0x20

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
REVERSE															

Bit	Field	Type	Reset	Description
31:1	Reserved			always read as 0.
0	REVERSE	rw	0x00	REVERSE: data reverse register bits The register writes data into HWData[31:0] and reads data HW-Data[0:31].

**4**

# Hardware Division (HWDIV)

## Hardware Division (HWDIV)

### 4.1 Hardware division introduction

Hardware division is very useful for some high-performance applications as it is capable of automatic division of signed or unsigned 32-bit integers.

### 4.2 Main features of hardware division

- Signed or unsigned division of integers
- Outputs 32-bit quotient and remainder from 32-bit dividend and divisor
- Completes in 8 HCLK cycles
- If the divisor is zero, an overflow interrupt flag will be generated
- Writing the divisor will trigger the division operation automatically
- Waits automatically until the calculation is completed when reading the quotient and remainder, without checking the state bit

### 4.3 Hardware division functions

The hardware division unit consists of four 32-bit data registers which are dividend, divisor, quotient and remainder, and the unit is capable of signed or unsigned 32-bit division. The hardware division control register USIGN can be used to select whether the division is signed or unsigned.

Each time the divisor register is written, the division operation is automatically triggered. After the operation is completed, the result is written to the quotient and remainder registers. If the quotient register, remainder register, or status register is read before the end, the read operation is suspended. The operation result will not return until the end.

If the divisor is zero, an overflow interrupt flag will be generated.

### 4.4 Hardware division registers

Table 12. Overview of hardware division registers

Offset	Acronym	Register Name	Reset	Section
0x00	HWDIV_DVDR	Dividend register	0x00000000	Section 4.4.1
0x04	HWDIV_DVSR	Divisor register	0x00000001	Section 4.4.2

Offset	Acronym	Register Name	Reset	Section
0x08	HWDIV_QUOTR	Quotient register	0x00000000	Section 4.4.3
0x0C	HWDIV_RMDR	Remainder register	0x00000000	Section 4.4.4
0x10	HWDIV_SR	HWDIV status register	0x00000000	Section 4.4.5
0x14	HWDIV_CR	HWDIV control register	0x00000001	Section 4.4.6

#### 4.4.1 Dividend register (HWDIV\_DVDR)

Address offset: 0x00

Reset value: 0x0000 0000

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

DIVIDEND															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:0	DIVIDEND	rw	0x0000 0000	Dividend data Division operation is automatically triggered after the register is written

#### 4.4.2 Divisor register (HWDIV\_DVSR)

Address offset: 0x04

Reset value: 0x0000 0001

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

DIVISOR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:0	DIVISOR	rw	0x0000 0001	Divisor data

#### 4.4.3 Quotient register (HWDIV\_QUOTR)

Address offset: 0x08

Reset value: 0x0000 0000

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

QUOTIENT																	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
QUOTIENT																	
Bit	Field	Type	Reset	Description													
31:0	QUOTIENT	r	0x0000 0000	Quotient data													

#### 4.4.4 Remainder register (HWDIV\_RMDR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
REMAINDER																	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
REMAINDER																	
Bit	Field	Type	Reset	Description													
31:0	REMAINDER	r	0x0000 0000	Remainder data													

#### 4.4.5 HWDIV status register (HWDIV\_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.																	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Res.																	OVF
Bit	Field	Type	Reset	Description													
31:1	Reserved			Reserved, always read as 0.													
0	OVF	r	0x00	Overflow flag bit Set to '1' by software before the next division operation. 1: Current divisor is zero. 0: Current divisor is not zero.												r	

#### 4.4.6 HWDIV control register (HWDIV\_CR)

Address offset: 0x14

Reset value: 0x0000 0001

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

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

Bit	Field	Type	Reset	Description
31: 2	Reserved			Reserved, always read as 0.
1	OVFE	rw	0x00	Overflow interrupt enable 1: overflow interrupt enabled 0: overflow interrupt disabled
0	USIGN	rw	0x01	Unsigned enable 1: unsigned division 0: signed division

# 5 | Serial Transceiver Module (CSM)

## Serial Transceiver Module (CSM)

### 5.1 Introduction

The serial transceiver module (CSM) is mainly used to receive and transmit the serial data. The data is input through the comparator or GPIO, converted to 32-bit data by internal baud rate sampling, and saved in the memory by CPU or DMA. Then the software will analyze the received data.

### 5.2 Main features

- Input through the comparator or GPIO can be selected
- Supports voltage setting of transmitted signal
- Supports asynchronous transmission and reception only
- Supports DMA request
- 16-bit programmable baud rate generator
- Programmable idle bus status (low/high)

### 5.3 Function description

#### 5.3.1 Block diagram

The block diagram of CSM is shown as below. Input serial signals are converted to 32-bit registers, or 32-bit transmit registers are shifted and output to serial data by shift registers.

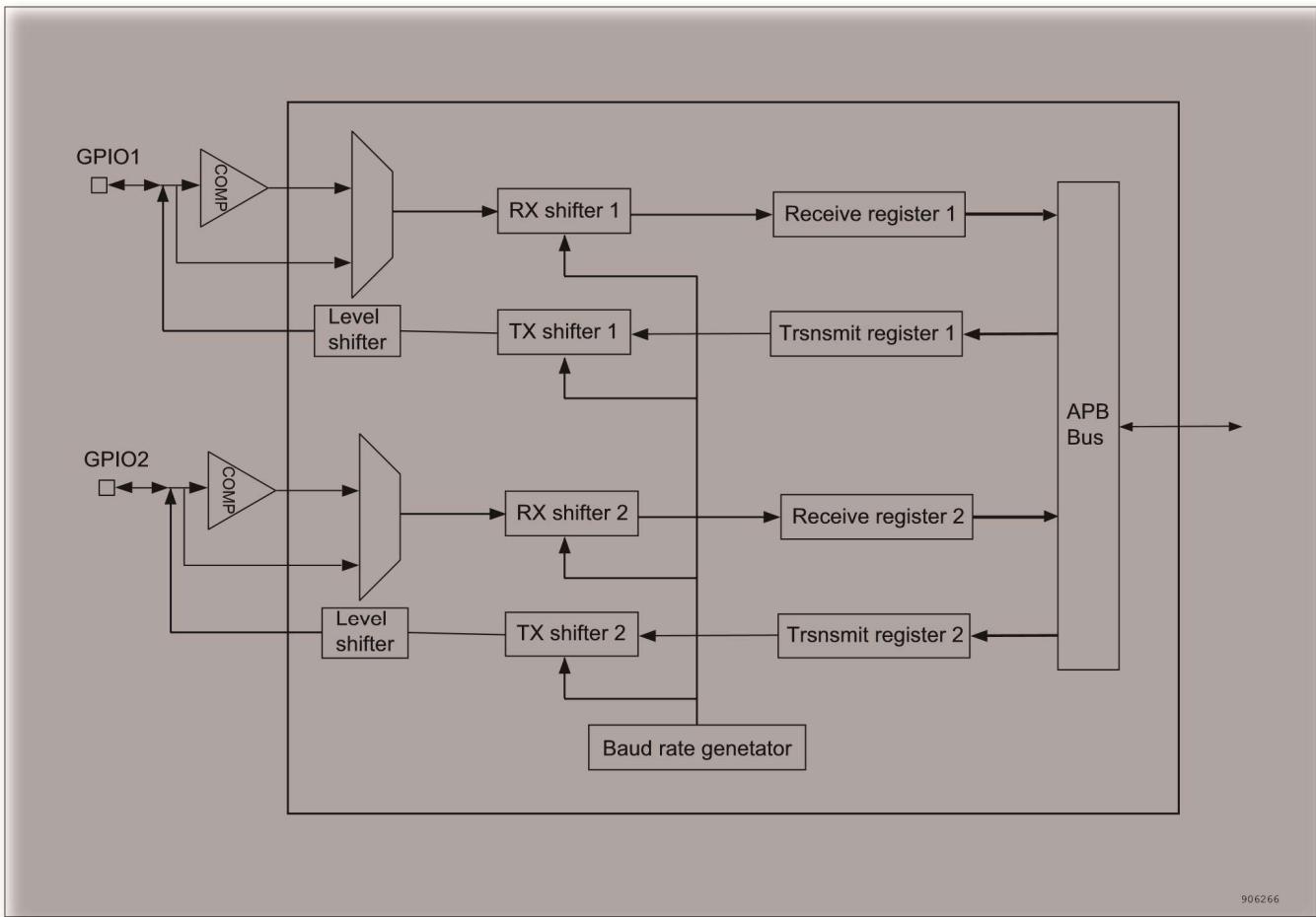


Figure 8. CSM block diagram

The diagram of PD application is shown as below. CC1 and CC2 are received by the comparator and driven by the bleeder circuit respectively. The driving voltage is controlled by the VTXSEL bit in the register. ADC monitors the voltages of CC1 and CC2. The other two digital GPIOs control whether to enable the pull-up resistor Rp. If the GPIO output is high, then CC1/2 is pulled up to VCC through Rp; if the GPIO is configured as floating input, then the pull-up resistor Rp has no effect.

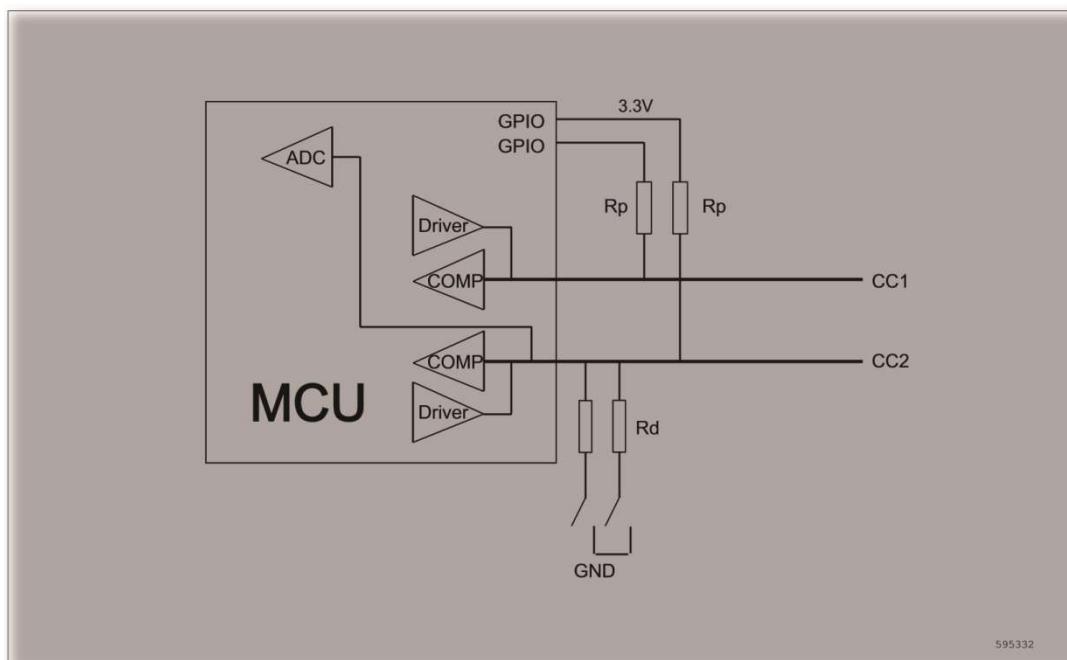


Figure 9. CSM block diagram

### 5.3.2 External port

Either the GPIO or the comparator can be used as the external port for input. If the comparator is selected, the input signal can be 0.9V or 1.2V.

The voltage for the output signal and the chip is the same.

### 5.3.3 Data reception

During a reception, data is input in least significant bit first. After 32-bit data is received, the data shifts from the shift register to the corresponding reception register. If the interruption reception is enabled, then an interrupt is generated. If the DMA is enabled, then a DMA request signal is generated.

Serial data reception can be configured as ongoing sampling after enabling channels; the sampling start condition and stop condition can also be configured as below:

Start condition: the sampling starts after a non-idle level. If the idle level is high, then the sampling starts after a low level. If the idle level is low, then the sampling starts after a high level.

Stop condition: if the data length is fixed, the sampling stops after selected bits in the reception setting; if the data length is not fixed, the sampling stops after a selected number of idle levels in the reception setting.

Two reception channels can operate independently; or one channel is set as the master channel, and the other channel works with synchronized start and stop conditions.

### 5.3.4 Data transmission

During a transmission, once data is written to the 32-bit data register, it will shift to the Tx shift register. Data shifts out the least significant bit first. When the data shifts to the shift register, an interrupt is generated if the interruption transmission is enabled. If the DMA is enabled, then a DMA request signal is generated.

If the transmission starts in the idle state, a non-idle level will be generated first as the start condition. The transmission stops after configured bits are sent.

Two transmission channels can operate independently; or one channel is set as the master channel, and the other channel works with synchronized start and stop conditions.

### 5.3.5 Status flag

To ensure the convenient operation of the software, interrupt status flags are provided for the application to monitor the bus state. The current status flags are automatically set by hardware and cleared by writing a '1'. Interrupt status flag bit is set when an event occurs. A CPU interrupt is generated when the interrupts are enabled. They are cleared by software.

### 5.3.6 Baud rate configuration

The baud rate is the generated frequency of sampling clock, generally the frequency division of PCLK. BRG is a 16-bit baud rate generator. The SPBREG register controls the counting cycle of the 16-bit counter.

If expected baud rate and  $f_{\text{pclk}}$  (frequency of APB module) are provided, the formula shown in the following table can be used to calculate an approximate value to assign the SPBRG register, where X represents the value for SPBRG register (2 ~ 65535).

Table 13. Baud rate formula

Mode	Formula
------	---------

Transmission/Sampling mode	Baud rate = $f_{\text{pclk}}/X$
----------------------------	---------------------------------

### 5.3.7 DMA communication

In order to reach the maximum communication speed, the Tx buffer should be filled with data timely; similarly, the data in the Rx buffer should be read timely to prevent overflow. To ensure the convenient high-speed data transfer, a simple DMA request/response system is adopted.

When the DMAEN bit is set, the module sends the request of DMA data transfer. DMA requests of both Tx and Rx buffers are enabled through DMAEN.

## 5.4 Register file and memory mapping description

Table 14. Overview of CSM register description

Offset	Acronym	Register Name	Reset	Section
0x00	CSM_TXREG1	CSM transmit data register 1	0x00000000	Section 5.4.1
0x04	CSM_TXREG2	CSM transmit data register 2	0x00000000	Section 5.4.2
0x08	CSM_RXREG1	CSM receive data register 1	0x00000000	Section 5.4.3
0x0C	CSM_RXREG2	CSM receive data register 2	0x00000000	Section 5.4.4
0x10	CSM_INTSTAT	CSM interrupt status register	0x00000000	Section 5.4.5
0x14	CSM_INTEN	CSM interrupt enable register	0x00000000	Section 5.4.6
0x18	CSM_CTL1	CSM control register 1	0x00000008	Section 5.4.7
0x1C	CSM_CTL2	CSM control register 2	0x00000008	Section 5.4.8
0x20	CSM_CFG	CSM configuration register	0x00000008	Section 5.4.9
0x24	CSM_SPBRG	CSM baud rate generator	0x00000002	Section 5.4.10
0x28	CSM_BCNT	CSM data bit count	0x00000000	Section 5.4.11

### 5.4.1 CSM transmit data register 1 (CSM\_TXREG1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXREG1[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXREG1[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description											

31: 0	TXREG1	rw	0x0000	TXREG1: Transmit data register 1
			0000	

#### 5.4.2 CSM transmit data register 2 (CSM\_TXREG2)

Address offset: 0x04

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXREG2[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXREG2[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31: 0	TXREG2	rw	0x0000	TXREG2: Transmit data register 2
			0000	

#### 5.4.3 CSM receive data register 1 (CSM\_RXREG1)

Address offset: 0x08

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXREG1[31: 16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXREG1[15: 0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31: 0	RXREG1	r	0x0000	RXREG1: Receive data register 1
			0000	

#### 5.4.4 CSM receive data register 2 (CSM\_RXREG2)

Address offset: 0x0C

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXREG2[31: 16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXREG2[15: 0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

31: 0	RXREG2	r	0x0000	RXREG2: Receive data register 2
			0000	

#### 5.4.5 CSM interrupt status register (CSM\_INTSTAT)

Address offset: 0x10

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										STOP_INTF	START_INTF	TXC_INTF	RX_INTF	TX_INTF	rw

Bit	Field	Type	Reset	Description
31:5	Reserved			always read as 0.
4	STOP_INTF	rw	0x00	<p>STOP_INTF: Stop data reception interrupt flag</p> <p>When the stop status is detected, it is set automatically by hardware. It is cleared by writing a '1' to it.</p> <p>1 = stop reception of effective bytes</p> <p>0 = no stop flag</p>
3	START_INTF	rw	0x00	<p>START_INTF: Start data reception interrupt flag</p> <p>When the start status is detected, it is set automatically by hardware. It is cleared by writing a '1' to it.</p> <p>1 = start reception of effective bytes</p> <p>0 = no start flag</p>
2	TXC_INTF	rw	0x00	<p>TXC_INTF: Transmit shift register complete interrupt flag bit</p> <p>When the Tx buffer and shift register are both empty, it is set automatically by hardware. It is cleared by writing a '1' to it.</p> <p>1= Transmission of shift register data is completed</p> <p>0= Shift register is empty or shift transmission is ongoing</p>
1	RX_INTF	rw	0x00	<p>RX_INTF: Receive data available interrupt flag bit</p> <p>This bit is set automatically by hardware when the Rx buffer receives a complete byte. It is cleared by writing a '1' to it.</p> <p>1 = Rx buffer has received data of effective byte</p> <p>0 = Rx buffer is empty</p>

Bit	Field	Type	Reset	Description

0	TX_INTF	rw	0x00	TX_INTF: Transmit FIFO available interrupt flag bit (one byte is sent) This bit is automatically set by hardware when the Tx buffer is empty. It is cleared by writing a '1' to it. 1 = Tx buffer is valid 0 = Tx buffer is invalid
---	---------	----	------	--

#### 5.4.6 CSM interrupt enable register (CSM\_INTEN)

Address offset: 0x14

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										STOP_IEN	START_IEN	TXC_IEN	RX_IEN	TX_IEN	
										rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:5	Reserved			always read as 0.
4	STOP_IEN	rw	0x00	STOP_IEN: Stop data reception interrupt enable bit 1 = interrupt enabled 0 = interrupt disabled
3	START_IEN	rw	0x00	START_IEN: Start data reception interrupt enable bit 1 = interrupt enabled 0 = interrupt disabled
2	TXC_IEN	rw	0x00	TXC_IEN: Transmit shift register complete interrupt enable bit 1 = interrupt enabled 0 = interrupt disabled
1	RX_IEN	rw	0x00	RX_IEN: Receive FIFO interrupt enable bit 1 = interrupt enabled 0 = interrupt disabled
0	TX_IEN	rw	0x00	TX_IEN: Transmit FIFO empty interrupt enable bit 1 = interrupt enabled 0 = interrupt disabled

#### 5.4.7 CSM control register 1 (CSM\_CTL1)

Address offset: 0x18

Reset value: 0x00000008

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

Reserved															
----------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												TXSEL1	DMAEN1	INSEL1	IDLEP1	EN1
												rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:5	Reserved			always read as 0.
4	TXSEL	rw	0x00	TXSEL: Select to transmit or receive data 1 = transmit data 0 = receive data
3	DMAEN	rw	0x01	DMAEN: enable the DMA 1 = enabled 0 = disabled
2	INSEL	rw	0x00	INSEL: Select the input signal source 1 = input through the comparator 0 = input through the GPIO
1	IDLEP	rw	0x00	IDLEP: idle level 1: a high level when idle 0: a low level when idle
0	EN	rw	0x00	EN: enable the module 1 = enabled 0 = disabled

#### 5.4.8 CSM control register 2 (CSM\_CTL2)

Address offset: 0x1C

Reset value: 0x00000008

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												TXSEL2	DMAEN2	INSEL2	IDLEP2	EN2
												rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:5	Reserved			always read as 0.
4	TXSEL	rw	0x00	TXSEL: Select to transmit or receive data 1 = transmit data 0 = receive data
3	DMAEN	rw	0x01	DMAEN: enable the DMA 1 = enabled 0 = disabled
2	INSEL	rw	0x00	INSEL: Select the input signal source 1 = input through the comparator 0 = input through the GPIO
1	IDLEP	rw	0x00	IDLEP: idle level 1: a high level when idle 0: a low level when idle
0	EN	rw	0x00	EN: enable the module 1 = enabled 0 = disabled

#### 5.4.9 CSM configuration register (CSM\_CFG)

Address offset: 0x20

Reset value: 0x00000008

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

Bit	Field	Type	Reset	Description
31:16	MAXBIT	rw	0x00	MAXBIT: the length of data transferred (in bit) STOPSEL=0 represents the sampling stops until the length of idle level is reached STOPSEL=1 represents the sampling stops until the length of data is reached
15:7	Reserved			always read as 0
6:4	VTXSEL	rw	0x00	VTXSEL: select the transmit voltage The high level voltage to transmit signal is VDD * n/16.
3	VTXEN	rw	0x01	VTXEN: transmit bleeder circuit enable 1: enabled 0: disabled

Bit	Field	Type	Reset	Description
2	MSEL	rw	0x00	MSEL: select the master/slave channel 1: channel 1 is set as the master channel, and channel 2 is synchronized with channel 1 in transmission and reception 0: channel 2 is set as the master channel, and channel 1 is synchronized with channel 2 in transmission and reception
1	STOPSEL	rw	0x00	STOPSEL: sampling stop condition 1 = the sampling stops until a given length of data is reached 0 = the sampling stops until a given length of idle level is reached; MAXBIT=0 means the sampling keeps on going.
0	STARTSEL	rw	0x00	STARTSEL: start condition detection The start condition is an idle level turns into a non-idle level. 1 = start sampling after this bit is enabled, instead of detecting the start condition 0 = start sampling after detecting the start condition

#### 5.4.10 CSM baud rate generator (CSM\_SPBRG)

Address offset: 0x24

Reset value: 0x00000002															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bit	Field	Type	Reset	Description
31:16	Reserved			always read as 0.
15:0	SPBRG	rw	0x0002	SPBRG: baud rate control register for sampling clock Baud rate formula: Baud rate = $f_{\text{pclk}}/(SPBRG+1)$ ( $f_{\text{pclk}}$ is the APB clock frequency)

#### 5.4.11 CSM data bit count (CSM\_BCNT)

Address offset: 0x28

Reset value: 0x00000000

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

Bit	Field	Type	Reset	Description
31:16	Reserved			always read as 0
15:0	BCNT	rw	0x0000	BCNT: the length of data bits that are currently received or transmitted. This field is read-only.

# 6

# Power Control (PWR)

## Power Control (PWR)

### 6.1 Power supplies

The chip requires a 2.0-to-5.5 V operating voltage supply ( $V_{DD}$ ). An embedded regulator is used to supply the required core power.

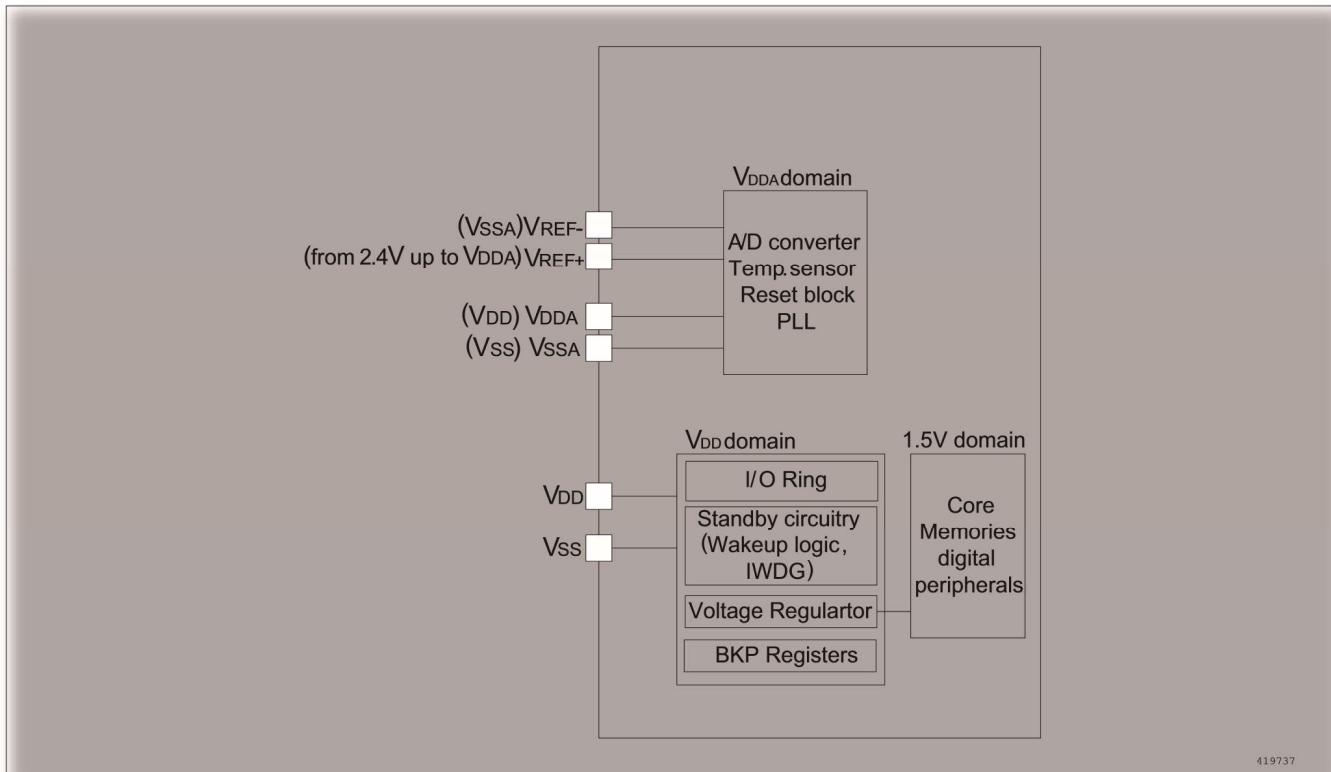


Figure 10. Power supply block diagram

Note:  $V_{DDA}$  and  $V_{SSA}$  should not be floating and the voltage variations between  $V_{DD}$  and  $V_{DDA}$  should be less than 50mV.

#### 6.1.1 Independent A/D converter supply and reference voltage

To improve conversion accuracy, the ADC has an independent power supply which can be separately filtered and shielded from spikes on the PCB.

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

When the  $V_{REF-}$  pin is available (according to package), it must be tied to  $V_{SSA}$ .

### 6.1.2 Battery backup domain

If the 1.5V domain of the regulator is powered off, the content of the backup register can be saved.

### 6.1.3 Voltage regulator

The voltage regulator is always enabled after reset. It works in three different modes depending on the application modes.

- In Run mode, the regulator supplies power to the 1.5V domain (core, memories and peripherals) normally.
- In Stop mode, the regulator supplies low-power to the 1.5 V domain, preserving contents of registers and SRAM.
- In Standby mode, the regulator is powered off. The contents of the registers and SRAM are all lost except for the Standby circuitry and backup domain.

## 6.2 Power supply supervisor

### 6.2.1 Power-on reset (POR)/power-down reset (PDR)

The chip has an integrated POR/PDR circuitry that allows proper operation starting from/down to 2.0V of supply.

The device remains in Reset mode when  $V_{DD}/V_{DDA}$  is below a specified threshold,  $V_{POR}/V_{PDR}$ , without the need for an external reset circuitry. For more details concerning the power-on/power-down reset, refer to the electrical characteristics of the datasheet.

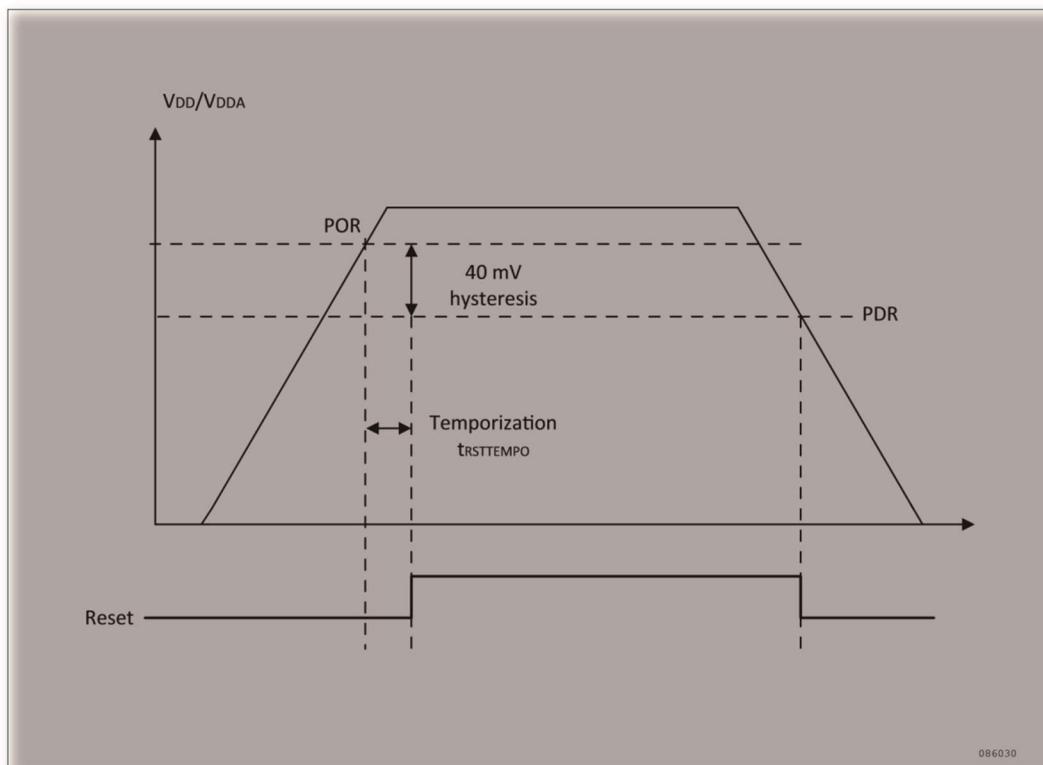


Figure 11. Power-on reset/power-down reset waveform

### 6.2.2 Programmable voltage detector (PVD)

The PVD can be used by the user to monitor the power supply by comparing  $V_{DD}$  to the PLS bits in the power control register (PWR\_CR). These bits can select the threshold of the monitored voltage.

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the power control/status register (PWR\_CSR), to indicate if  $V_{DD}$  is higher or lower than the PVD voltage threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if the interrupt is enabled through the EXTI registers. The PVD interrupt can be generated when  $V_{DD}$  drops below the PVD threshold or when  $V_{DD}$  rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example, the service routine could perform emergency shutdown tasks.

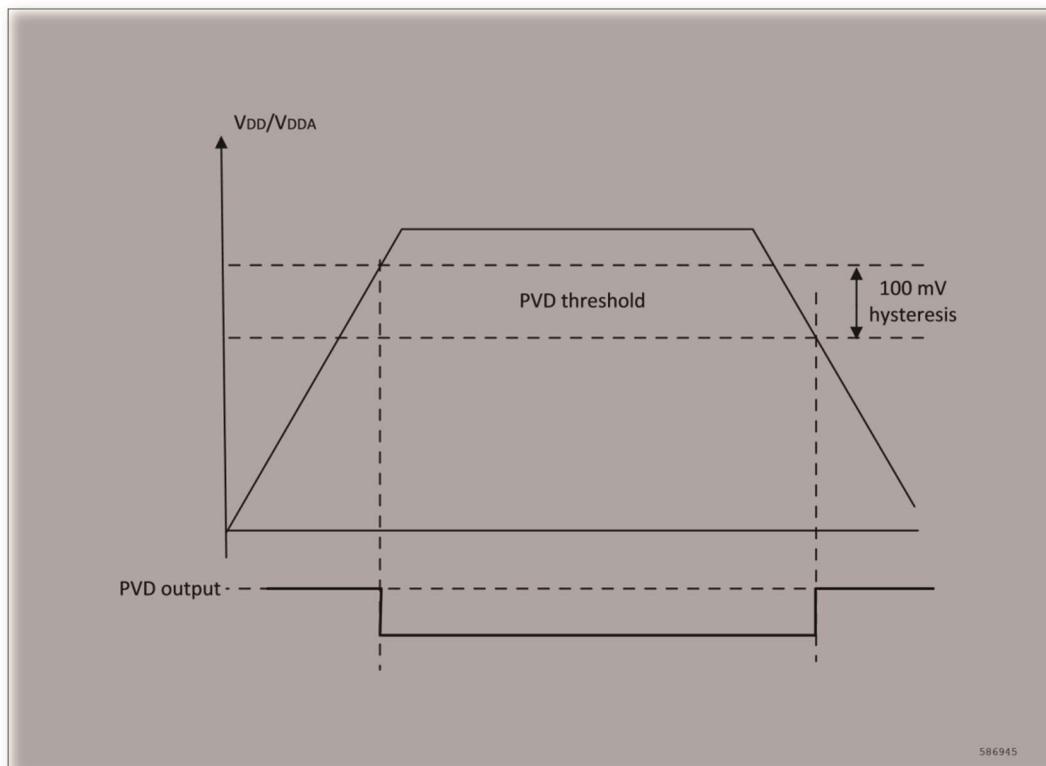


Figure 12. PVD thresholds

## 6.3 Low-power modes

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

The chip features three low-power modes:

- Sleep mode (CPU stops working, but all peripherals including CPU peripherals like NVIC, SysTick, etc. are kept running)
- Stop mode (all clocks are stopped while the SRAM and register contents are kept intact)

- Standby mode (the 1.5V power is off and the contents of the registers and SRAM are all lost except for the standby circuitry and backup domain.)

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

- Slowing down the system clock frequency
- Turning off the clocks on the APB and AHB buses when they are unused

Table 15. Low-power mode summary

Mode	Entry	Wakeup	Effect on 1.5V domain clocks	Effect on V <sub>DD</sub> domain clocks	Voltage regulator
SLEEP (SLEEP NOW or SLEEP ON EXIT)	WFI (Wait for Interrupt)	Any interrupt	CPU clock OFF no effect on other clocks and ADC clock	None	ON
	WFE (Wait for Event)	Wakeup event			
Stop mode	PDDS bit SLEEPDEEP bit WFI or WFE	Any EXTI (configured in the EXTI registers)	All 1.5V domain clocks OFF	PLL, HSI and HSE oscillators OFF	ON
Standby mode	PDDS bit SLEEPDEEP bit WFI or WFE	WKUP pin rising edge, RTC alarm event, external reset in NRST pin, IWDG reset			OFF

### 6.3.1 Slowing down system clocks

In Run mode, the speed of any system clock (SYSCLK, HCLK, PCLK1, PCLK2) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down peripheral clocks before entering Sleep mode.

For more details refer to: Clock configuration register (RCC\_CFGR).

### 6.3.2 Peripheral clock gating

In Run mode, the HCLK and PCLK<sub>x</sub> for individual peripherals and memories can be stopped at any time to reduce power consumption.

To further reduce power consumption in Sleep mode the peripheral clocks can be disabled prior to executing the WFI or WFE instructions.

Peripheral clock gating is controlled by the AHB peripheral clock enable register (RCC\_AHBENR), the APB2 peripheral clock enable register (RCC\_APB2ENR) and the APB1 peripheral clock enable register (RCC\_APB1ENR).

### 6.3.3 Sleep mode

#### Entering Sleep mode

The Sleep mode is entered by executing the WFI or WFE instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the CPU System Control register:

- SLEEP-NOW: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
- SLEEP-ON-EXIT: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority interrupt service routine.

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

Refer to Table 16 and Table 17 for details on how to enter Sleep mode.

Table 16. SLEEP NOW mode

SLEEP NOW mode	Description
Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: - SLEEPDEEP = 0 - SLEEPONEXIT = 0 Refer to the CPU system Control register.
Exit	If WFI was used for sleep mode entry: Interrupt: Refer to the interrupt vector table If WFE was used for sleep mode entry: Wakeup event: Refer to wakeup event management
Wakeup latency	None

Table 17. SLEEP ON EXIT mode

SLEEP ON EXIT mode	Description
Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: - SLEEPDEEP = 0 - SLEEPONEXIT = 1 Refer to the CPU system Control register.
Exit	If WFI was used for sleep mode entry: Interrupt or clear the CPU control register bit 1 If WFE was used for sleep mode entry: Wakeup event: Refer to wakeup event management
Wakeup latency	None

### 6.3.4 Stop mode

The Stop mode is based on the CPU deepsleep mode combined with peripheral clock gating. In Stop mode, the voltage regulator can be configured in normal mode, all clocks in the 1.5V domain are stopped, and the PLL, HSI and HSE oscillators are disabled.

SRAM and register contents are preserved.

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

### Entering Stop mode

Refer to Table 18 for details on how to enter the Stop mode.

The following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its key register or by hardware option. Once enabled, it cannot be disabled again except by a system reset.

- Internal oscillator (LSI oscillator): this is configured by the LSION bit in the control/status register (RCC\_CSR).

The ADC can also consume power during the Stop mode, unless it is disabled before entering it. To disable it, the ADEN bit in the ADC\_ADCFG register must be written to 0. Other unused GPIOs also consume power unless they are configured with analog inputs.

### **Exiting Stop mode**

Refer to Table 18 for details on how to exit the Stop mode.

When exiting Stop mode by issuing an interrupt or a wakeup event, the HSI oscillator is selected as system clock. The clock frequency is the HSI divided by 6.

When the voltage regulator operates in normal power consumption mode, an additional startup delay is incurred when waking up from Stop mode.

Table 18. Stop mode

Stop mode	Description
Entry	<p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> <li>- Set SLEEPDEEP bit in CPU system control register</li> <li>- Clear PDSS bit in power control register (PWR_CR)</li> <li>- Switch the system clock to LSI or HSI</li> </ul> <p>Note: To enter Stop mode, all flags of EXTI request bits (pending register (EXTI_PEND)) must be cleared. Otherwise, the Stop mode entry procedure is ignored and program execution continues.</p>
Exit	<p>WFI (Wait for Interrupt) while:</p> <p>Any EXTI line configured in interrupt mode (the corresponding EXTI vector must be enabled in the NVIC).</p> <p>Refer to the interrupt vector table.</p> <p>WFE (Wait for Event) while:</p> <p>Any EXTI line configured in event mode. Refer to the wakeup event management.</p>
Wakeup latency	LSI or HSI wakeup time

### **6.3.5 Standby mode**

The Standby mode allows to achieve the lowest power consumption. It is based on the CPU deepsleep mode, with the voltage regulator disabled. The 1.5 V domain is consequently powered off. The PLL, HSI and HSE oscillators are also switched off. SRAM and register contents are lost except for registers in the backup domain and Standby circuitry.

### **Entering Standby mode**

Refer to Table 19 for details on how to enter the Standby mode.

In Standby mode, the following features can be selected by setting individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its key register or by hardware option.
- Internal oscillator (LSI oscillator): this is configured by the LSION bit in the control/status register (RCC\_CSR).

### Exiting Standby mode

The microcontroller exits the Standby mode when an external reset (NRST pin), an IWDG reset, or a rising edge on the WKUP pin occurs. All registers are reset after wakeup from Standby except for power control/status register (PWR\_CSR). After waking up from Standby mode, program execution restarts in the same way as after a reset (boot pins sampling, vector reset is fetched, etc.).

The power control/status register (PWR\_CSR) indicates that the core exits the Standby mode.

Refer to Table 19 for more details on how to exit Standby mode.

Table 19. Standby mode

Standby mode	Description
Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: - Set SLEEPDEEP bit in CPU system control register - Set PDDS bit in power control register (PWR_CR) - Clear WUF bit in power control/status register (PWR_CSR)
Exit	WKUP pin rising edge, RTC alarm, external reset in NRST pin, IWDG reset.
Wakeup latency	For voltage regulator startup in the reset phase

### I/O states in Standby mode

In Standby mode, all I/O pins are high impedance except:

- reset pin (still available)
- Enabled WKUP pins

### Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the CPU core is no longer clocked.

However, by setting some configuration bits in the DBGMCU\_CR register, the software can be debugged even when using the low-power modes extensively. For more details, refer to debug support for low-power modes.

## 6.4 Power control registers

Table 20. Overview of power control registers

Offset	Acronym	Register Name	Reset	Section
0x00	PWR_CR	Power control register	0x00000000	Section 6.4.1
0x04	PWR_CSR	Power control/status register	0x00000000	Section 6.4.2

### 6.4.1 Power control register (PWR\_CR)

Address offset: 0x00

Reset value: 0x0000 0000 (cleared by wakeup from Standby mode)

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

Reserved

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

Reserved PLS DBP Reserved PVDE CSBF CWUF PDSS LPDS

rw rw rw rw rw w w rw rw rw rw

Bit	Field	Type	Reset	Description
31:13	Reserved			always read as 0.
12:9	PLS	rw	0x00	<p>PVD level selection</p> <p>These bits are used to select the voltage threshold detected by the power voltage detector</p> <p>0000: 1.8V 0100: 3.0V 1000: 4.2V            0001: 2.1V 0101: 3.3V 1001: 4.5V            0010: 2.4V 0110: 3.6V 1010: 4.8V            0011: 2.7V 0111: 3.9V Others: reserved</p> <p>Note: Refer to the electrical characteristics of the datasheet for more details.</p>
8	DBP	rw	0x00	<p>disable backup domain write protection</p> <p>In reset state, the RTC and Standby registers are protected against accidental write access. This bit can be set to enable write access to these registers.</p> <p>1=write access to RTC and standby registers enabled            0=write access to RTC and standby registers disabled</p> <p>Note: If the RTC clock is HSE/128, this bit must be kept as '1'.</p>
7:5	Reserved			always read as 0
4	PVDE	rw	0x00	<p>Power voltage detector (PVD) enable</p> <p>1=PVD enabled            0=PVD disabled</p>
3	CSBF	w	0x00	<p>Clear standby flag</p> <p>always read as 0</p> <p>1=clear the SBF Standby Flag (write)            0=no effect</p>

2	CWUF	w	0x00	Clear wakeup flag always read as 0 1=clear the WUF wakeup flag after 2 system clock cycles (write) 0=no effect
1	PDDS	rw	0x00	Power down deepsleep 1 = enter Standby mode when the CPU enters Deepsleep 0 = enter Stop mode when the CPU enters Deepsleep
0	LPDS	rw	0x00	Low-power deepsleep It works together with the PDDS bit when PDDS = 0. 1 = voltage regulator in low-power mode during Stop mode 0 = voltage regulator in normal power consumption mode during Stop mode During Stop mode, the current under the condition that LPDS = 1 is smaller than LPDS = 0. For details, refer to the corresponding chip datasheet.

#### 6.4.2 Power control/status register (PWR\_CSR)

Address offset: 0x04

Reset value: 0x0000 0000 (not cleared by wakeup from Standby mode)

Additional APB cycles are needed to read this register versus a standard APB read.

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

Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								EWUP	Reserved				PVDO	SBF	WUF	
								rw					rw	rw	r	

Bit	Field	Type	Reset	Description
31:9	Reserved			always read as 0.
8	EWUP	rw	0x00	Enable WKUP pin 1 = WKUP pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP pin wakes up the system from Standby mode) 0 = WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wake up the CPU from Standby mode. Note: This bit is cleared by a system reset.
7:3	Reserved			always read as 0

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

2	PVDO	rw	0x00	<p>PVD output</p> <p>This bit is valid only if PVD is enabled by the PVDE bit.</p> <p>1 = <math>V_{DD}/V_{DDA}</math> is lower than the PVD threshold selected by the PLS</p> <p>0 = <math>V_{DD}/V_{DDA}</math> is higher than the PVD threshold selected by the PLS</p> <p>Note: The PVD is stopped by Standby mode. For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set.</p>
1	SBF	rw	0x00	<p>Standby flag</p> <p>This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CSBF bit in the power control register (PWR_CR).</p> <p>1 = system has been in Standby mode</p> <p>0 = system has not been in Standby mode</p>
0	WUF	r	0x00	<p>Wakeup flag</p> <p>This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CWUF bit in the power control register (PWR_CR).</p> <p>1 = A wakeup event was received from the WKUP pins or an RTC alarm event occurred</p> <p>0 = no wakeup event occurred</p> <p>Note: An additional event is detected if the WKUP pin is enabled (by setting the EWUP bit) when the WKUP pin level is already high.</p>

**7**

# Backup Registers (BKP)

## Backup Registers (BKP)

### 7.1 BKP introduction

Backup registers are ten 16-bit registers for storing 20 bytes of user application data. They are not reset when the system wakes up from the Standby mode, or by a system reset or power reset.

In addition, the BKP control registers are used to manage the tamper detection feature and RTC calibration. After reset, access to the backup registers and RTC is disabled and the backup domain is protected against possible accidental write access.

To enable access to the backup registers and the RTC, proceed as follows:

- enable the power clock by setting the PWREN bit in the RCC\_APB1ENR register.
- set the DBP bit in the power control register (PWR\_CR) to enable access to the backup registers and RTC.

### 7.2 BKP features

- 20-byte data backup registers
- Status/control register for managing tamper detection with interrupt capability
- Calibration register for storing the RTC calibration value
- Possibility to output the RTC calibration clock, RTC alarm pulse or second pulse on PC13 pin (when this pin is not used for tamper detection)

### 7.3 BKP functional description

#### 7.3.1 Tamper detection

The TAMPER pin generates a Tamper detection event when the pin changes from 0 to 1 or from 1 to 0 depending on the TPAL bit in the backup control register (BKP\_CR). A tamper detection event resets all data backup registers. However to avoid losing tamper events, the signal used for edge detection is logically ANDed with the Tamper enable in order to detect a tamper event in case it occurs before the TAMPER pin is enabled.

- When TPAL=0: If the TAMPER pin is already high before it is enabled (by setting TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no rising edge on the TAMPER pin after TPE was set)
- When TPAL=1: If the TAMPER pin is already low before it is enabled (by setting the TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no falling edge on the TAMPER pin after TPE was set)

By setting the TPIE bit in the BKP\_CSR register, an interrupt is generated when a Tamper detection event occurs. After a Tamper event has been detected and cleared, the TAMPER pin should be disabled and then re-enabled with TPE before writing to the backup data registers again. This prevents software from writing to the backup data registers, while the TAMPER pin value still indicates a Tamper detection. This is equivalent to a level detection on the TAMPER pin.

Note: Tamper detection is still active when V<sub>DD</sub> power is switched off. To avoid unwanted resetting of the data backup registers, the TAMPER pin should be externally tied to the correct level.

### 7.3.2 RTC calibration

For measurement purposes, the RTC clock with a frequency divided by 64 can be output on the TAMPER pin. This is enabled by setting the CCO bit in the RTC calibration register (BKP\_RTCCR). The clock can be slowed down by up to 121 ppm by configuring CAL[6:0] bits.

## 7.4 BKP register description

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

Table 21. Overview of BKP registers

Offset	Acronym	Register Name	Reset	Section
0x50 + 4 × (n - 1)	BKP_DRn	Backup data register x	0x00000000	Section 7.4.1
0x40	BKP_RTCCR	RTC clock calibration register	0x00000000	Section 7.4.2
0x44	BKP_CR	Backup control register	0x00000000	Section 7.4.3
0x48	BKP_CSR	Backup control/status register	0x00000000	Section 7.4.4

### 7.4.1 Backup data register n(BKP\_DRn)(n = 1...20)

Address offset: 0x50 + 4 × (serial number of backup data register - 1)

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

BKP															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:0	BKP	rw	0x0000	<p>BKP: Backup data</p> <p>These bits can be written with user data.</p> <p>Note: The BKP_DRn registers are not reset by a system reset or power reset or when the device wakes up from Standby mode. They are reset by a Backup Domain reset or by a TAMPER pin event (if the TAMPER pin function is activated).</p>

### 7.4.2 RTC clock calibration register (BKP\_RTCCR)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					ASOS	ASOE	CCO	CAL							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
9	ASOS	rw	0x00	<p>ASOS: Alarm or second output selection</p> <p>When the ASOE bit is set, the ASOS bit can be used to select whether the signal output on the TAMPER pin is the RTC Second pulse signal or the Alarm pulse signal.</p> <p>0: RTC Alarm pulse output selected 1: RTC Second pulse output selected</p> <p>Note: This bit is reset only by a backup domain reset.</p>
8	ASOE	rw	0x00	<p>ASOE: Alarm or second output enable</p> <p>Setting this bit outputs either the RTC Alarm pulse signal or the Second pulse signal on the TAMPER pin depending on the ASOS bit.</p> <p>The output pulse duration is one RTC clock period. The TAMPER pin must not be enabled while the ASOE bit is set.</p> <p>Note: This bit is reset only by a backup domain reset.</p>
7	CCO	rw	0x00	<p>CCO: Calibration clock output</p> <p>0: No effect 1: Setting this bit outputs the RTC clock with a frequency divided by 64 on the TAMPER pin. The TAMPER pin must not be enabled while the CCO bit is set in order to avoid unwanted Tamper detection.</p> <p>Note: This bit is reset when the V<sub>DD</sub> supply is powered off.</p>
6:0	CAL	rw	0x00	<p>CAL: Calibration value</p> <p>This value indicates the number of clock pulses that will be ignored every 2<sup>20</sup> clock pulses. This allows the calibration of the RTC, slowing down the clock by steps of 1000000/2<sup>20</sup> PPM.</p> <p>The clock of the RTC can be slowed down from 0 to 121PPM.</p>

15:10      Reserved                          always read as 0.

### 7.4.3 Backup control register (BKP\_CR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TPAL	TPE
														rw	rw

Bit	Field	Type	Reset	Description
15:2	Reserved			always read as 0.
Bit	Field	Type	Reset	Description

1	TPAL	rw	0x00	TPAL: TAMPER pin active level 0: A high level on the TAMPER pin resets all data backup registers (if TPE bit is set). 1: A low level on the TAMPER pin resets all data backup registers (if TPE bit is set).
0	TPE	rw	0x00	TPE: TAMPER pin enable 0: The TAMPER pin is free for general purpose I/O 1: Tamper pin enabled for tamper detection

Note: Setting the TPAL and TPE bits at the same time is always safe, however resetting both at the same time can generate a spurious Tamper event. For this reason, it is recommended to change the TPAL bit only when the TPE bit is reset.

#### 7.4.4 Backup control/status register (BKP\_CSR)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TIF	TEF	Reserved			TPIE	CTI	CTE				
				r	r				rw	w	w				

Bit	Field	Type	Reset	Description
15:10	Reserved			always read as 0.
9	TIF	r	0x90	TIF: Tamper interrupt flag This bit is set by hardware when a Tamper event is detected and the TPIE bit is set. It is cleared by writing 1 to the CTI bit (also clears the interrupt). It is also cleared if the TPIE bit is reset. 0: No Tamper interrupt 1: A Tamper interrupt occurred Note: This bit is reset only by a system reset or wakeup from Standby mode.
8	TEF	r	0x00	TEF: Tamper event flag This bit is set by hardware when a Tamper event is detected. It is cleared by writing '1' to the CTE bit. 0: No Tamper event 1: A Tamper event occurred Note: A Tamper event resets all the BKP_DRn registers. They are held in reset as long as the TEF bit is set. If a write to the BKP_DRn registers is performed while this bit is set, the value will not be stored.
7:3	Reserved			always read as 0
2	TPIE	rw	0x00	TPIE: TAMPER pin interrupt enable

0: Tamper interrupt disabled

1: Tamper interrupt enabled (the TPE bit must also be set in the BKP\_CR register)

Note 1: A Tamper interrupt does not wake up the core from low-power modes.

Note 2: This bit is reset only by a system reset or wakeup from Standby mode.

1	CTI	w	0x00	CTI: Clear tamper interrupt This bit is write only, and is always read as 0. 0: No effect 1: Clear the Tamper interrupt and the TIF Tamper interrupt flag
0	CTE	w	0x00	CTE: Clear tamper event This bit is write only, and is always read as 0. 0: No effect 1: Reset the TEF Tamper event flag (and the Tamper detector)

## Real-Time Clock (RTC)

### 8.1 RTC introduction

---

The real-time clock is an independent timer. The RTC module provides a set of continuously running counters which can be used, with suitable software configuration, to provide a clock-calendar function. The counter values can be written to set the current time and date of the system.

The RTC module and clock configuration system (RCC\_BDCR register) are in the backup area, which means RTC setting and time are kept after system reset or wakeup from Standby mode.

After reset, access to the backup registers and RTC is disabled and the backup domain (BKP) is protected against possible parasitic write access. To enable access to the backup registers and the RTC, proceed as follows:

- enable the power clock by setting the PWREN bit in the RCC\_APB1ENR register.
- set the DBP bit in the power control register PWR\_CR to enable access to the backup registers and RTC.

### 8.2 Main features

---

- Programmable prescaler: division factor up to  $2^{20}$
- 32-bit programmable counter for long-term measurement
- Two separate clocks: PCLK1 for the APB1 interface and RTC clock (RTC clock must be at least four times slower than the PCLK1 clock)
- The RTC clock source could be any of the following ones:
  - HSE clock divided by 128
  - LSE oscillator clock
  - LSI oscillator clock
- Two separate reset types:
  - The APB1 interface is reset by system reset
  - The RTC core (prescaler, alarm, counter and divider) is reset only by a backup domain reset
- Three dedicated maskable interrupt lines:
  - Alarm interrupt, for generating a software programmable alarm interrupt.
  - Seconds interrupt, for generating a periodic interrupt signal with a programmable period length (up to 1 second).
  - Overflow interrupt, to detect when the internal programmable counter rolls over to zero.

### 8.3 Functional description

---

### 8.3.1 Overview

The RTC consists of two main units (see the figure below). The first one (APB1 Interface) is used to interface with the APB1 bus. This unit also contains a set of 16-bit registers accessible from the APB1 bus in read or write mode. The APB1 interface is clocked by the APB1 bus clock in order to interface with the APB1 bus.

The other unit (RTC core) consists of a chain of programmable counters made of two main blocks. The first block is the RTC prescaler block, which generates the RTC time base TR\_CLK that can be programmed to have a period of up to 1 second. It includes a 20-bit programmable divider (RTC prescaler). Every TR\_CLK period, the RTC generates an interrupt (second interrupt) if it is enabled in the RTC\_CR register. The second block is a 32-bit programmable counter that can be initialized to the current system time. The system time is incremented at the TR\_CLK rate and compared with a programmable date (stored in the RTC\_ALR register) in order to generate an alarm interrupt, if enabled in the RTC\_CR control register.

Here is a simplified RTC block diagram.

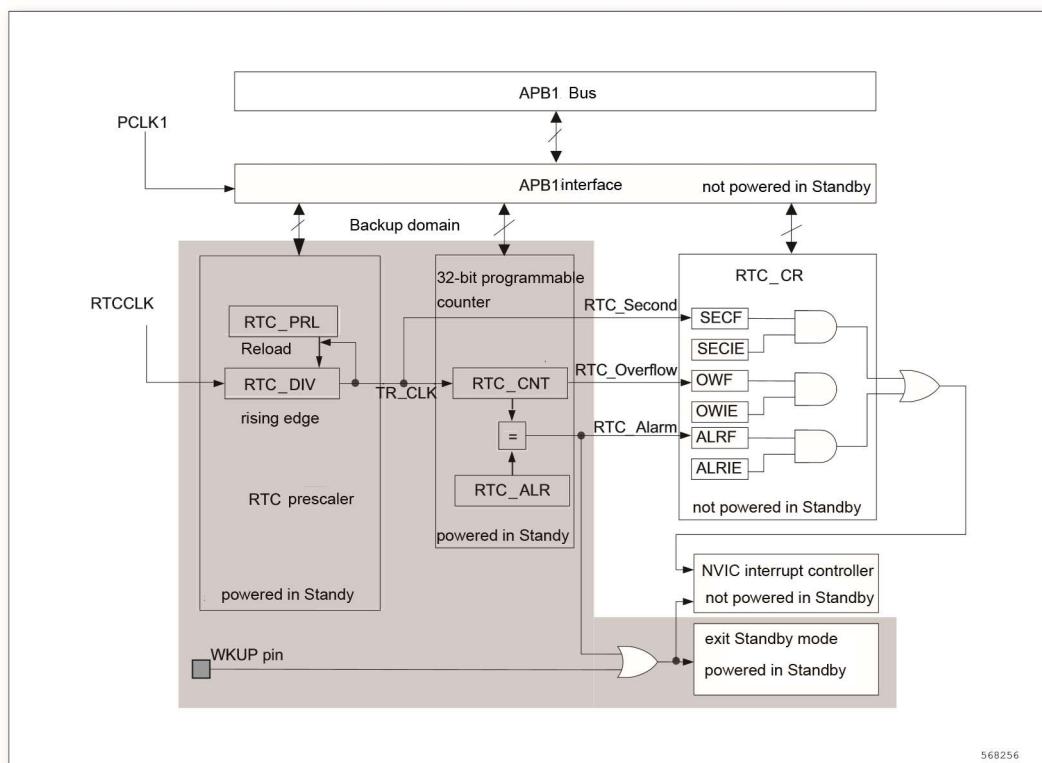


Figure 13. RTC block diagram

### 8.3.2 Resetting RTC registers

All system registers are asynchronously reset by a System Reset or Power Reset, except for RTC\_PRL, RTC\_ALR, RTC\_CNT, and RTC\_DIV.

The RTC\_PRL, RTC\_ALR, RTC\_CNT, and RTC\_DIV registers are reset only by a Backup Domain reset.

### 8.3.3 Reading RTC registers

The RTC core is completely independent from the RTC APB1 interface.

Software accesses the RTC prescaler, counter and alarm values through the APB1 interface but the associated readable registers are internally updated at each rising edge of the RTC clock resynchronized by the RTC APB1 clock. This is also true for the RTC flags.

This means that the first read to the RTC APB1 registers may be corrupted (generally read as 0) if the APB1 interface has previously been disabled and the read occurs immediately after the APB1 interface is enabled but before the first internal update of the registers.

This can occur if:

- A system reset or power reset has occurred
- The MCU has just woken up from Standby mode
- The MCU has just woken up from Stop mode

In all the above cases, the RTC core has been kept running while the APB1 interface was disabled (reset, not clocked or unpowered).

Consequently when reading the RTC registers, after having disabled the RTC APB1 interface, the software must first wait for the RSF bit (Register Synchronized Flag) in the RTC\_CRL register to be set by hardware.

Note that the RTC APB1 interface is not affected by WFI and WFE low-power modes.

### 8.3.4 Configuring RTC registers

To write in the RTC\_PRL, RTC\_CNT, RTC\_ALR registers, the CNF bit must be set in the RTC\_CRL register to allow the RTC enter the configuration mode.

In addition, writing to any RTC register is only enabled if the previous write operation is finished. To enable the software to detect this situation, the RTOFF status bit is provided in the RTC\_CR register to indicate that an update of the registers is in progress. A new value can be written to the RTC registers only when the RTOFF status bit value is '1'.

#### Configuration procedure:

- Poll RTOFF, wait until its value goes to '1'
- Set the CNF bit to enter configuration mode
- Write to one or more RTC registers
- Clear the CNF bit to exit configuration mode
- Poll RTOFF, wait until its value goes to '1' to check the end of the write operation
- The write operation only executes when the CNF bit is cleared; it takes at least three RTCCLK cycles to complete

### 8.3.5 RTC flag assertion

The RTC second flag (SECF) is asserted on each RTC core clock cycle before the update of the RTC Counter.

- The RTC\_ALARM flag is asserted when the RTC\_CNT value is equal to RTC\_ALR and the RTC\_PR restarts counting from 0 after an overflow.
- The RTC\_ALARM flag is asserted when the RTC\_CNT value is equal to RTC\_MSR and the RTC\_PR restarts counting from 0 after an overflow.
- The RTC\_ALARM flag is asserted when RTC\_CNT value is equal to RTC\_ALR and the RTC\_PR counts to MSR after an overflow.

The RTC Overflow flag (OWF) is asserted on the last RTC core clock cycle before the counter reaches 0x0000.

The RTC\_Alarm and RTC Alarm flag (ALRF) are asserted on the last RTC core clock cycle before the counter reaches the value stored in the Alarm register increased by one (RTC\_ALR

+ 1). The write operation in the RTC Alarm and RTC Second flag must be synchronized by using one of the following sequences:

- Use the RTC Alarm interrupt and inside the RTC interrupt routine, the RTC Alarm and/or RTC Counter registers are updated.
- Wait for SECF bit to be set in the RTC Control register. Update the RTC Alarm and/or the RTC Counter register.

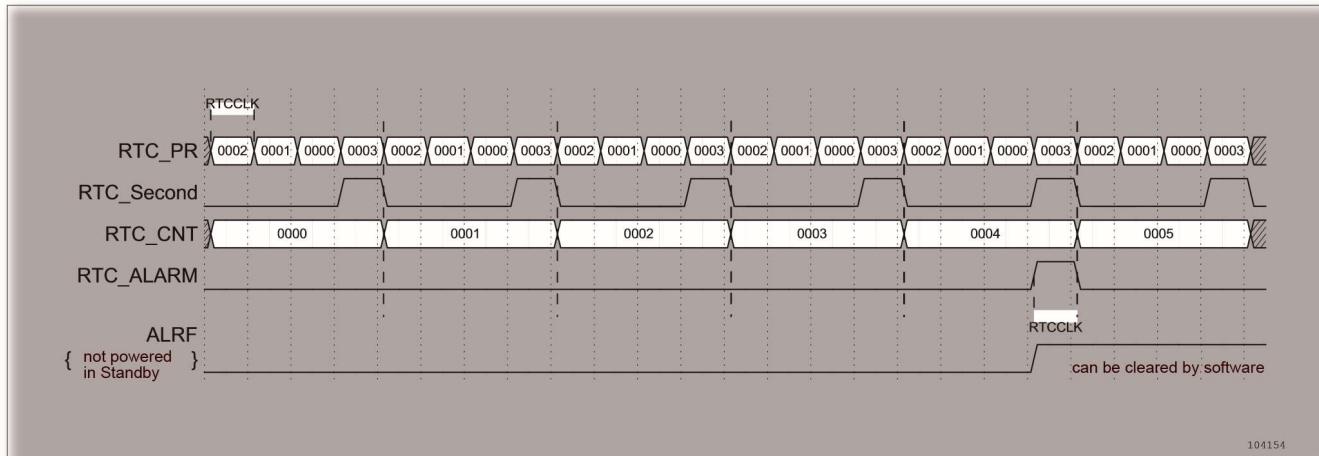


Figure 14. RTC second and alarm waveform example with PR=0003, ALARM=00004

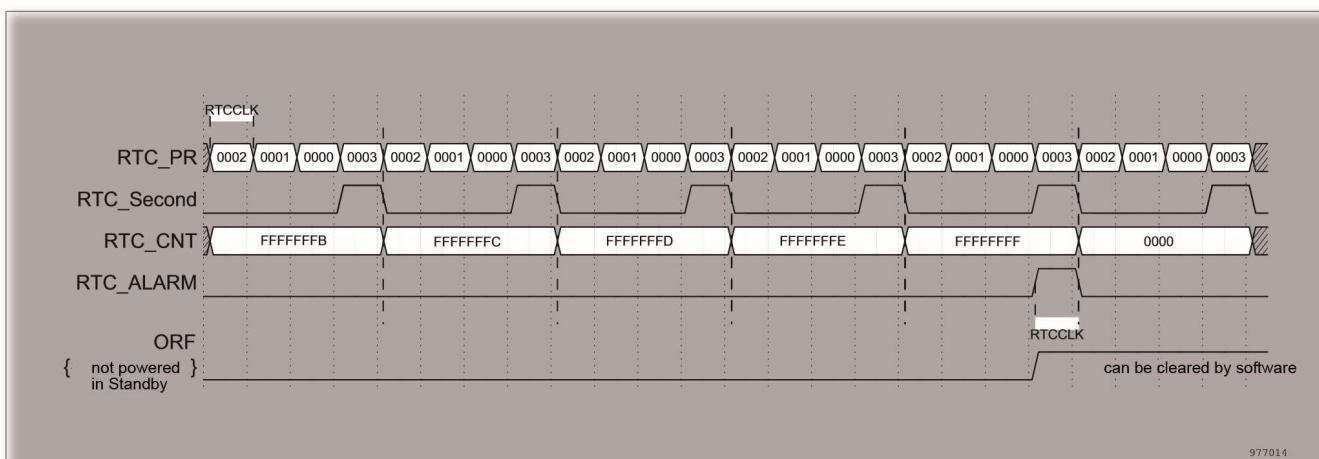


Figure 15. RTC Overflow waveform example with PR=0003

## 8.4 RTC alarm description

RTC alarm consists of two parts:

Millisecond timing alarm: The RTC\_ALR register is required to be 0. The RTC\_MSR should be configured to set the time in millisecond. The alarm flag will be set when the time is up.

Second timing alarm: The RTC\_MSR register is required to be 0. The RTC\_ALR should be configured to set the time in second. The alarm flag will be set when the time is up.

The Millisecond Alarm and Second Alarm can be used at the same time to set a time not in multiple second.

RTC clock loop: The ALPEN in the RTC\_CRL can be configured to set a single alarm/an alarm loop.

Table 22. Overview of RTC registers

Offset	Acronym	Register Name	Reset	Section
0x00	RTC_CRH	RTC control register high	0x00000000	Section 8.4.1
0x04	RTC_CRL	RTC control register low	0x00000020	Section 8.4.2
0x08	RTC_PRLH	RTC prescaler load register high	0x00000000	Section 8.4.3
0x0C	RTC_PRLL	RTC prescaler load register low	0x00000000	Section 8.4.3
0x10	RTC_DIVH	RTC prescaler divider register high	0x00000000	Section 8.4.4
0x14	RTC_DIVL	RTC prescaler divider register low	0x00000000	Section 8.4.4
0x18	RTC_CNTH	RTC counter register high	0x00000000	Section 8.4.5
0x1C	RTC_CNTL	RTC counter register low	0x00000000	Section 8.4.5
0x20	RTC_ALRH	RTC alarm register high	0x0000FFFF	Section 8.4.6
0x24	RTC_ALRL	RTC alarm register low	0x0000FFFF	Section 8.4.6
0x28	RTC_MSRH	RTC millisecond register high	0x00000000	Section 8.4.7
0x2C	RTC_MSRL	RTC millisecond register low	0x00000000	Section 8.4.7

#### 8.4.1 RTC control register

##### (RTC\_CRH)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													OWIE	ALRIE	SECIE
rw		rw		rw		rw		rw		rw		rw		rw	

Bit	Field	Type	Reset	Description
15:3	Reserved			always read as 0.
2	OWIE	rw	0x00	Overflow interrupt enable 0: Overflow interrupt is masked (disabled). 1: Overflow interrupt is enabled.
1	ALRIE	rw	0x00	Alarm interrupt enable 0: Alarm interrupt is masked (disabled). 1: Alarm interrupt is enabled.

Bit	Field	Type	Reset	Description
0	SECIE	rw	0x00	Second interrupt enable 0: Second interrupt is masked (disabled). 1: Second interrupt is enabled.

These bits are used to mask interrupt requests.

Note that at reset all interrupts are disabled, so it is possible to write to the RTC registers to ensure that no interrupt requests are pending after initialization. It is not possible to write to the RTC\_CRH register when the peripheral is completing a previous write operation (flag bit RTOFF = 0).

The RTC functions are controlled by this control register. Some bits must be written using a specific configuration procedure.

#### 8.4.2 RTC control register low (RTC\_CRL)

Address offset: 0x04

Reset value: 0x0020

15	14	13	12	11	10	9	8	16	5	4	3	2	1	0	
								Reserved	ALPEN	RTOFF	CNF	RSF	OWF	ALRF	SECF
									rw	r	rw	rc_w0	rc_w0	rc_w0	rc_w0

Bit	Field	Type	Reset	Description
15:7	Reserved			always read as 0.
6	ALPEN	rw	0x00	RTC Alarm Loop Enable 0: a single alarm event 1: an alarm loop event
5	RTOFF	r	0x01	RTC operation OFF With this bit the RTC reports the status of the last operation performed on its registers, indicating if it has been completed or not. If its value is '0' then it is not possible to write to/read any of the RTC registers. This bit is read only. 0: Last write operation on RTC registers is still ongoing. 1: Last write operation on RTC registers terminated.
4	CNF	rw	0x00	CNF: Configuration flag This bit must be set by software to enter in configuration mode so as to allow new values to be written in the RTC_CNTL/H, RTC_ALRL/H or RTC_PRLL/H registers. The write operation is only executed when the CNF bit is reset by software after has been set. 0: Exit configuration mode (start update of RTC registers). 1: Enter configuration mode.

Bit	Field	Type	Reset	Description

3	RSF	rc_w0	0x00	<p>Registers synchronized flag</p> <p>This bit is set by hardware at each time the RTC_CNT and RTC_DIV registers are updated and cleared by software. After an APB1 reset or an APB1 clock stop, this bit must be cleared by software. Before any read operation, the user application must wait until it is set to be sure that the RTC_CNT, RTC_ALR or RTC_PRL registers are synchronized.</p> <p>0: Registers not yet synchronized. 1: Registers synchronized.</p>
2	OWF	rc_w0	0x00	<p>Overflow flag</p> <p>This bit is set by hardware when the 32-bit programmable counter overflows. An interrupt is generated if OWIE = 1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect.</p> <p>0: Overflow not detected. 1: 32-bit programmable counter overflow occurred.</p>
1	ALRF	rc_w0	0x00	<p>Alarm flag</p> <p>This bit is set by hardware when the 32-bit programmable counter reaches the threshold set in the RTC_ALR register. An interrupt is generated if ALRIE = 1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect.</p> <p>0: Alarm not detected 1: Alarm detected</p>
0	SECF	rc_w0	0x00	<p>Second flag</p> <p>This bit is set by hardware when the 32-bit programmable prescaler overflows, thus incrementing the RTC counter by 1. Hence this flag provides a periodic signal with a period corresponding to the resolution programmed for the RTC counter (usually one second). An interrupt is generated if SECIE = 1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect.</p> <p>0: Second flag condition not met. 1: Second flag condition met.</p>

The functions of the RTC are controlled by this control register. It is not possible to write to the RTC\_CR register while a previous write operation is not completed (when RTOFF = 0).

Note: 1. Any flag remains pending until the appropriate RTC\_CR request bit is reset by software, indicating that the interrupt request has been granted.

2. At reset the interrupts are disabled, no interrupt requests are pending and it is possible to write to the RTC registers.
3. The OWF, ALRF, SECF and RSF bits are not updated when the APB1 clock is not running.
4. The OWF, ALRF, SECF and RSF bits can only be set by hardware and only cleared by software.
5. If ALRF = 1 and ALRIE = 1, the RTC global interrupt is enabled. If EXTI Line 17 is also enabled through the EXTI control register, both the RTC global interrupt and the RTC Alarm interrupt are enabled.

6. If ALRF = 1, the RTC Alarm interrupt is enabled if EXTI Line 17 is enabled through the EXTI Controller in interrupt mode. When the EXTI Line 17 is enabled in event mode, a pulse is generated on this line (no RTC Alarm interrupt generation).

### 8.4.3 RTC prescaler load register (RTC\_PRLH/RTC\_PRLL)

RTC prescaler load register high (RTC\_PRLH)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PRL			
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15:4	Reserved			always read as 0.
3:0	PRL	w	0x00	RTC prescaler reload value high These bits are used to define the counter clock frequency according to the following formula: $f_{TR\_CLK} = f_{RTCCLK}/(PRL + 1)$ Note: The zero value is not recommended, as RTC interrupts and flags cannot be asserted correctly.

### RTC prescaler load register low (RTC\_PRLL)

Address offset: 0x0C

Reset value: 0x0000 (this value will update to 0x0000 after the RTC clock is enabled)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRL															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15:0	PRL	w	0x0000	RTC prescaler reload value low These bits are used to define the counter clock frequency according to the following formula: $f_{TR\_CLK} = f_{RTCCLK}/(PRL + 1)$

Note: If the input clock frequency is 32.768KHz ( $f_{RTCCLK}$ ), write 7FFFh in this register to get a signal period of 1 second.

### 8.4.4 RTC prescaler divider register (RTC\_DIVH/RTC\_DIVL)

During each period of TR\_CLK, the counter inside the RTC prescaler is reloaded with the value stored in the RTC\_PRL register. To get an accurate time measurement it is possible to read the current value of the prescaler counter, stored in the RTC\_DIV register, without stopping it. This register is read-only and it is reloaded by hardware after any change in the RTC\_PRL or RTC\_CNT registers.

**RTC prescaler divider register high (RTC\_DIVH)**

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV														r	r

Bit	Field	Type	Reset	Description
15:4	Reserved			always read as 0
3:0	DIV	r	0x00	RTC clock prescaler divider high

**RTC prescaler divider register low (RTC\_DIVL)**

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV														r	r

Bit	Field	Type	Reset	Description
15:0	DIV	r	0x0000	RTC clock prescaler divider low

**8.4.5 RTC counter register (RTC\_CNTH/RTC\_CNTL)**

The RTC core has one 32-bit programmable counter, accessed through two 16-bit registers; the count rate is based on the TR\_CLK time reference, generated by the prescaler. RTC\_CNT registers keep the counting value of this counter. They are write-protected by bit RTOFF in the RTC\_CR register, and a write operation is allowed if the RTOFF value is '1'. A write operation on the upper (RTC\_CNTH) or lower (RTC\_CNTL) registers directly loads the corresponding programmable counter and reloads the RTC Prescaler. When reading, the current value in the counter (system date) is returned.

**RTC counter register high (RTC\_CNTH)**

Address offset: 0x18

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description
15:0	CNT	r	0x0000	RTC counter register high

15:0	CNT	rw	0x0000	RTC counter high Reading the RTC_CNTH register, the current value of the high part of the RTC Counter register is returned. To write to this register, it is necessary to enter configuration mode.
------	-----	----	--------	--

**RTC counter register low (RTC\_CNTL)**

Address offset: 0x1C

Reset value: 0x0000

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

CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:0	CNT	rw	0x0000	RTC counter low Reading the RTC_CNTL register, the current value of the lower part of the RTC Counter register is returned. To write to this register, it is necessary to enter configuration mode.

**8.4.6 RTC alarm register (RTC\_ALRH/RTC\_ALRL)**

When the programmable counter reaches the 32-bit value stored in the RTC\_ALR register, an alarm is triggered and the RTC alarm interrupt is generated. This register is write-protected by the RTOFF bit in the RTC\_CR register, and a write operation is allowed if the RTOFF value is '1'.

**RTC alarm register high (RTC\_ALRH)**

Address offset: 0x20

Reset value: 0xFFFF

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

ALR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:0	ALR	rw	0xFFFF	RTC alarm high This register protects the high part of the alarm time written by software. To write to this register, it is necessary to enter configuration mode

**RTC alarm register low (RTC\_ALRL)**

Address offset: 0x24

Reset value: 0xFFFF

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

ALR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Real-time Clock (RTC)										UM_MM32F013x_Ver1.00							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description													
15:0	ALR	rw	0xFFFF	RTC counter low	This register protects the low part of the alarm time written by software. To write to this register, it is necessary to enter configuration mode.												

#### 8.4.7 RTC millisecond register (RTC\_MSRH/RTC\_MSRL)

If RTC\_ALR = 0 and RTC\_MSR ≠ 0: When the programmable counter reaches the 32-bit value stored in the RTC\_MSR register, a millisecond event is triggered and the RTC alarm interrupt is generated;

If RTC\_ALR ≠ 0 and RTC\_MSR ≠ 0: When a millisecond event is triggered after a second event is triggered, the RTC alarm interrupt is generated;

If RTC\_ALR ≠ 0 and RTC\_MSR = 0: When a second event is triggered, the RTC alarm interrupt is generated;

This register is write-protected by the RTOFF bit in the RTC\_CR register, and a write operation is allowed if the RTOFF value is '1'.

##### RTC millisecond alarm register high (RTC\_MSRH)

Address offset: 0x28

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description												
15:4	Reserved			always read as 0.												
3:0	MSR	rw	0x00	RTC msec high This register protects the high part of the millisecond alarm time written by software. To write to this register, it is necessary to enter configuration mode.												

##### RTC millisecond alarm register low (RTC\_MSRL)

Address offset: 0x2C

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description												
15:0	MSR	rw	0x0000	RTC msec low This register protects the low part of the millisecond alarm time written by software. To write to this register, it is necessary to enter configuration mode.												

# 9

# Reset and Clock Control (RCC)

## Reset and Clock Control (RCC)

### 9.1 Reset

There are three types of reset, defined as system reset, power-on reset and backup domain reset.

#### 9.1.1 System reset

A system reset sets all registers to their reset values except the reset flag and internal low-speed oscillator enable flag in the clock control register (RCC\_CSR), the Standby and Wakeup flags in the power control register (PWR\_CSR) and the DBG control register (DBG\_CR).

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

1. A low level on the nRST pin (external reset)
2. Window watchdog end of count condition (WWDG reset)
3. Independent watchdog end of count condition (IWDG reset)
4. A software reset (SW reset)
5. A CPU deadlock reset
6. A PVD reset

The reset source can be identified by checking the reset flags in the RCC\_CSR control status register.

#### Software reset

A software reset is available if the SYSRESETREQ bit in the CPU application interrupt and reset control register (AIRCR) is set to '1'.

#### 9.1.2 Power reset

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

1. Power-on/power-down reset (POR/PDR reset)
2. When exiting Standby mode

A power reset sets all registers to their reset values except the Backup domain.

The reset source in Figure 16 will act on the RESET pin and it is always kept at a low level during the reset phase. The RESET service routine vector is fixed at address 0x0000\_0004.

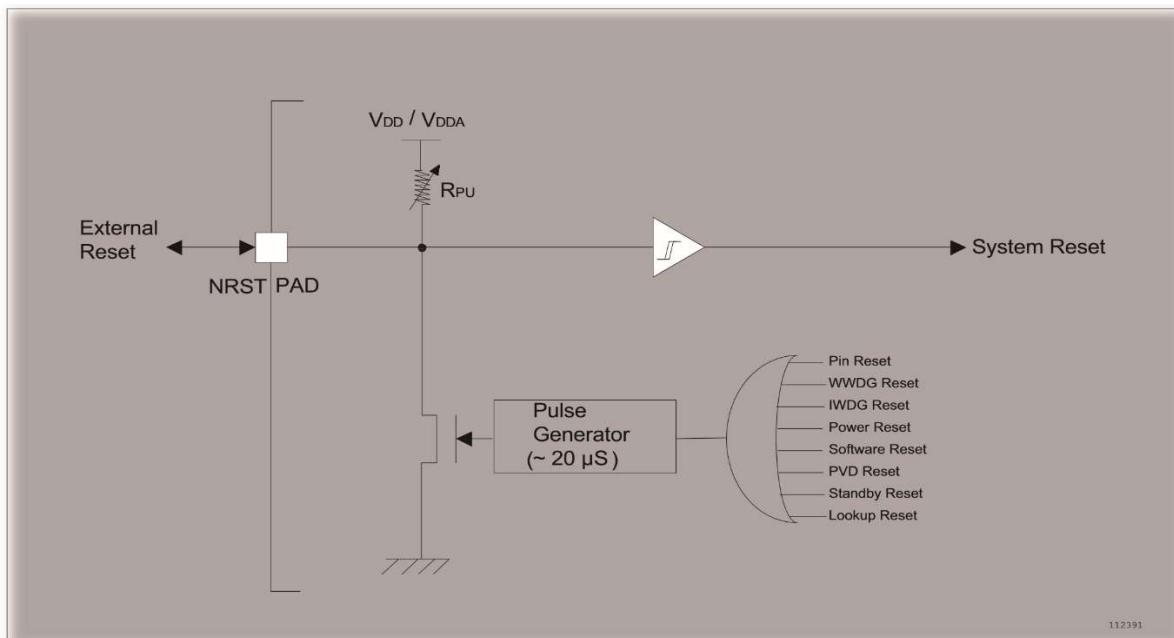


Figure 16. Reset circuit

### 9.1.3 Backup domain reset

The backup domain has specific resets that affect only the backup domain. A backup domain reset is generated by the BDRST bit in the backup domain control register RCC\_BDCR, a power reset or VDD power down.

## 9.2 Clocks

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI oscillator clock divided by 6
- HSE oscillator clock
- PLL clock
- LSI clock

Each clock source can be switched on or off independently when it is not used, to optimize system power consumption.

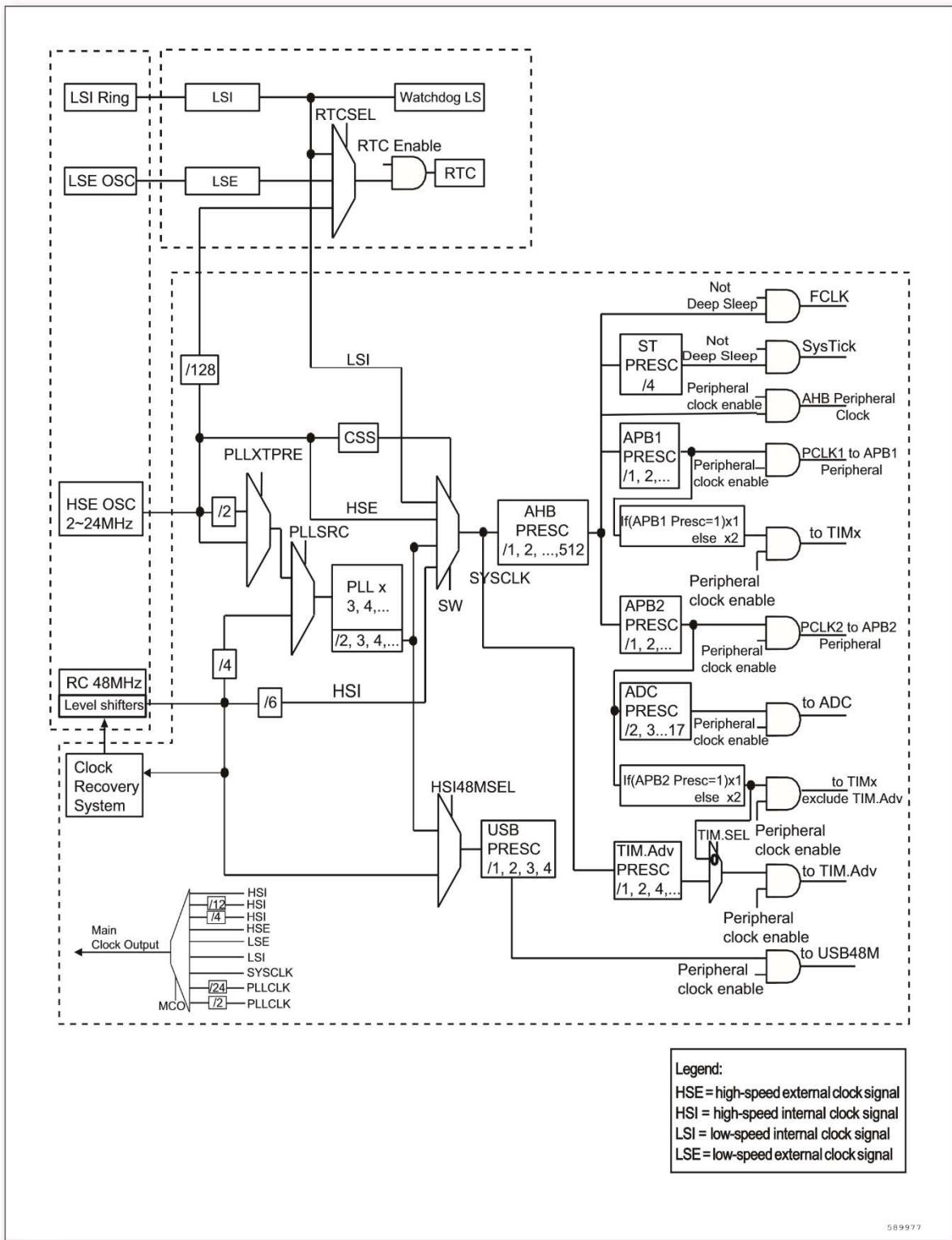


Figure 17. Clock tree

Several prescalers allow the configuration of the AHB frequency, the high speed APB (APB2) and the low speed APB (APB1) domains. The maximum frequency of the AHB, APB1 and APB2 domains is 72MHz.

The RCC feeds the CPU system timer (SysTick) external clock with the AHB clock divided by 4. The SysTick can work either with this clock or with the AHB clock, configurable in the SysTick control and status register. The ADCs are clocked by the APB2 clock divided.

The timer clock frequencies are automatically fixed by hardware. There are two cases:

1. If the APB prescaler coefficient is 1, the timer clock frequencies are set to the same frequency as that of the APB bus to which the timers are connected.
2. Otherwise, they are set to twice ( $\times 2$ ) the frequency of the APB bus to which the timers are connected.

FCLK acts as CPU free-running clock.

### 9.2.1 HSE clock

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

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

The crystal/ceramic resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

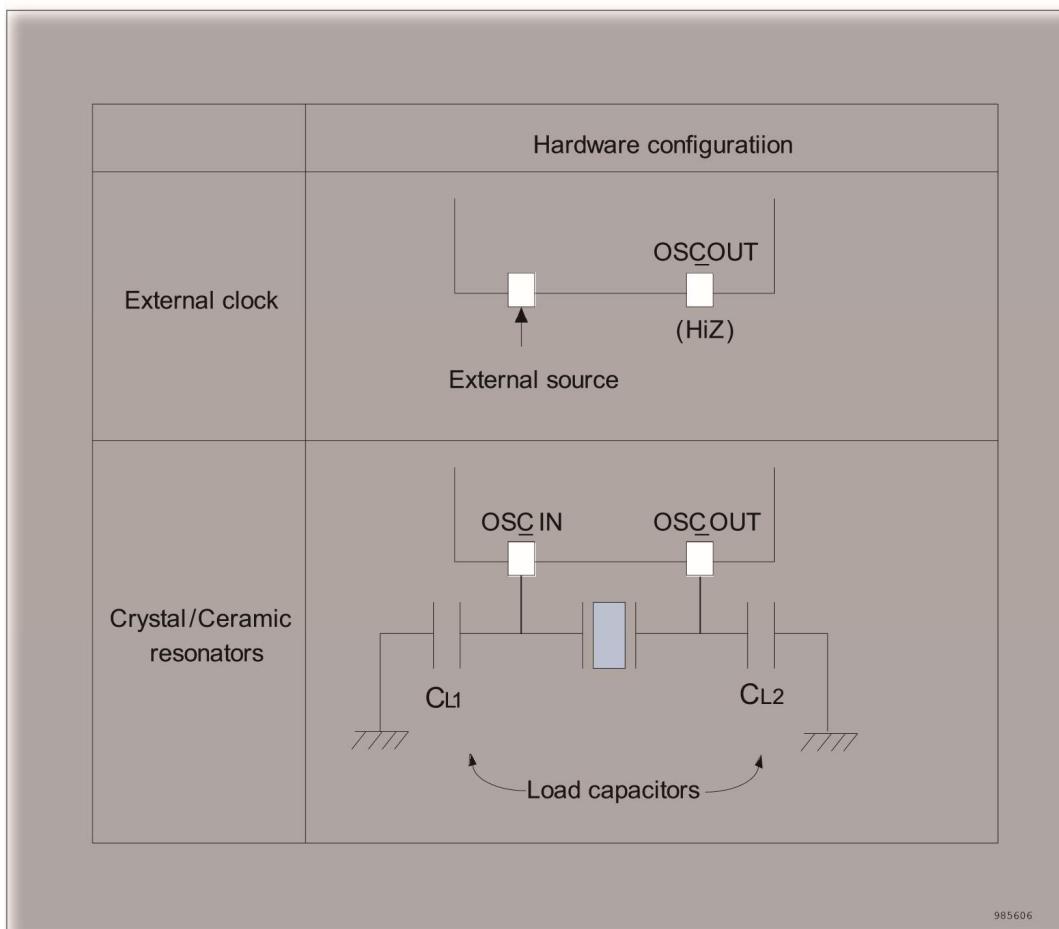


Figure 18. Clock source

#### External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 24MHz. You select this mode by setting the HSEBYP and HSEON bits in the clock control register. The external clock signal (square or sinus with ~50% duty cycle) has to drive the OSC\_IN pin while the OSC\_OUT pin should not be connected.

#### External crystal/ceramic resonator (HSE crystal)

The external oscillator has the advantage of producing a very accurate rate on the main clock. The associated hardware configuration is shown in Figure 18. Refer to the electrical characteristics of the datasheet for more details.

The HSERDY flag in the clock control register (RCC\_CR) indicates if the high-speed external oscillator is stable or not. At startup, the clock is not released until this bit is set to '1' by hardware. An interrupt can be generated if enabled in the clock interrupt register (RCC\_CIR).

The HSE crystal can be switched on and off using the HSEON bit in the clock control register (RCC\_CR).

### 9.2.2 HSI clock

The HSI clock signal is generated from an internal HSI oscillator and can be used directly as a system clock or PLL input. The HSI oscillator has the advantage of providing a system clock source without any external components. It also has a faster startup time than the HSE crystal oscillator however, even with calibration its frequency is less accurate.

#### Calibration

Oscillator frequencies of chips can vary because of manufacturing process variations, this is why the HSI clock frequency of each chip is factory calibrated for 1% accuracy at 25°C. After reset, the factory calibration value is loaded in the HSICAL flag in the clock control register.

The HSIRDY flag in the clock control register indicates if the HSI oscillator is stable or not. At clock startup, the HSI oscillator output clock is not released until this bit is set by hardware to '1'. The HSI oscillator can be switched on and off using the HSION bit in the clock control register.

The HSI clock can also be used as a backup source if the HSE crystal oscillator fails. Refer to section 9.2.7.

### 9.2.3 PLL

The internal PLL can be used to multiply the HSI oscillator output clock or HSE crystal output clock frequency. Refer to Figure 17 and Clock control register. The PLL configuration (selection of HSI oscillator or HSE oscillator for PLL input clock, and multiplication factor) must be done before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

An interrupt request can be generated when the PLL is ready if PLL interrupt is enabled in the Clock interrupt register.

If the USB interface is used in the application, the PLL must be programmed to output 48 or 96MHz. This is needed to provide a 48MHz USBCLK.

### 9.2.4 LSE clock

The LSE crystal is a 32.768KHz low speed external crystal or ceramic resonator. It has the advantage providing a low-power but highly accurate clock source to the real-time clock or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in Backup domain control register (RCC\_BDCR). The LSERDY flag in the Backup domain control register (RCC\_BDCR) indicates if the LSE crystal oscillator is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt request can be generated if enabled in the Clock interrupt register (RCC\_CIR).

### 9.2.5 LSI clock

The LSI oscillator acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and auto-wakeup unit (AWU). The LSI clock frequency is around 40KHz. For more details, refer to the electrical characteristics section of the datasheets.

The LSI oscillator can be switched on and off using the LSION bit in the control/status register (RCC\_CSR). The LSIRDY flag in the control/status register (RCC\_CSR) indicates if the low-speed internal oscillator is stable or not. At startup, the clock is not released until this bit is set to '1' by hardware. A LSI interrupt request can be generated if enabled in the clock interrupt register (RCC\_CIR).

### 9.2.6 System clock (SYSCLK) selection

After a system reset, the HSI oscillator divided by 6 is selected as system clock. When a clock source is used directly or through the PLL as system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the system clock switch will not occur. A switch occurs until the target clock source is ready.

Status bits in the clock configuration register (RCC\_CFGR) indicate which clock(s) is (are) ready and which clock is currently used as system clock.

### 9.2.7 Clock security system (CSS)

Clock security system can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when the HSE clock is stopped.

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled. A clock failure event is sent to the break input of the advanced timer (TIM1) and a clock security interrupt CSS is generated, allowing the software to perform rescue operations. The CSS interrupt is linked to the CPU NMI (Non-Maskable Interrupt) exception vector.

Note: Once the CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and an NMI is automatically generated. The NMI will be executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the clock interrupt register (RCC\_CIR).

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock, and the PLL clock is used as system clock), a detected clock failure causes a switch of the system clock to the HSI oscillator and the disabling of the external HSE oscillator. If the HSE oscillator clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

### 9.2.8 RTC clock

The RTCCLK clock source can be either the HSE/128, LSE or LSI clocks. This is selected by programming the RTCSEL[1:0] bits in the Backup domain control register (RCC\_BDCR). This selection cannot be modified without resetting the backup domain.

The LSE clock is in the backup domain, whereas the HSE and LSI clocks are not. Consequently:

- If LSE is selected as RTC clock:
  - The RTC stops working if the V<sub>DD</sub> supply is switched off.
- If LSI is selected as Auto-Wakeup unit (AWU) clock: refer to section 9.2.5 for more details.
  - The AWU state is not guaranteed if the V<sub>DD</sub> supply is powered off.
- If the HSE clock divided by 128 is used as the RTC clock:
  - The RTC state is not guaranteed if the V<sub>DD</sub> supply is powered off or if the internal voltage regulator is powered off (removing power from the 1.5 V domain).
  - The DPB bit (disable backup domain write protection) in the Power control register must be set to ‘1’.

### 9.2.9 Watchdog clock

If the independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator temporization, the clock is provided to the IWDG.

### 9.2.10 Clock-out capability

The microcontroller allows the clock to be output onto the external MCO pin.

The registers of the corresponding GPIO port must be configured with related functions. One of 9 clock signals can be selected as the MCO clock:

- SYSCLK
- HSI
- HSI/4
- HSI/12
- HSE
- LSI
- LSE
- PLLCLK/2
- PLLCLK/24

The clock selection is controlled by the MCO[3:0] bits of the clock configuration register (RCC\_CFGR).

## 9.3 RCC register file and memory mapping description

Table 23. Overview of RCC registers

Offset	Acronym	Register Name	Reset	Section
0x00	RCC_CR	Clock control register	0x0000XX01	Section 9.3.1
0x04	RCC_CFGR	Clock configuration register	0x00000000	Section 9.3.2
0x08	RCC_CIR	Clock interrupt register	0x00000000	Section 9.3.3
0x0C	RCC_APB2RSTR	APB2 peripheral reset register	0x00000000	Section 9.3.4
0x10	RCC_APB1RSTR	APB1 peripheral reset register	0x00000000	Section 9.3.5
0x14	RCC_AHBENR	AHB peripheral clock enable register	0x00000014	Section 9.3.6

Offset	Acronym	Register Name	Reset	Section
0x18	RCC_APB2ENR	APB2 peripheral clock enable register	0x00000000	Section 9.3.7
0x1C	RCC_APB1ENR	APB1 peripheral clock enable register	0x00000000	Section 9.3.8
0x20	RCC_BDCR	Backup domain control register	0x00000000	Section 9.3.9
0x24	RCC_CSR	Control status register	0x0C000000	Section 9.3.10
0x28	RCC_AHBRSTR	AHB peripheral reset register	0x00000000	Section 9.3.11
0x2C	RCC_CFGR2	Clock configuration register 2	0x00000020	Section 9.3.12
0x30	RCC_RNG	RNG register	0x0000000X	Section 9.3.13
0x40	RCC_SYSCFG	System configuration register	0x00100C03	Section 9.3.14

### 9.3.1 Clock control register (RCC\_CR)

Address offset: 0x00

Reset value: 0x0000 XX01

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLMUL						PLLRDY	PLLON	Res.	PLLDIV			CSSON	HSEBYPHSERDY	HSEON	
rw	rw	rw	rw	rw	rw	r	rw		rw	rw	rw	rw	rw	r	rw
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSITEN HSIRDY HSION															
rw		r													

Bit	Field	Type	Reset	Description
31:26	PLLMUL	rw	0x00	PLLMUL: PLL multiplication factor
25	PLLRDY	r	0x00	PLLRDY: PLL clock ready flag Set by hardware to indicate that the PLL is locked. 0: PLL unlocked 1: PLL locked
24	PLLON	rw	0x00	PLLON: PLL enable Set and cleared by software. Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the PLL clock is used as system clock or is selected to become the system clock. 0: PLL OFF 1: PLL ON
23	Reserved			always read as 0

Bit	Field	Type	Reset	Description
22:20	PLLDIV	rw	0x00	<p>PLLDIV: PLL divider factor</p> <p>PLL configuration formula: <math>F_{CLKO} = F_{REFIN} * N/M</math>, where:  <math>F_{CLKO}</math> is PLL output frequency, <math>F_{REFIN}</math> is PLL input reference clock frequency  <math>M = PLLDIV + 1</math>  <math>N = PLLMUL + 1</math></p>
19	CSSON	rw	0x00	<p>CSSON: Clock security system enable</p> <p>Set and cleared by software to enable the clock detector.</p> <p>0: clock detector OFF</p> <p>1: clock detector ON if the external oscillator is ready</p>
18	HSEBYP	rw	0x00	<p>HSEBYP: External high-speed clock bypass</p> <p>Set by software to bypass the external crystal oscillator in the debug mode. The HSEBYP bit can be written only if the external oscillator is disabled.</p> <p>0: external oscillator not bypassed</p> <p>1: external crystal oscillator bypassed</p>
17	HSERDY	r	0x00	<p>HSERDY: External high-speed clock ready flag</p> <p>Set by hardware to indicate that the external clock is stable.</p> <p>0: external clock not ready</p> <p>1: external clock ready</p>
16	HSEON	rw	0x00	<p>HSEON: External high-speed clock enable</p> <p>Set and cleared by software.</p> <p>Cleared by hardware to stop the external clock when entering Stop or Standby mode. This bit cannot be reset if the external clock is used or selected as the system clock.</p> <p>0: HSE oscillator OFF</p> <p>1: HSE oscillator ON</p>
15: 3	Reserved			always read as 0
2	HSITEN	rw	0x00	<p>HSITEN: Set and cleared by software to enable the internal high-speed clock temperature calibration.</p> <p>0: internal HSI clock not automatically calibrated with temperature</p> <p>1: internal HSI clock automatically calibrated with temperature</p>

Bit	Field	Type	Reset	Description
1	HSIRDY	r	0x00	<p>HSIRDY: Internal high-speed clock ready flag Set by hardware to indicate that internal clock is stable. After the HSION bit is cleared, HSIRDY goes low after 3 AHB clock cycles.</p> <p>0: internal high-speed clock not ready 1: internal high-speed clock ready</p>
0	HSION	rw	0x01	<p>HSION: Internal high-speed clock enable Set and cleared by software.</p> <p>Set by hardware to force the internal oscillator ON when leaving Stop or Standby mode or in case of failure of the external clock used as system clock. This bit cannot be reset if the internal high-speed clock is used directly or indirectly as system clock or is selected to become the system clock.</p> <p>0: internal high-speed clock OFF 1: internal high-speed clock ON</p>

### 9.3.2 Clock configuration register (RCC\_CFGR)

Address offset: 0x04

Reset value: 0x0000 0000

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

1 or 2 wait states inserted only if the access occurs during clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLDN		Reserved		MCO		USBPRE		Reserved		PLLXTPRE	PLLSRC				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLK48SEL	Res.	PPRE2		PPRE1		HPRE		SWS		SW					
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bit	Field	Type	Reset	Description
31:30	PLLDN	rw	0x00	PLL configuration factor extension
29:28	Reserved			always read as 0
27:24	MCO	rw	0x00	<p>MCO: Microcontroller clock output Set and cleared by software.</p> <p>0001: MCO outputs HSI/DIV4; 0010: MCO outputs LSI; 0011: MCO outputs LSE; 0100: MCO outputs SYSTEM CLOCK; 0101: MCO outputs HSI; 0110: MCO outputs HSE; 0111: MCO outputs PLL/DIV2; 1000 : MCO outputs HSI/DIV12; 1001 : MCO outputs PLL/DIV24;</p> <p>Note:</p> <ol style="list-style-type: none"> <li>1. This clock output may have some truncated cycles at startup or during MCO clock source switching.</li> <li>2. When the system clock is selected to output to the MCO pin, make sure that this clock does not exceed 50MHz (the maximum IO frequency).</li> </ol>
23:22	USBPRE	rw	0x00	<p>USBPRE: USB prescaler Set or cleared by software to generate 48MHz USB clock. This bit must be valid before enabling the USB clock in the RCC_APB1ENR register.</p> <p>00: PLL clock is used as system clock directly 01: PLL clock divided by 2 is used as the USB clock 10: PLL clock divided by 3 is used as the USB clock 11: PLL clock divided by 4 is used as the USB clock</p>
21:18	Reserved			always read as 0
17	PLLXTPRE	rw	0x00	<p>PLLXTPRE: HSE divider for PLL entry Set or cleared by software to divide HSE before PLL entry. This bit can be written only when PLL is disabled.</p> <p>0: HSE clock not divided 1: HSE clock divided by 2</p>
16	PLLSRC	rw	0x00	<p>PLLSRC: PLL entry clock source Set and cleared by software to select PLL clock source. This bit can be written only when PLL is disabled.</p> <p>0: HSI clock divided by 4 is used as the PLL input clock 1: HSE clock is used as the PLL input clock</p>
15	CLK48MSEL	rw	0x00	<p>CLK48MSEL: USB 48M clock source selection</p> <p>0: USB 48M clock from PLL 1: USB 48M clock from HSI48M</p>
14	Reserved			always read as 0

Bit	Field	Type	Reset	Description
13: 11	PPRE2	rw	0x00	<p>PPRE2: APB2 prescale coefficient</p> <p>Set and cleared by software to control the prescale coefficient of the APB2 high-speed clock (PCLK2).</p> <p>0xx: HCLK not divided</p> <p>100: HCLK divided by 2</p> <p>101: HCLK divided by 4</p> <p>110: HCLK divided by 8</p> <p>111: HCLK divided by 16</p>
10: 8	PPRE1	rw	0x00	<p>PPRE1: APB1 prescale coefficient</p> <p>Set and cleared by software to control the prescale coefficient of the APB1 low-speed clock (PCLK1).</p> <p>0xx: HCLK not divided</p> <p>100: HCLK divided by 2</p> <p>101: HCLK divided by 4</p> <p>110: HCLK divided by 8</p> <p>111: HCLK divided by 16</p>
7: 4	HPRE	rw	0x00	<p>HPRE: AHB prescale coefficient</p> <p>Set and cleared by software to control the prescale coefficient of the AHB clock.</p> <p>0xxx: SYSCLK not divided</p> <p>1000: SYSCLK divided by 2</p> <p>1001: SYSCLK divided by 4</p> <p>1010: SYSCLK divided by 8</p> <p>1011: SYSCLK divided by 16</p> <p>1100: SYSCLK divided by 64</p> <p>1101: SYSCLK divided by 128</p> <p>1110: SYSCLK divided by 256</p> <p>1111: SYSCLK divided by 512</p> <p><b>Note:</b></p> <ol style="list-style-type: none"> <li>1. The prefetch buffer must be kept on when using a prescale coefficient greater than 1 on the AHB clock. Refer to Reading the Flash memory section for more details.</li> </ol>
3: 2	SWS	r	0x00	<p>SWS: System clock switch status</p> <p>00: HSI divided by 6 used as system clock</p> <p>01: HSE oscillator used as system clock</p> <p>10: PLL used as system clock</p> <p>11: LSI used as system clock</p>

Bit	Field	Type	Reset	Description
1: 0	SW	rw	0x00	<p>SW: System clock switch Set and cleared by hardware to indicate which clock source is used as system clock.</p> <p>Set by hardware to force HSI divided by 6 to be the system clock when leaving Stop and Standby mode or in case of failure of the HSE used directly or indirectly as system clock.</p> <p>00: HSI divided by 6 used as system clock; 01: HSE used as system clock; 10: PLL used as system clock; 11: LSI used as system clock.</p>

### 9.3.3 Clock interrupt register (RCC\_CIR)

Address offset: 0x08

Reset value: 0x0000 0000

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

31	30	28	27	26	25	24	23	22	21	20	19	18	17	16	
									CSSC	Reserved	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
									rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	
15	14	1:	12	11	10	9	8	7	6	5	4	3	2	1	0
		Reserved	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Reserved	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF	
			rw	rw	rw	rw	rw	r		r	r	r	r	r	

Bit	Field	Type	Reset	Description
31:24	Reserved			always read as 0.
23	CSSC	rc_w1	0x00	<p>CSSC: clock security system interrupt clear Set to '1' by software to clear the CSSF security system interrupt flag.</p> <p>0: No effect 1: Clear CSSF security system interrupt flag</p>
22:21	Reserved			always read as 0
20	PLLRDYC	rc_w1	0x00	<p>PLLRDYC: PLL ready interrupt clear Set by software to clear the PLL ready interrupt flag PLLRDYF.</p> <p>0: No effect 1: Clear PLL ready interrupt flag PLLRDYF</p>
19	HSERDYC	rc_w1	0x00	<p>HSERDYC: HSE ready interrupt clear Set by software to clear the HSE ready interrupt flag HSERDYF.</p> <p>0: No effect 1: Clear HSE ready interrupt flag HSERDYF</p>

Bit	Field	Type	Reset	Description
18	HSIRDYC	rc_w1	0x00	HSIRDYC: HSI ready interrupt clear Set by software to clear the HSI ready interrupt flag HSIRDYF. 0: No effect 1: Clear HSI ready interrupt flag HSIRDYF
17	LSERDYC	rc_w1	0x00	LSERDYC: LSE ready interrupt clear Set by software to clear the LSE ready interrupt flag LSERDYF. 0: No effect 1: Clear LSE ready interrupt flag LSERDYC
16	LSIRDYC	rc_w1	0x00	LSIRDYC: LSI ready interrupt clear Set to '1' by software to clear the LSI ready interrupt flag LSIRDYF. 0: No action 1: Clear LSI ready interrupt flag LSIRDYF
15: 13	Reserved			always read as 0
12	PLL RDYIE	rw	0x00	PLL RDYIE: PLL ready interrupt enable Set and cleared by software to enable/disable the PLL ready interrupt. 0: PLL ready interrupt disabled 1: PLL ready interrupt enabled
11	HSE RDYIE	rw	0x00	HSE RDYIE: HSE ready interrupt enable Set and cleared by software to enable/disable interrupt caused by the external oscillator stabilization. 0: HSE ready interrupt disabled 1: HSE ready interrupt enabled
10	HSI RDYIE	rw	0x00	HSI RDYIE: HSI ready interrupt enable Set and cleared by software to enable/disable interrupt caused by the internal oscillator stabilization. 0: HSI ready interrupt disabled 1: HSI ready interrupt enabled
9	LSE RDYIE	rw	0x00	LSE RDYIE: LSE ready interrupt enable Set and cleared by software to enable/disable interrupt caused by the external 32KHz oscillator stabilization. 0: LSE ready interrupt disabled 1: LSE ready interrupt enabled
8	LSI RDYIE	rw	0x00	LSI RDYIE: LSI ready interrupt enable Set to '1' or cleared as '0' by software to enable/disable interrupt caused by the internal 40KHz oscillator stabilization. 0: LSI ready interrupt disabled 1: LSI ready interrupt enabled

Bit	Field	Type	Reset	Description
7	CSSF	r	0x00	CSSF: Clock security system interrupt flag Set by hardware when a failure is detected in the external oscillator clock. Cleared by software setting the CSSC bit to '1'. 0: No clock security interrupt caused by HSE clock failure 1: Clock security interrupt caused by HSE clock failure
6: 5	Reserved			always read as 0
4	PLLRDYF	r	0x00	PLLRDYF: PLL ready interrupt flag Set by hardware when the PLL is ready. cleared by software setting the PLLRDYC bit to '1'. 0: No clock ready interrupt caused by PLL lock 1: Clock ready interrupt caused by PLL lock
3	HSERDYF	r	0x00	HSERDYF: HSE ready interrupt flag Set by hardware when the external low speed clock becomes stable. Cleared by software setting the HSERDYC bit to '1'. 0: No clock ready interrupt caused by the external oscillator 1: Clock ready interrupt caused by the external oscillator
2	HSIRDYF	r	0x00	HSIRDYF: HSI ready interrupt flag Set by hardware when the internal high speed clock becomes stable. Cleared by software setting the HSIRDYC bit to '1'. 0: No clock ready interrupt caused by the internal HSI oscillator 1: Clock ready interrupt caused by the internal HSI oscillator
1	LSERDYF	r	0x00	LSERDYF: LSE ready interrupt flag Set by hardware when the external low speed clock becomes stable. Cleared by software setting the LSERDYC bit to '1'. 0: No clock ready interrupt caused by the internal 32KHz oscillator; 1: Clock ready interrupt caused by the internal 32KHz oscillator.
0	LSIRDYF	r	0x00	LSIRDYF: LSI ready interrupt flag Set by hardware when the internal low speed clock becomes stable. Cleared by software setting the LSIRDYC bit to '1'. 0: No clock ready interrupt caused by the internal 40KHz oscillator; 1: Clock ready interrupt caused by the internal 40KHz oscillator.

### 9.3.4 APB2 peripheral reset register (RCC\_APB2RSTR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20		18	17	16
Reserved					DBGMCU	Reserved	TIM17	TIM16	TIM14						
						rw						rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP	UART1	Res.	SPI1	TIM1	Res.	ADC1	Reserved					SYSCFG			
rw	rw		rw	rw		rw							rw		

Bit	Field	Type	Reset	Description
31: 23	Reserved			always read as 0
22	DBGMCU	rw	0x00	DBGMCU: DBGMCU reset
21: 19	Reserved			always read as 0
18	TIM17	rw	0x00	TIM17: TIM17 timer reset Set and cleared by software. 0: No effect 1: reset TIM17 timer
17	TIM16	rw	0x00	TIM16: TIM16 timer reset Set and cleared by software. 0: No effect 1: reset TIM16 timer
16	TIM14	rw	0x00	TIM14: TIM14 timer reset Set and cleared by software. 0: No effect 1: reset TIM14 timer
15	COMP	rw	0x00	COMP: Comparator reset Set and cleared by software. 0: No effect 1: reset comparator interface
14	UART1	rw	0x00	UART1: UART1 reset Set and cleared by software. 0: No effect 1: reset UART1
13	Reserved			always read as 0
12	SPI1	rw	0x00	SPI1: SPI1 reset Set and cleared by software. 0: No effect 1: reset SPI1

Bit	Field	Type	Reset	Description
11	TIM1	rw	0x00	TIM1: TIM1 timer reset Set and cleared by software. 0: No effect 1: reset TIM1 timer
10	Reserved			always read as 0
9	ADC1	rw	0x00	ADC1: ADC1 interface reset Set and cleared by software. 0: No effect 1: reset ADC1 interface
8: 1	Reserved			always read as 0
0	SYSCFG	rw	0x00	SYSCFG: system configuration register reset Set and cleared by software. 0: No effect 1: reset SYSCFG

### 9.3.5 APB1 peripheral reset register (RCC\_APB1RSTR)

Address offset: 0x10

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PWR	CRS	CSM	CAN	Res.	USB	Res.	I2C1	Reserved	UART2	Res.				
	rw	rw	rw	rw		rw	rw	rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2	Reserved	WWDG											TIM3	TIM2
	rw		rw											rw	rw

Bit	Field	Type	Reset	Description
31:29	Reserved			always read as 0.
28	PWR	rw	0x00	PWR: power interface reset Set and cleared by software. 0: No effect 1: Reset power interface
27	CRS	rw	0x00	CRS: CRS interface reset Set and cleared by software. 0: No effect 1: Reset CRS
26	CSM	rw	0x00	CSM: CSM reset Set and cleared by software. 0: No effect 1: Reset CSM

Bit	Field	Type	Reset	Description
25	CAN	rw	0x00	CAN: CAN reset Set and cleared by software. 0: No effect 1: Reset CAN
24	Reserved			always read as 0
23	USB	rw	0x00	USB: USB reset Set and cleared by software. 0: No effect 1: Reset USB
22	Reserved			always read as 0
21	I2C1	rw	0x00	I2C1: I2C1 reset Set and cleared by software. 0: No effect 1: Reset I2C1
20:18	Reserved			always read as 0
17	UART2	rw	0x00	UART2: UART2 reset Set and cleared by software. 0: No effect 1: Reset UART2
16:15	Reserved			always read as 0
14	SPI2	rw	0x00	SPI2: SPI2 reset Set and cleared by software. 0: No effect 1: Reset SPI2
13:12	Reserved			always read as 0
11	WWDG	rw	0x00	WWDG: Window watchdog reset Set and cleared by software. 0: No effect 1: Reset window watchdog
10:2	Reserved			always read as 0
1	TIM3	rw	0x00	TIM3: Timer 3 reset Set and cleared by software. 0: No effect 1: Reset TIM3 timer
0	TIM2	rw	0x00	TIM2: Timer 2 reset Set and cleared by software. 0: No effect 1: Reset TIM2 timer

### 9.3.6AHB peripheral clock enable register (RCC\_AHBENR)

Address offset: 0x14

Reset value: 0x0000 0014

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				HWDIV	Reserved				GPIOD	GPIOC	GPIOB	GPIOA	Res.		
					rw				rw	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CRC	Res.	FLASH	Res.	SRAM	Res.	DMA	rw
									rw	rw	rw	rw			

Bit	Field	Type	Reset	Description
31:27	Reserved			always read as 0.
26	HWDIV	rw	0x00	HWDIV: HWDIV clock enable Set and cleared by software. 0: HWDIV clock disabled 1: HWDIV clock enabled
25:21	Reserved			always read as 0
20	GPIOD	rw	0x00	GPIOD: GPIOD clock enable 0: GPIOD clock disabled 1: GPIOD clock enabled
19	GPIOC	rw	0x00	GPIOC: GPIOC clock enable 0: GPIOC clock disabled 1: GPIOC clock enabled
18	GPIOB	rw	0x00	GPIOB: GPIOB clock enable 0: GPIOB clock disabled 1: GPIOB clock enabled
17	GPIOA	rw	0x00	GPIOA: GPIOA clock enable 0: GPIOA clock disabled 1: GPIOA clock enabled
16:7	Reserved			always read as 0
6	CRC	rw	0x00	CRC: CRC clock enable Set and cleared by software. 0: CRC clock disabled 1: CRC clock enabled
5	Reserved			always read as 0
4	FLASH	rw	0x01	FLASH: FLASH clock enable 0: FLASH clock disabled 1: FLASH clock enabled

Bit	Field	Type	Reset	Description
3	Reserved			always read as 0
2	SRAM	rw	0x01	SRAM: SRAM interface clock enable 0: SRAM clock disabled 1: SRAM clock enabled
1	Reserved			always read as 0
0	DMA	rw	0x00	DMA: DMA clock enable Set and cleared by software. 0: DMA clock disabled 1: DMA clock enabled

### 9.3.7 APB2 peripheral clock enable register (RCC\_APB2ENR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Note: if the peripheral clock is OFF, the software cannot read the value of the peripheral register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					DBGMCU	Reserved		TIM17	TIM16	TIM14					
								rw			rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP	UART1	Res.	SPI1	TIM1	Res.	ADC1	Reserved					SYSCFG			
rw	rw		rw	rw		rw							rw		

Bit	Field	Type	Reset	Description
31: 23	Reserved			always read as 0
22	DBGMCU	rw	0x00	DBGMCU enable
21: 19	Reserved			always read as 0
18	TIM17	rw	0x00	TIM17: TIM17 timer enable Set and cleared by software. 0: TIM17 timer clock disabled 1: TIM17 timer clock enabled
17	TIM16	rw	0x00	TIM16: TIM16 timer enable Set and cleared by software. 0: TIM16 timer clock disabled 1: TIM16 timer clock enabled
16	TIM14	rw	0x00	TIM14: TIM14 timer enable Set and cleared by software. 0: TIM14 timer clock disabled 1: TIM14 timer clock enabled

Bit	Field	Type	Reset	Description
15	COM	rw	0x00	COMP: Comparator enable Set and cleared by software. 0: comparator interface clock disabled 1: comparator interface clock enabled
14	UART1	rw	0x00	UART1: UART1 clock enable Set and cleared by software. 0: UART1 clock disabled 1: UART1 clock enabled

Bit	Field	Type	Reset	Description
13	Reserved			always read as 0
12	SPI1	rw	0x00	SPI1: SPI1 clock enable Set and cleared by software. 0: SPI1 clock disabled 1: SPI1 clock enabled
11	TIM1	rw	0x00	TIM1: TIM1 Timer clock enable Set and cleared by software. 0: TIM1 timer clock disabled 1: TIM1 timer clock enabled
10	Reserved			always read as 0
9	ADC1	rw	0x00	ADC1: ADC1 interface clock enable Set and cleared by software. 0: ADC1 interface clock disabled 1: ADC1 interface clock enabled
8: 1	Reserved			always read as 0
0	SYSCFG	rw	0x00	SYSCFG: System configuration register enable Set and cleared by software. 0: System configuration register clock disabled 1: System configuration register clock enabled

### 9.3.8 APB1 peripheral clock enable register (RCC\_APB1ENR)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Note: if the peripheral clock is OFF, the software cannot read the value of the peripheral register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PWR	CRS	CSM	CAN	Res.	USB	Res.	I2C1	Reserved	UART2	Res.				
	rw	rw	rw	rw	rw	rw	rw	rw		rw				rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2	Reserved	WWDG											TIM3	TIM2
	rw		rw											rw	rw

Bit	Field	Type	Reset	Description
31:29	Reserved			always read as 0.
28	PWR	rw	0x00	PWR: Power interface clock enable Set and cleared by software. 0: power interface clock disabled 1: power interface clock enabled
27	CRS	rw	0x00	CRS: CRS clock enable Set and cleared by software. 0: CRS clock disabled 1: CRS clock enabled
26	CSM	rw	0x00	CSM: CSM clock enable Set and cleared by software. 0: CSM clock disabled 1: CSM clock enabled
25	CAN	rw	0x00	CAN: CAN clock enable Set and cleared by software. 0: CAN clock disabled 1: CAN clock enabled
24	Reserved			always read as 0
23	USB	rw	0x00	USB: USB clock enable Set and cleared by software. 0: USB clock disabled 1: USB clock enabled
22	Reserved			always read as 0
21	I2C1	rw	0x00	I2C1: I2C1 clock enable Set and cleared by software. 0: I2C1 clock disabled 1: I2C1 clock enabled
20:18	Reserved			always read as 0
17	UART2	rw	0x00	UART2: UART2 clock enable Set and cleared by software. 0: UART2 clock disabled 1: UART2 clock enabled
16:15	Reserved			always read as 0
14	SPI2	rw	0x00	SPI2: SPI2 clock enable Set and cleared by software. 0: SPI2 clock disabled 1: SPI2 clock enabled

Bit	Field	Type	Reset	Description
13:12	Reserved			always read as 0
11	WWDG	rw	0x00	WWDG: Window watchdog clock enable Set and cleared by software. 0: Window watchdog clock disabled 1: Window watchdog clock enabled
10:2	Reserved			always read as 0
1	TIM3	rw	0x00	TIM3: Timer3 clock enable Set and cleared by software. 0: Timer 3 clock disabled 1: Timer 3 clock enabled
0	TIM2	rw	0x00	TIM2: Timer 2 clock enable Set and cleared by software. 0: Timer 2 clock disabled 1: Timer 1 clock enabled

### 9.3.9 Backup domain control register (RCC\_BDCR)

Address offset: 0x20

Reset value: 0x0000 0000, only reset by Backup domain

Reset.

Access: 0 ~3 wait states, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

Note: The LSEON, LSEBYP, RTCSEL and RTCEN bits of the Backup domain control register (RCC\_BDCR) are in the Backup domain.

As a result, after Reset, these bits are write-protected and the DBP bit in the Power control register (PWR\_CR) has to be set before these can be modified. These bits are only reset after a Backup domain Reset. Any internal or external Reset will not have any effect on these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved															BDRST	
																rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RTCEN	Reserved			RTCSEL		Reserved					LSEBYP	LSERDY	LSEON			
	rw				rw						rw	r	rw			

Bit	Field	Type	Reset	Description
31:17	Reserved			always read as 0.
16	BDRST	rw	0x00	BDRST: Backup domain software reset Set and cleared by software. 0: Reset not activated 1: Resets the entire Backup domain
15	RTCEN	rw	0x00	RTCEN: RTC clock enable Set and cleared by software. 0: RTC clock disabled 1: RTC clock enabled
14:10	Reserved			always read as 0

Bit	Field	Type	Reset	Description
9:8	RTCSEL	rw	0x00	<p>RTCSEL: RTC clock source selection</p> <p>Set by software to select the clock source for the RTC.</p> <p>Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. The BDRST bit can be used to reset them.</p> <p>00: No clock</p> <p>01: LSE oscillator clock used as RTC clock</p> <p>10: LSI oscillator clock used as RTC clock</p> <p>11: HSE oscillator clock divided by 128 used as RTC clock</p>
7:3	Reserved			always read as 0
2	LSEBYP	rw	0x00	<p>LSEBYP: External low-speed oscillator bypass</p> <p>Set by software to bypass the LSE in the debug mode. This bit can be written only when the external 32KHz oscillator is disabled.</p> <p>0: LSE clock not bypassed</p> <p>1: LSE clock bypassed</p>
1	LSERDY	r	0x00	<p>LSERDY: External low-speed oscillator ready</p> <p>Set and cleared by hardware to indicate when the external 32KHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.</p> <p>0: External 32KHz oscillator not ready</p> <p>1: External 32KHz oscillator ready</p>
0	LSEON	rw	0x00	<p>LSEON: External low-speed oscillator enable</p> <p>Set and cleared by software.</p> <p>0: External 32KHz oscillator OFF</p> <p>1: External 32KHz oscillator ON</p>

### 9.3.10 Control/status register (RCC\_CSR)

Address offset: 0x24

Reset value: 0x0C00 0000

Access: 0 ~ 3 wait cycles, word, half-word and byte access

Wait states are inserted in the case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	Res.	RMVF	LOCK UPF	PVD RSTF	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved				LOCK UPEN	PVD RSTEN	Reserved	LSIRDY	LSION
				rw	rw		r	rw
Bit	Field	Type	Reset	Description				
31	Reserved			always read as 0				
30	WWDGRSTF	r	0x0x	<p>WDGRSTF: Window watchdog reset flag</p> <p>Set by hardware when a window watchdog reset occurs.</p> <p>Cleared only by resetting the power or writing to the RMVF bit via software.</p> <p>0: No window watchdog reset occurred</p> <p>1: Window watchdog reset occurred</p>				
29	IWDGRSTF	r	0x0x	<p>IWDGRSTF: Independent watchdog reset flag</p> <p>Set by hardware when an independent watchdog reset from V<sub>DD</sub> domain occurs.</p> <p>Cleared only by resetting the power or writing to the RMVF bit via software.</p> <p>0: No independent watchdog reset occurred</p> <p>1: Independent watchdog reset occurred</p>				
28	SFTRSTF	r	0x0x	<p>SFTRSTF: Software reset flag</p> <p>Set by hardware when a software reset occurs.</p> <p>Cleared only by resetting the power or writing to the RMVF bit via software.</p> <p>0: No software reset occurred</p> <p>1: Software reset occurred</p>				
27	PORRSTF	r	0x01	<p>PORRSTF: POR/PDR reset flag</p> <p>Set by hardware when a POR/PDR reset occurs.</p> <p>Cleared only by resetting the power or writing to the RMVF bit via software.</p> <p>0: No POR/PDR reset occurred</p> <p>1: POR/PDR reset occurred</p>				
26	PINRSTF	r	0x01	<p>PINRSTF: nRST PIN reset flag</p> <p>Set by hardware when a reset from the nRST pin occurs.</p> <p>Cleared only by resetting the power or writing to the RMVF bit via software.</p> <p>0: No reset from nRST pin occurred</p> <p>1: Reset from nRST pin occurred</p>				
25	Reserved			always read as 0				
24	RMVF	w	0x00	<p>RMVF: Remove reset flag</p> <p>Set by software to clear the reset flags.</p> <p>0: No effect</p> <p>1: Clear the reset flags</p>				

23	LOCKUPF	r	0x00	<p>LOCKUPF: CPU lockup reset flag Set by hardware when a reset from the CPU lockup occurs. Cleared by resetting the power or writing to the RMVF bit via software. 0: No reset from the CPU deadlock occurred 1: Reset from the CPU deadlock occurred</p>
----	---------	---	------	---

Bit	Field	Type	Reset	Description
22	PVDRSTF	r	0x00	<p>PVDRSTF: PVD reset flag Set to '1' by hardware when a PVD reset occurs. Cleared only by resetting the power or writing to the RMVF bit via software. 0: No PVD reset occurred 1: PVD reset occurred</p>
21:8	Reserved			always read as 0
7	LOCKUPEN	rw	0x00	<p>LOCKUPEN: CPU lockup reset enable 0: CPU lockup reset disabled 1: CPU lockup reset enabled</p>
6	PVDRSTEN	rw	0x00	<p>PVDRSTEN: PVD reset enable 0: PVD reset disabled 1: PVD reset enabled</p>
5:2	Reserved			always read as 0
1	LSIRDY	r	0x00	<p>LSIRDY: Internal low-speed oscillator ready Set and cleared by hardware to indicate when the internal 40KHz oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 AHB clock cycles. 0: Internal 40KHz oscillator not ready 1: Internal 40KHz oscillator ready</p>
0	LSION	rw	0x00	<p>LSION: Internal low-speed oscillator enable) Set and cleared by software, or cleared by resetting the power. 0: Internal 40KHz oscillator OFF 1: Internal 40KHz oscillator ON</p>

### 9.3.11 AHB peripheral reset register (RCC\_AHBRSTR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: 0 ~ 3 wait states, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			HWDIV	Reserved			GPIOD	GPIOC	GPIOB	GPIOA	Res.				
				rw					rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
Bit	Field	Type	Reset	Description											
31: 27	Reserved			always read as 0											
26	HWDIV	rw	0x00	HWDIV: HWDIV reset Set and cleared by software. 1: No effect 0: Reset HWDIV											
25: 21	Reserved			always read as 0											
20	GPIOD	rw	0x00	GPIOD: GPIOD reset Set and cleared by software. 0: No effect 1: Reset GPIOD											
19	GPIOC	rw	0x00	GPIOC: GPIOC reset Set and cleared by software. 0: No effect 1: Reset GPIOC											
18	GPIOB	rw	0x00	GPIOB: GPIOB reset Set and cleared by software. 0: No effect 1: Reset GPIOB											
17	GPIOA	rw	0x00	GPIOA: GPIOA reset Set and cleared by software. 0: No effect 1: Reset GPIOA											
16: 0	Reserved			always read as 0											

### 9.3.12 Clock configuration register 2(RCC\_CFGR2)

Address offset: 0x2C

Reset value: 0x0000 0020

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

1 or 2 wait states inserted only if the access occurs during clock source switch.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TIMADV_PRE	TIMADV_CKSEL		
												rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31: 4	Reserved			
3: 1	TIMADV_PRE	rw	0x00	<p>TIMADV_PRE: timadv_clk prescale The prescale coefficient of the system clock (SYSCLK) is controlled by software.</p> <p>0xx: timadv_clk not divided 100: timadv_clk divided by 2 101: timadv_clk divided by 4 110: timadv_clk divided by 8 111: timadv_clk divided by 16</p>
0	TIMADV_CKSEL	rw	0x00	<p>TIMADV_CKSEL: TIMADV clock selection 0: timbas_clk selected as TIMADV clock(PCLK2 divided by 2) 1: timadv_clk selected as TIMADV clock</p>

### 9.3.13 RNG register (RCC\_RNG)

Address offset: 0x30

Reset value: 0x0000 000X

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

1 or 2 wait states inserted only if the access occurs during clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
RNG_DATA																
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved														RNG_DONE	RNG_LDSD	RNG_EN
														rw	rw	

Bit	Field	Type	Reset	Description
31: 16	RNG_DATA	r	0x0000	RNG_DATA: Random number generated
15: 3	Reserved			
2	RNG_DONE	r	0xXX	<p>RNG_DONE: Random number generation is done 0: Random number generation is not done 1: Random number generation is done</p>

1	RNG_LDSD	rw	0x00	RNG_LDSD: Load seed Seed is generated after the hardware is powered. After this bit is set by software, the loaded seed will generate a random number. 0: Not load seed 1: Load seed
---	----------	----	------	---

Bit	Field	Type	Reset	Description
0	RNG_EN	rw	0x00	RNG_EN: RNG module enable 0: RNG module disabled 1: RNG module enabled

### 9.3.14 System configuration register (RCC\_SYSCFG)

Address offset: 0x40

Reset value: 0x0010 0C03

Access: 0 ~ 3 wait states, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SECTOR_1K_CFG	PROG_CHECK_EN
														rw	rw

Bit	Field	Type	Reset	Description
31: 2	Reserved			Reserved.
1	SECTOR_1K_CFG	rw	0x01	SECTOR_1K_CFG: size for Flash page erase. 1: 1K bytes 0: 512 bytes
0	PROG_CHECK_EN	rw	0x01	PROG_CHECK_EN: check whether the data in the Flash is FF while writing the Flash.(hardware is fixed to 1) 1: Check 0: Not to check

# 10

# General-Purpose I/Os (GPIOs)

## General-Purpose I/Os (GPIOs)

### 10.1 GPIO functional description

Each of the general-purpose I/O ports has two 32-bit configuration registers (GPIOx\_CRL, GPIOx\_CRH), two 32-bit data registers (GPIOx\_IDR, GPIOx\_ODR), a 32-bit set/reset register (GPIOx\_BSRR), a 16-bit reset register (GPIOx\_BRR), a 32-bit locking register (GPIOx\_LCKR), a 32-bit output open drain control register (GPIOx\_DCR) and two alternate function selection registers (GPIOx\_AFRH, GPIOx\_AFRL).

Each port bit of the general-purpose IO (GPIO) ports, can be individually configured by software in several modes.

- Input floating
- Input pull-up
- Input pull-down
- Analog input
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words (half-word or byte accesses are not allowed).

The purpose of the GPIOx\_BSRR and GPIOx\_BRR registers is to allow atomic read/modify accesses to any of the GPIO registers. This way, there is no risk that an IRQ occurs between the read and the modify access.

The figure below shows the basic structure of an I/O port bit.

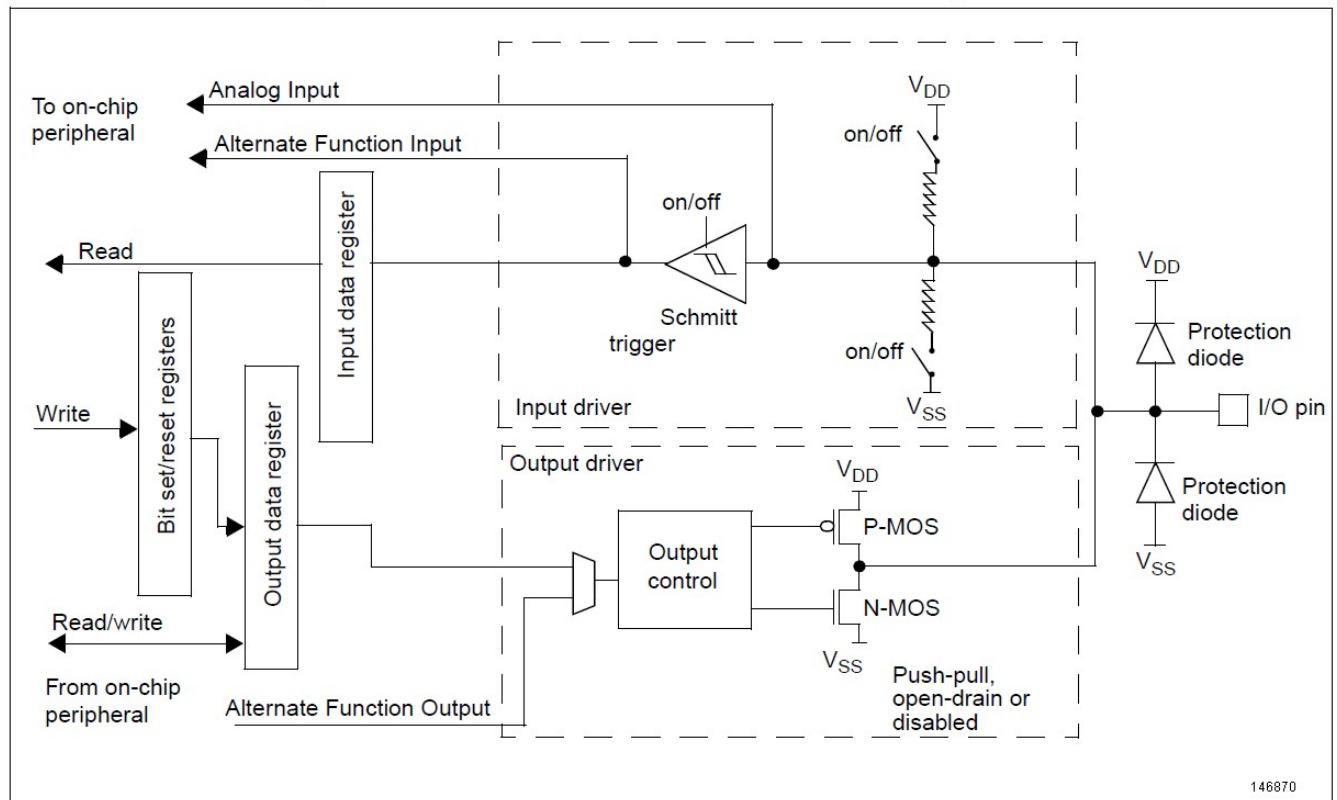


Figure 19. Basic structure of an I/O port bit

Table 24. Port bit configuration table

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR Register
General purpose output	Push-pull	0	0	01		0 or 1
	Open-drain		1			0 or 1
Alternate function output	Push-pull	1	0		not used	not used
	Open-drain		1			not used
Input	Analog input	0	0	00	not used	not used
	Input floating		1			not used
	Input pull-down	1			0	0
	Input pull-up		0			1

Table 25. Output MODE bits

MODE[1:0]	Meaning
00	Input
01	Output

### 10.1.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and the I/O ports are configured in Input Floating mode ( $CNFx[1:0] = 01$ ,  $MODEx[1:0] = 00$ ).

The SWD pins are in input pull-up (PU)/pull-down (PD) after reset:

- PA14: SWCLK in PD
- PA13: SWDIO in PU

All GPIO ports except debug ports are set in the floating state after reset.

When configured as output, the value written to the output data register ( $GPIOx\_ODR$ ) is output on the I/O pin. It is possible to use the output driver in Push-Pull mode or Open-Drain mode (only the N-MOS is activated when outputting 0).

The Input Data register ( $GPIOx\_IDR$ ) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have an internal weak pull-up and weak pull-down that can be activated or not when configured as input.

### 10.1.2 Atomic bit set or reset

There is no need for the software to disable interrupts when programming the  $GPIOx\_ODR$  at bit level:

It is possible to modify only one or several bits in a single atomic AHB write access.

This is achieved by programming to '1' the Bit Set/Reset register ( $GPIOx_BSRR$ , or for reset only  $GPIOx_BRR$ ) to select the bits to modify.

The unselected bits will not be modified.

### 10.1.3 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode. For more information on external interrupts, refer to Section 10.3: External interrupt/event controller (EXTI).

### 10.1.4 Alternate functions

It is necessary to program the Port Bit Configuration register before using a default alternate function.

- For alternate function inputs, the port must be configured in Input mode (floating, pull-up or pull-down) and the input pin must be driven externally.

Note: It is also possible to emulate the AF input pin by software by programming the GPIO controller. In this case, the port should be configured in Alternate Function Output mode. And obviously, the corresponding port should not be driven externally as it will be driven by the software using the GPIO controller.

- For alternate function outputs, the port must be configured in Alternate Function Output mode (Push-Pull or Open-Drain).
- For bidirectional Alternate Functions, the port bit must be configured in Alternate Function Output mode (Push-Pull or Open-Drain). In this case the input driver is configured in input floating mode.

If a port bit is configured as Alternate Function Output, this disconnects the output register and connects the pin to the output signal of an on-chip peripheral. If software configures a GPIO pin as Alternate Function Output, but peripheral is not activated, its output is not specified.

### 10.1.5 Software remapping of I/O alternate functions

To optimize the number of peripheral I/O functions for different device packages, it is possible to remap some alternate functions to some other pins. This is achieved by software, by programming the corresponding registers (refer to AFR register description).

In that case, the alternate functions are no longer mapped to their original assignations.

### 10.1.6 GPIO locking mechanism

The locking mechanism allows the IO configuration to be frozen. When the LOCK sequence has been applied on a port bit, it is no longer possible to modify the value of the port bit until the next reset.

### 10.1.7 Input configuration

When the I/O port is programmed as Input:

- The Output Buffer is disabled
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are activated or not depending on input configuration (pull-up, pull-down or floating)
- The data present on the I/O pin is sampled into the Input Data register every AHB clock cycle
- A read access to the Input Data register obtains the I/O State

The figure below shows the input configuration of the I/O port bit:

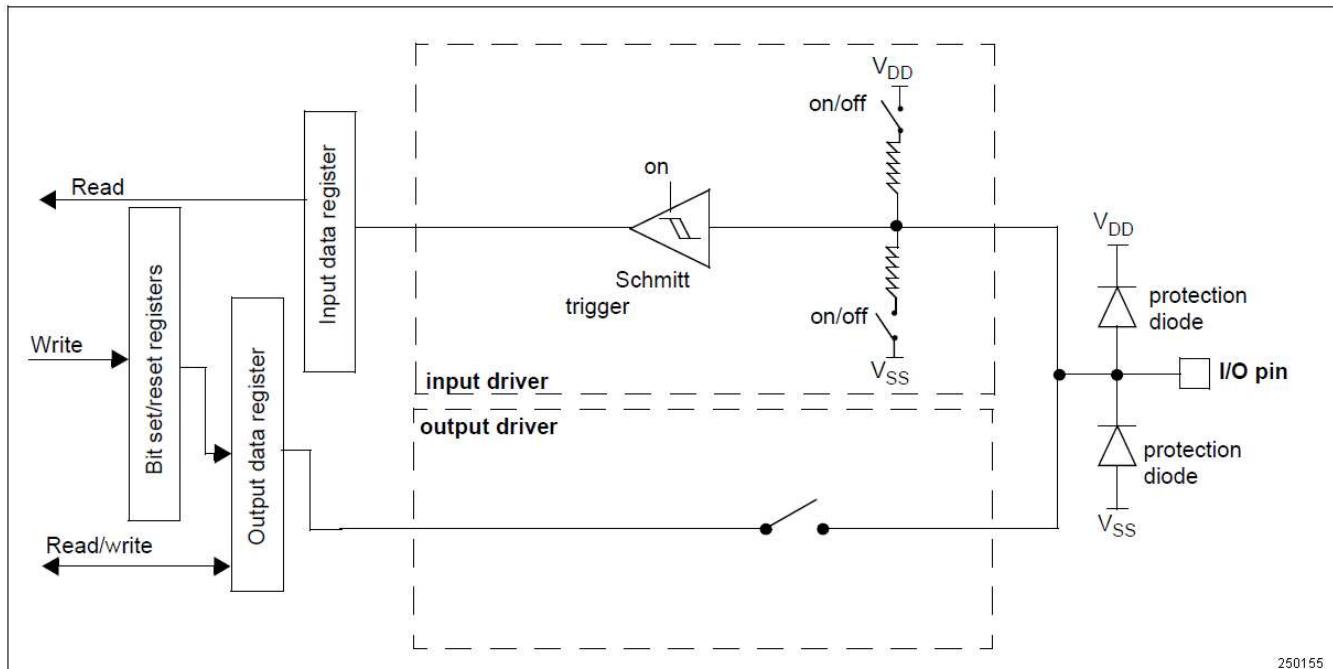


Figure 20. Input floating/pull up/pull down configurations

### 10.1.8 Output configuration

When the I/O port is configured as output:

- The Output Buffer is enabled
  - Open Drain Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-Pull Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register activates the P-MOS
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are disabled.
- The data present on the I/O pin is sampled into the Input Data register every AHB clock cycle
- A read access to the Input Data register gets the I/O state in open drain mode
- A read access to the Output Data register gets the last written value in push-pull mode

The figure below shows the output configuration of the I/O port bit:

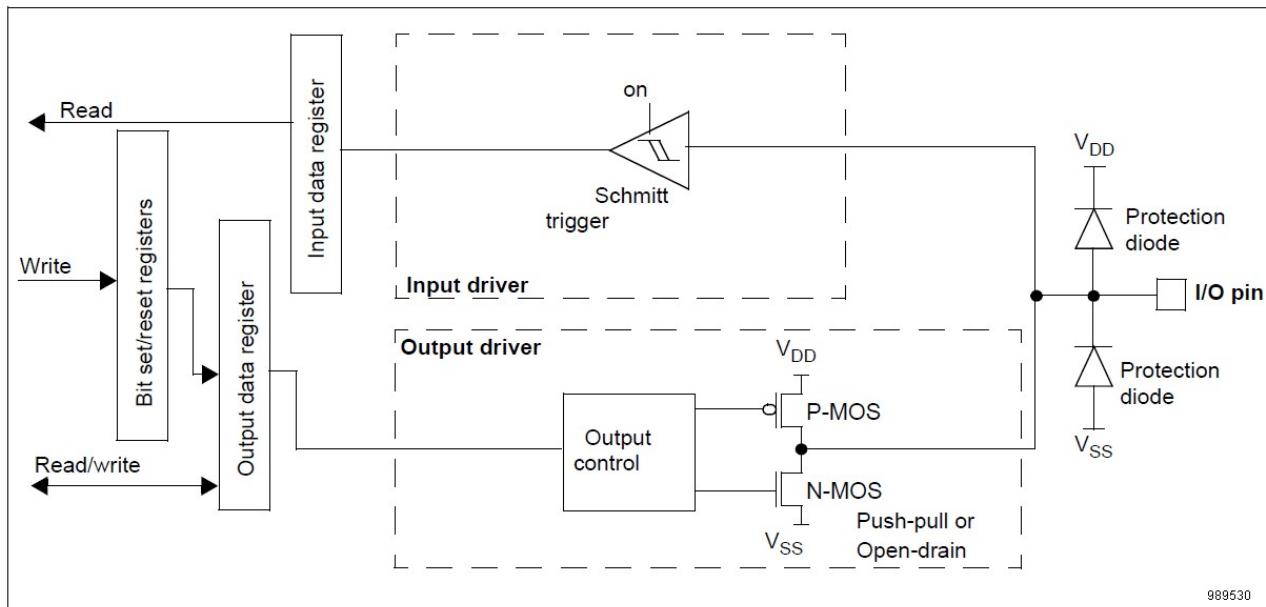


Figure 21. Output configuration

### 10.1.9 Alternate function configuration

When the I/O port is configured as alternate function:

- The Output Buffer is turned on in Open Drain or Push-Pull configuration
- The Output Buffer is driven by the signal coming from the peripheral (alternate function output)
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are disabled
- The data present on the I/O pin is sampled into the Input Data register every AHB clock cycle
- A read access to the Input Data register gets the I/O state in open drain mode
- A read access to the Output Data register gets the last written value in push-pull mode

The figure below shows the alternate function configuration of the I/O port bit. Also, refer to AFIO registers for further information.

A set of Alternate Function I/O registers allows the user to remap some alternate functions to different pins.

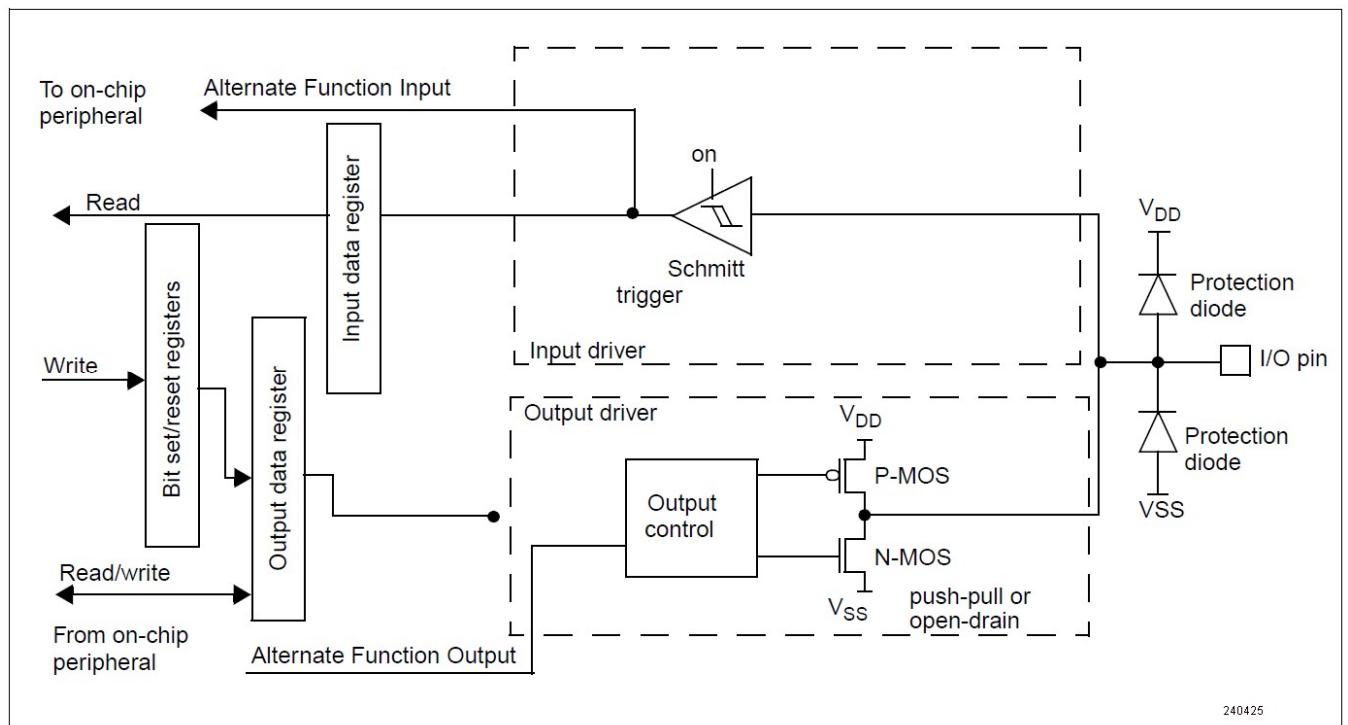


Figure 22. Alternate function configuration

### 10.1.10 Analog input configuration

When the I/O port is configured as analog input configuration:

- The Output Buffer is disabled.
- The Schmitt Trigger Input is de-activated providing zero consumption for every analog value of the I/O pin. The output of the Schmitt Trigger is forced to '0'.
- The weak pull-up and pull-down resistors are disabled.
- A read access to the Input Data register gets the value '0'.

The figure below shows the high impedance input configuration of the I/O port bit:

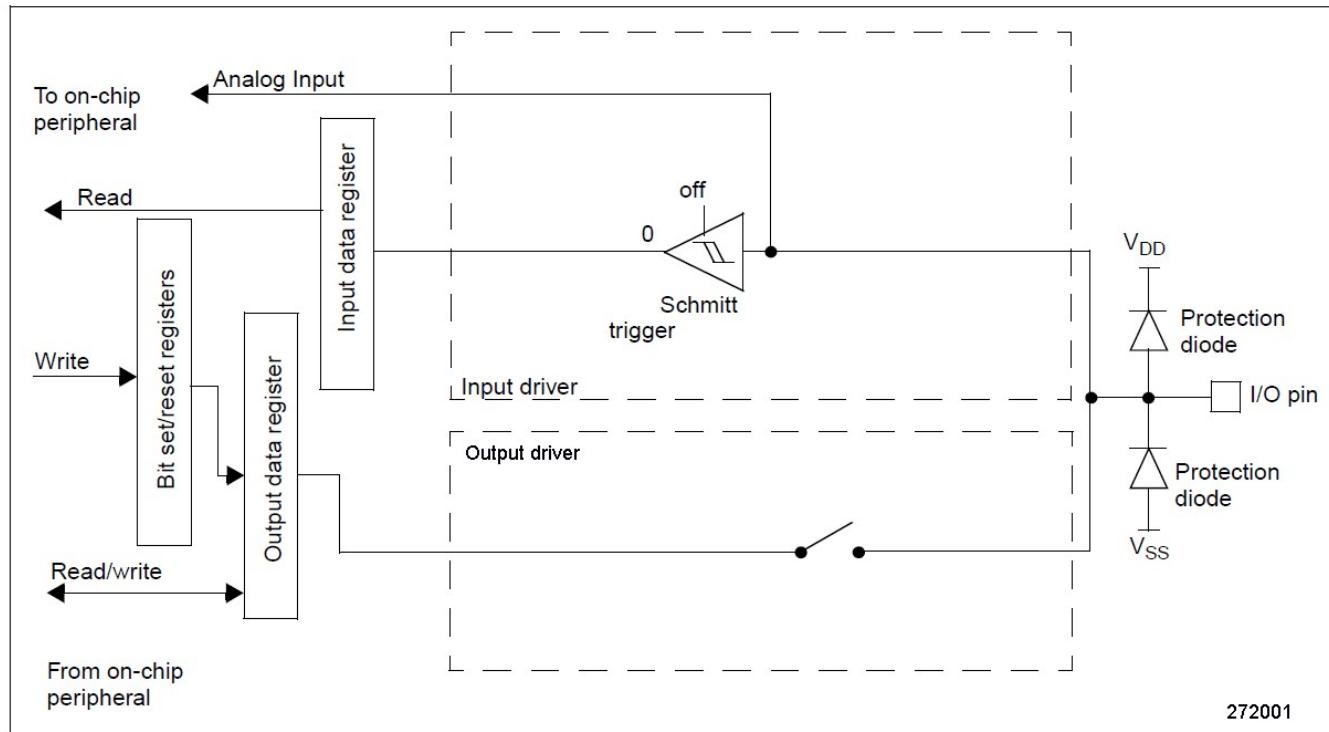


Figure 23. High impedance-analog input configuration

### 10.1.11 GPIO configurations for device peripherals

The following tables give the GPIO configurations of the device peripherals:

Table 26. Advanced timer TIM1

TIM1 pin	Configuration	GPIO configuration
TIM1_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function output push-pull
TIM1_CHxN	Complementary output channel x	Alternate function output push-pull
TIM1_BKIN	Break input	Input floating
TIM1_ETR	External trigger timer input	Input floating

Table 27. General-purpose timers TIM2/3/14/16/17

TIM2/3/14/16/17 pinout	Configuration	GPIO configuration
TIM2/3/14/16/17_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function output push-pull
TIM2/3_ETR	External trigger timer input x	Input floating

Table 28. UART

UART pin	Configuration	GPIO configuration
UARTx_TX	Data sent via serial port	Alternate function output push-pull
UARTx_RX	Data received via serial port	Input floating/Input pull-up
UARTx_RTS	Hardware flow control	Alternate function output push-pull
UARTx_CTS	Hardware flow control	Input floating/Input pull-up

Table 29. SPI

SPI pin	Configuration	GPIO configuration
SPIx_SCK	Master mode	Alternate function output push-pull
	Slave mode	Input floating
SPIx_MOSI	Full duplex/master	Alternate function output push-pull
	Full duplex/slave	Input floating/Input pull-up
SPIx_MISO	Full duplex/master	Input floating/Input pull-up
	Full duplex/slave	Alternate function output push-pull
SPIx_NSS	Hardware master/slave	Input floating/Input pull-up/Input pull-down
	Hardware master/NSS output enabled	Alternate function output push-pull
	Software mode	Not used. Can be used as a GPIO

Table 30. I2C

I2C pin	Configuration	GPIO configuration
I2Cx_SCL	I2C clock	Alternate function output open drain
I2Cx_SDA	I2C data	Alternate function output open drain

Table 31. CAN

CAN pin	GPIO configuration
CAN_TX	Alternate function output push-pull
CAN_RX	Input floating/Input pull-up

Table 32. ADC

ADC pin	GPIO configuration
ADC	Analog input

Table 33. Other IOs

Pins	Configuration	GPIO configuration
MCO	Clock output	Alternate function output push-pull
EXTI input lines	External interrupt input	Input floating/Input pull-up/Input pull-down

## 10.2. Alternate function I/O and debug configuration

To optimize the number of peripherals available, it is possible to remap some alternate functions to some other pins. This is achieved by setting the alternate function register (AFR). In this case, the alternate functions are no longer mapped to their original assignations.

### 10.2.1 OSC\_IN/OSC\_OUT used as GPIO port PD0/PD1

The external oscillator pins OSC\_IN/OSC\_OUT can be used as general-purpose I/O PD0/PD1 by turning off the high-speed internal clock first and then setting the alternate function register (AFR).

PD0/PD1 must be set as an analog port when used as OSC\_IN/OSC\_OUT.

Note: The external interrupt/event function is not remapped.

### 10.2.2 SWD alternate function remapping

The debug interface signals are mapped on the GPIO ports as shown in the table below.

Table 34. Debug interface signal

Alternate function	GPIO port
SWDIO	PA13
SWCLK	PA14

To optimize the number of free GPIOs during debugging, this mapping can be configured in different ways by setting the alternate function remapping and alternate function register.

## 10.3 GPIO register description

Table 35. Overview of GPIO registers

Offset	Acronym	Register Name	Reset	Section
0x00	GPIOx_CRL	Port configuration register low	0x44444444	Section 10.3.1
0x04	GPIOx_CRH	Port configuration register high	0x44444444	Section 10.3.2
0x08	GPIOx_IDR	Port input data register	0x0000XXXX	Section 10.3.3
0x0C	GPIOx_ODR	Port output data register	0x00000000	Section 10.3.4
0x10	GPIOx_BSRR	Port bit set/reset register	0x00000000	Section 10.3.5
0x14	GPIOx_BRR	Port bit reset register	0x00000000	Section 10.3.6
0x18	GPIOx_LCKR	Port configuration lock register	0x00000000	Section 10.3.7

Offset	Acronym	Register Name	Reset	Section
0x1C	GPIOx_DCR	Port output open drain control register	0x00000000	Section 10.3.8
0x20	GPIOx_AFRL	Port alternate function register low	0xFFFFFFFF	Section 10.3.9
0x24	GPIOx_AFRH	Port alternate function register high	0xFFFFFFFF	Section 10.3.10

### 10.3.1 Port configuration register low (GPIOx\_CRL)(x = A..D)

Address offset: 0x00

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7	MODE7	CNF6	MODE6	CNF5	MODE5	CNF4	MODE4								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3	MODE3	CNF2	MODE2	CNF1	MODE1	CNF0	MODE0								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31: 30	CNFy	rw	0x01	Port x configuration bits (0...7) These bits are written by software to configure the corresponding I/O port. Refer to Table 24: Port bit configuration table.
27: 26				In input mode (MODE = 00): 00: Analog input mode
23: 22				01: Floating input mode
19: 18				10: Input pull-up/pull-down mode
15: 14				11: Reserved
11: 10				In output mode (MODE > 00): 00: General purpose output push-pull
7: 6				01: General purpose output open-drain
3: 2				10: Alternate function output push-pull
				11: Alternate function output open-drain
29: 28	MODEy	rw	0x00	Port x mode bits (y = 0...7) These bits are written by software to configure the corresponding I/O port. Refer to Table 24: Port bit configuration table.
25: 24				00: Input mode (reset state)
21: 20				01: Output mode
17: 16				10: Reserved
13: 12				11: Reserved
9: 8				
5: 4				
1: 0				

### 10.3.2 Port configuration register high (GPIOx\_CRH)(x = A..D)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15	MODE15		CNF14		MODE14		CNF13		MODE13		CNF12		MODE12		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11	MODE11		CNF10		MODE10		CNF9		MODE9		CNF8		MODE8		

Bit	Field	Type	Reset	Description
31: 30	CNFy	rw	0x01	Port x configuration bits (8...15) These bits are written by software to configure the corresponding I/O port. Refer to Table 24: Port bit configuration table.
27: 26				In input mode (MODE = 00): 00: Analog input mode
23: 22				01: Input floating mode
19: 18				10: Input pull-up/pull-down mode
15: 14				11: Reserved
11: 10				In output mode (MODE[1: 0] > 00): 00: General purpose output push-pull 01: General purpose output open-drain
7: 6				10: Alternate function output push-pull 11: Alternate function output open-drain
3: 2				
29: 28	MODEy	rw	0x00	Port x mode bits (y = 8...15) These bits are written by software to configure the corresponding I/O port. Refer to Table 24: Port bit configuration table.
25: 24				00: Input mode (reset state)
21: 20				01: Output mode
17: 16				10: Reserved
13: 12				11: Reserved
9: 8				
5: 4				
1: 0				

### 10.3.3 Port input data register (GPIOx\_IDR)(x = A..D)

Address offset: 0x08

Reset value: 0x0000 XXXX

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR															

r

Bit	Field	Type	Reset	Description
31: 16	Reserved			always read as 0.

15: 0	IDRy	r	0xXXXX	Port input data ( $y = 0..15$ ) These bits are read only and can be accessed in Word (16-bit) mode only. They contain the input value of the corresponding I/O port.
-------	------	---	--------	---

### 10.3.4 Port output data register (GPIOx\_ODR)( $x = A..D$ )

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31: 16	Reserved			always read as 0.
15: 0	ODRy	rw	0x0000	Port output data ( $y = 0..15$ ) These bits are read only and can be accessed in Word (16-bit) mode only. Note: The ODR bits can be individually set and cleared by writing to the GPIOx_BSRR ( $x = A..D$ ).

### 10.3.5 Port bit set/reset register (GPIOx\_BSRR)( $x = A..D$ )

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
31: 16	BRy	w	0x0000	Port x Reset bit y ( $y = 0..15$ ) These bits are write-only and can be accessed in Word (32-bit) mode only.

					1: Clear the corresponding ODRy bit as '0'
15: 0	BSy	w	0x0000	Port x Set bit y (y = 0...15)	
				These bits are write-only and can be accessed in Word (32-bit) mode only.	
				0: No effect on the corresponding ODRy bit	
				1: Set the corresponding ODRy bit to '1'	

### **10.3.6 Port bit reset register (GPIOx\_BRR)(x = A..D)**

Address offset: 0x14

Reset value: 0x0000 0000

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

A horizontal scale with numerical labels from 15 to 0. The labels are: 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0. A red rectangular box highlights the number 7.

Bit	Field	Type	Reset	Description
31: 16	Reserved			always read as 0
15: 0	BRy	w	0x0000	<p>Port x Reset bit y (y = 0...15)</p> <p>These bits are write-only and can be accessed in Word (16-bit) mode only.</p> <p>0: No effect on the corresponding ODRy bit</p> <p>1: Clear the corresponding ODRy bit as '0'</p>

#### 10.3.7 Port configuration lock register (GPIOx\_LCKR)(x = A..D)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit it is no longer possible to modify the value of the port bit until the next reset.

Each lock bit freezes the corresponding 4 bits of the control register (CRL, CRH).

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31: 17	Reserved			always read as 0
16	LCKK	rw	0x00	<p>Lock key</p> <p>This bit can be read anytime. It can only be modified using the Lock Key Writing Sequence.</p> <p>0: Port configuration lock key not active</p> <p>1: Port configuration lock key active. GPIOx_LCKR register is locked until the next reset.</p> <p>LOCK key writing sequence:</p> <p>Write 1-&gt; Write 0-&gt; Write 1-&gt; Read 0-&gt; Read 1 (The last read is optional but confirms that the lock is active)</p> <p>Note: During the LOCK Key Writing sequence, the value of LCK must not change. Any error in the lock key writing sequence will abort the lock.</p>
15: 0	LCKy	rw	0x00	<p>Port x Lock bit y (y = 0...15)</p> <p>These bits are read write but can only be written when the LCKK bit is 0.</p> <p>0: Port configuration not locked</p> <p>1: Port configuration locked</p>

### 10.3.8 Port output open drain control register (GPIOx\_DCR)(x = A..D)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PX15	PX14	PX13	PX12	PX11	PX10	PX9	PX8								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PX7	PX6	PX5	PX4	PX3	PX2	PX1	PX0								
rw	rw	rw	rw	rw	rw	rw	rw	rw							

Bit	Field	Type	Reset	Description
31: 2	PX15-PX1	rw	0x00	see PX0
1: 0	PX0	rw	0x00	<p>PX0[1: 0]:</p> <p>2'b11: output open-drain mode with port pull-up</p> <p>2'b01: output open-drain mode with port pull-down</p> <p>2'bx0: output open-drain mode without port pull-up/pull-down</p>

### 10.3.9 Port alternate function register low (GPIOx\_AFRL)(x = A..D)

Address offset: 0x20

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15				AFR14				AFR13				AFR12			
rw				rw				rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7				AFR6				AFR5				AFR4			
rw				rw				rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR3				AFR2				AFR1				AFR0			
rw				rw				rw				rw			

Bit	Field	Type	Reset	Description
31: 0	AFRy	rw	0xFFFF FFFF	Port x alternate function bit y (y = 0...7) These bits can be accessed by software to configure the IO alternate function. 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7

### 10.3.10 Port alternate function register high (GPIOx\_AFRH)(x = A..D)

Address offset: 0x24

Reset value: GPIOA\_AFRH: 0xF00F FFFF

GPIOB\_AFRH: 0xFFFF FFFF

GPIOC\_AFRH: 0xFFFF FFFF

GPIOD\_AFRH: 0xFFFF FFFF

	AFR11	AFR10	AFR9	AFR8
	rw	rw	rw	rw
Bit	Field	Type	Reset	Description
31: 0	AFRy	rw		<p>Port x alternate function bit y (y = 8...15)            These bits can be accessed by software            to configure the IO alternate function.</p> <p>0000: AF0            0001: AF1            0010: AF2            0011: AF3            0100: AF4            0101: AF5            0110: AF6            0111: AF7</p>

# 11

# Interrupt and Events (EXTI)

## Interrupt and Events (EXTI)

### 11.1 Nested vectored interrupt controller

#### Features

- All interrupts are maskable (except NMI)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The nested vectored interrupt controller (NVIC) and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to CPU Technical Reference Manual.

#### 11.1.1 SysTick calibration value register

The SysTick calibration value is set to 9000, which gives a reference time base of 1ms with the SysTick clock set to 9MHz (HCLK/4, HCLK = 36MHz).

#### 11.1.2 Interrupt and exception vectors

The following table is the vector table for this product series.

Table 36. Vector table for this product series

Position	Priority	Type of priority	Name	Description	Address
-	-	-	-	Reserved	0x0000_0000
-3	fixed		Reset	Reset	0x0000_0004
-2	fixed		NMI	Non maskable interrupt The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
-1	fixed		HardFault	All types of fault	0x0000_000C
0	settable		MemManage	Memory management	0x0000_0010
1	settable		BusFault	Pre-fetch fault, memory access fault	0x0000_0014
2	settable		UsageFault	Undefined instruction or illegal state	0x0000_0018

	-	-	-	Reserved	0x0000_001C ~ 0x0000_002B
--	---	---	---	----------	---------------------------------

Position	Priority	Type of priority	Name	Description	Address
	3	settable	SVCALL	System service call via SWI instruction	0x0000_002C
	4	settable	DebugMonitor	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV	Pendable request for system service	0x0000_0038
	6	settable	SysTick	System tick timer	0x0000_003C
0	7	settable	WWDG_IWDG	Window watchdog and independent watchdog interrupts	0x0000_0040
1	8	settable	PVD	PVD through EXTI16 detection interrupt	0x0000_0044
2	9	settable	RTC_BKP	RTC and BKP global interrupt	0x0000_0048
3	10	settable	FLASH	Flash global interrupt	0x0000_004C
4	11	settable	RCC_CRS	RCC_CRS global interrupt	0x0000_0050
5	12	settable	EXTI0_1	EXTI line [1: 0] interrupt	0x0000_0054
6	13	settable	EXTI2_3	EXTI line [3:2] interrupt	0x0000_0058
7	14	settable	EXTI4_15	EXTI line [15:4] interrupt	0x0000_005C
8	15	settable	HWDIV	HWDIV global interrupt	0x0000_0060
9	16	settable	DMA1 channel 1	DMA1 channel 1 global interrupt	0x0000_0064
10	17	settable	DMA1 channel 2&3	DMA1 channel 2&3 global interrupts	0x0000_0068
11	18	settable	DMA1 channel 4&5	DMA1 channel 4&5 global interrupts	0x0000_006C
12	19	settable	ADC1_COMP	ADC1 interrupt and comparator interrupt (Combined with EXIT19&20)	0x0000_0070
13	20	settable	TIM1_BRK_UP_TRG_COM	TIM1 brake, update, trigger, and COM interrupt	0x0000_0074
14	21	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000_0078
15	22	settable	TIM2	TIM2 global interrupt	0x0000_007C
16	23	settable	TIM3	TIM3 global interrupt	0x0000_0080
17	24	settable	-	Reserved	0x0000_0084
18	25	settable	-	Reserved	0x0000_0088

Position	Priority	Type of priority	Name	Description	Address
19	26	settable	TIM14	TIM14 global interrupt	0x0000_008C
20	27	settable	-	Reserved	0x0000_0090
21	28	settable	TIM16	TIM16 global interrupt	0x0000_0094
22	29	settable	TIM17	TIM17 global interrupt	0x0000_0098
23	30	settable	I2C1	I2C1 global interrupt	0x0000_009C
24	31	settable	-	Reserved	0x0000_00A0
25	32	settable	SPI1	SPI1 global interrupt	0x0000_00A4
26	33	settable	SPI2	SPI2 global interrupt	0x0000_00A8
27	34	settable	UART1	UART1 global interrupt	0x0000_00AC
28	35	settable	UART2	UART2 global interrupt	0x0000_00B0
29	36	settable	CSM	CSM global interrupt	0x0000_00B4
30	37	settable	CAN	CAN global interrupt	0x0000_00B8
31	38	settable	USB	USB global interrupt (EXTI18)	0x0000_00BC

## 11.2 External interrupt/event controller (EXTI)

The external interrupt/event controller (EXTI) manages external and internal asynchronous events/interrupts and generates corresponding event requests to CPU/interrupt controller and wakeup requests to power management.

Edge detectors generate event/interrupt requests. Each input line can be independently configured to select the input type (pulse or pending) and the corresponding trigger event (rising or falling or both). Each input line can also be masked independently. A pending register maintains the status line of the interrupt requests.

### 11.2.1 Main features

The EXTI controller main features are the following:

- Independent trigger and mask on each interrupt/event line
- Dedicated status bit for each interrupt line
- Generation of software event/interrupt requests
- Detection of external signal with pulse width lower than APB2 clock period. Refer to the electrical characteristics section of the datasheet for details on this parameter.

### 11.2.2 Block diagram

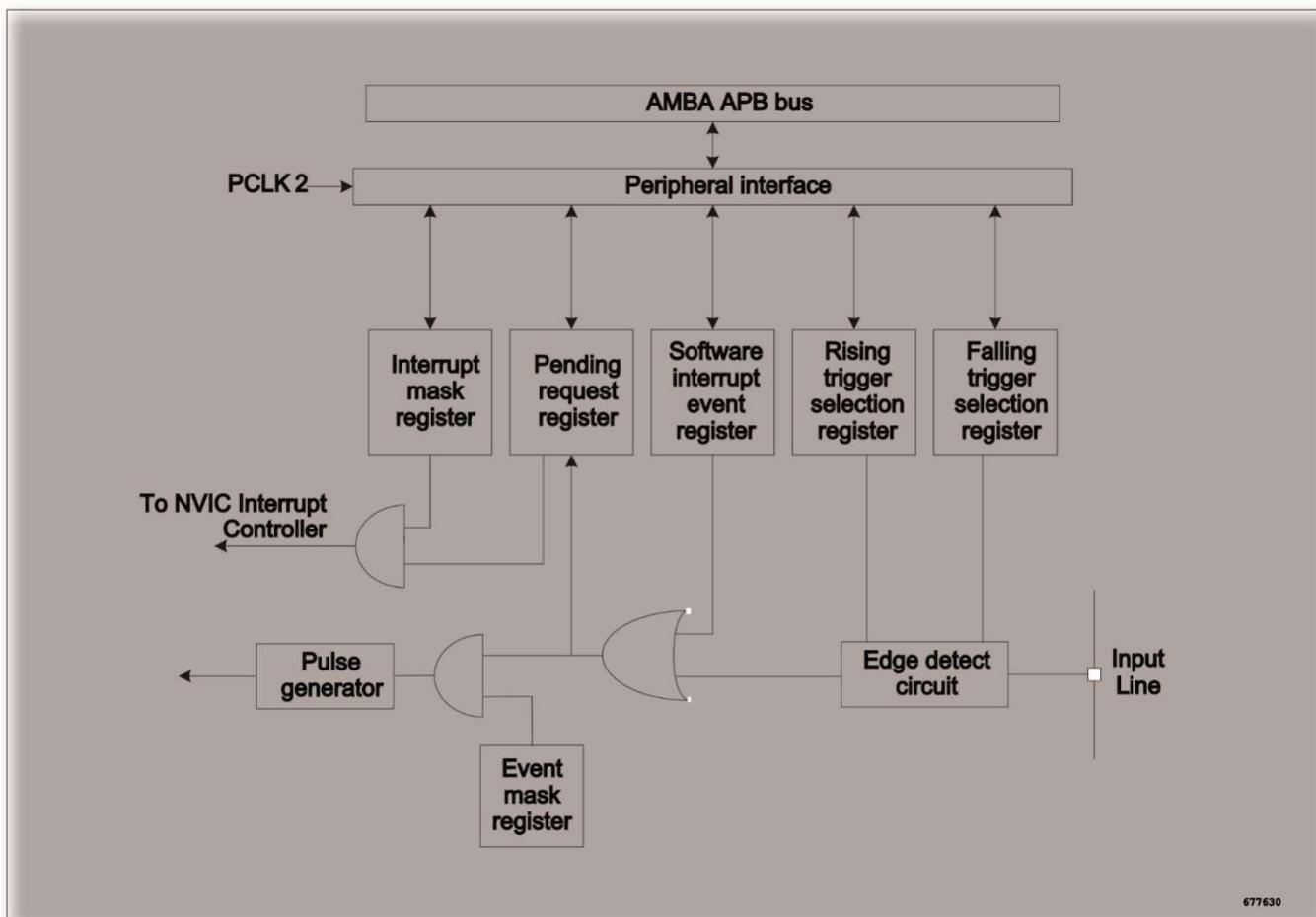


Figure 24. External interrupt/event controller block diagram

### 11.2.3 Wakeup event management

It is possible to wake up the core (WFE) by handling external or internal events. The wakeup event can be generated either by:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the CPU system control register.

When the CPU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC interrupt channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC interrupt channel pending bit as the pending bit corresponding to the event line is not set.

To use an external I/O port as a wakeup event, refer to functional description below.

### 11.2.4 Functional description

To generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a '1' to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt

request is generated. The pending bit corresponding to the interrupt line is also set to ‘1’. This interrupt request is reset by writing a ‘1’ in the pending register.

To generate the event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a ‘1’ to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event request pulse is generated. The corresponding pending bit is not set to ‘1’.

An interrupt/event request can also be generated by software by writing a ‘1’ in the software interrupt/event register.

### **Hardware interrupt selection**

To configure multiple lines as interrupt sources, use the following procedure:

- Configure the mask bits of the Interrupt lines (EXTI\_IMR)
- Configure the Trigger Selection bits of the Interrupt lines (EXTI\_RTSR and EXTI\_FTSR)
- Configure the enable and mask bits that control the NVIC interrupt channel mapped to the External Interrupt Controller (EXTI) so that a request coming from interrupt lines can be correctly acknowledged.

### **Hardware event selection**

To configure multiple lines as event sources, use the following procedure:

- Configure the mask bits of Event lines (EXTI\_EMR)
- Configure the Trigger Selection bits of the Event lines (EXTI\_RTSR and EXTI\_FTSR)

### **Software interrupt/event selection**

Multiple lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

- Configure the mask bits of Interrupt/Event lines (EXTI\_IMR, EXTI\_EMR)
- Set the required bit of the software interrupt register (EXTI\_SWIER)

### **11.2.5 External interrupt/event line mapping**

The GPIOs are connected to the 16 external interrupt/event lines in the following manner:

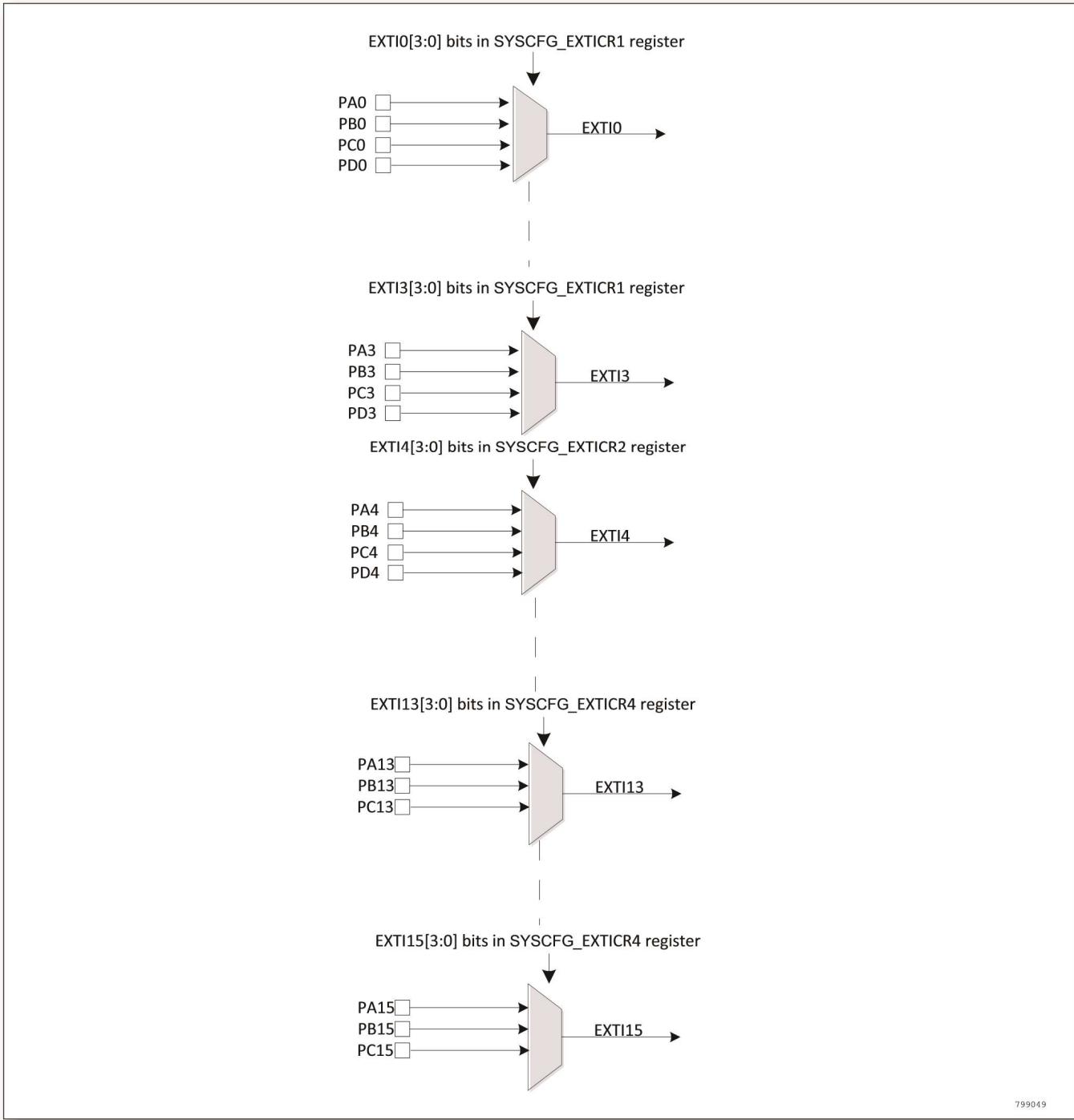


Figure 25. External interrupt GPIO mapping

Note: The corresponding GPIOs in the above diagram may vary due to the actual chip package and they are subject to the actual chip.

The other EXTI/event controllers are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC alarm event
- EXTI line 18 is connected to the USB bus pending interrupt
- EXTI line 19 is connected to the comparator 1 output
- EXTI line 20 is connected to the comparator 2 output

- EXTI line 24 is connected to the IWDG interrupt

## 11.3 EXTI register description

Table 37. Overview of EXTI registers

Offset	Acronym	Register Name	Reset	Section
0x00	EXTI_IMR	Interrupt mask register	0x00000000	Section 11.3.1
0x04	EXTI_EMR	Event mask register	0x00000000	Section 11.3.2
0x08	EXTI_RTSR	Rising trigger selection register	0x00000000	Section 11.3.3
0x0C	EXTI_FTSR	Falling trigger selection register	0x00000000	Section 11.3.4
0x10	EXTI_SWIER	Software interrupt event register	0x00000000	Section 11.3.5
0x14	EXTI_PR	Pending register	0x00000000	Section 11.3.6

### 11.3.1 Interrupt mask register (EXTI\_IMR)

Address offset: 0x00

Reset value: 0x0000 0000

Bit	Field	Type	Reset	Description
31:25	Reserved			always read as 0.
24	IMRx	rw	0x00	Interrupt Mask on line x 1 = Interrupt request from line x is not masked 0 = Interrupt request from line x is masked
23:21	Reserved			always read as 0
20:0	IMRx	rw	0x00	Interrupt Mask on line x 1 = Interrupt request from line x is not masked 0 = Interrupt request from line x is masked

### 11.3.2 Event mask register (EXTI\_EMR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							Reserved		EMR24	Reserved	EMR20	EMR19	EMR18	EMR17	EMR16
									rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMR15	EMR14	EMR13	EMR12	EMR11	EMR10	EMR9	EMR8	EMR7	EMR6	EMR5	EMR4	EMR3	EMR2	EMR1	EMR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:25	Reserved			always read as 0.
24	EMRx	rw	0x00	Event Mask on line x 1 = Event request from line x is not masked 0 = Event request from line x is masked
23:21	Reserved			always read as 0
20:0	EMRx	rw	0x00	Event Mask on line x 1 = Event request from line x is not masked 0 = Event request from line x is masked

### 11.3.3 Rising trigger selection register (EXTI\_RTSR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							Reserved		TR24	Reserved	TR20	TR19	TR18	TR17	TR16
									rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:25	Reserved			always read as 0.
24	TRx	rw	0x00	Rising trigger event configuration bit of line x 1 = Rising trigger enabled (for Event and Interrupt) for input line x. 0 = Rising trigger disabled (for Event and Interrupt) for input line x.
23:21	Reserved			always read as 0
20:0	TRx	rw	0x00	Rising trigger event configuration bit of line x 1 = Rising trigger enabled (for Event and Interrupt) for input line x. 0 = Rising trigger disabled (for Event and Interrupt) for input line x.

### 11.3.4 Falling trigger selection register (EXTI\_FTSR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TR24	Reserved				TR20	TR19	TR18	TR17	TR16		
							rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:25	Reserved			always read as 0.
24	TRx	rw	0x00	Falling trigger event configuration bit of line x 1 = Falling trigger enabled (for Event and Interrupt) for input line x. 0 = Falling trigger disabled (for Event and Interrupt) for input line x.
23:21	Reserved			always read as 0
20:0	TRx	rw	0x00	Falling trigger event configuration bit of line x 1 = Falling trigger enabled (for Event and Interrupt) for input line x. 0 = Falling trigger disabled (for Event and Interrupt) for input line x.

### 11.3.5 Software interrupt event register (EXTI\_SWIER)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				SWIER 24	Reserved				SWIER 20	SWIER 19	SWIER 18	SWIER 17	SWIER 16		
					rw				rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIE R9	SWIE R8	SWIE R7	SWIE R6	SWIE R5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:25	Reserved			always read as 0.
24	SWIERx	rw	0x00	Software interrupt on line x Writing a '1' to this bit when it is set to '0' sets the corresponding pending bit in the EXTI_PR. At that time, an interrupt generates if the interrupt is enabled in the EXTI_INTMASK and EXTI_EVNTMASK. Note: This bit is cleared as '0' by clearing the corresponding bit of EXTI_PEND (by writing a 1 into the bit).

Bit	Field	Type	Reset	Description
23:21	Reserved			always read as 0.
20:0	SWIERx	rw	0x00	<p>Software interrupt on line x</p> <p>Writing a '1' to this bit when it is set to '0' sets the corresponding pending bit in the EXTI_PR. At that time, an interrupt generates if the interrupt is enabled in the EXTI_INTMASK and EXTI_EVNTMASK.</p> <p>Note: This bit is cleared as '0' by clearing the corresponding bit of EXTI_PEND (by writing a 1 into the bit).</p>

### 11.3.6 Pending register (EXTI\_PR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							PR24	Reserved			PR20	PR19	PR18	PR17	PR16
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1															

Bit	Field	Type	Reset	Description
31:25	Reserved			always read as 0.
24	PRx	rc_w1	0x00	<p>Pending bit</p> <p>1 = Selected trigger request occurred</p> <p>0 = No trigger request occurred</p> <p>This bit is set to '1' when the selected edge event arrives on the external interrupt line.</p> <p>This bit is cleared by writing a '1' into the bit or changing the polarity of edge detection.</p>
23:21	Reserved			always read as 0
20:0	PRx	rc_w1	0x00	<p>Pending bit</p> <p>1: Selected trigger request occurred</p> <p>0: No trigger request occurred</p> <p>This bit is set to '1' when the selected edge event arrives on the external interrupt line.</p> <p>This bit is cleared by writing a '1' into the bit or changing the polarity of edge detection.</p>

# Direct memory access controller (DMA)

Direct memory access controller (DMA)

## 12.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory as well as memory to memory.

Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.

The DMA controller has 5 channels, each dedicated to managing memory access requests from one or more peripherals.

## 12.2 DMA main features

- 5 independently configurable channels
- Each channel is connected to dedicated hardware DMA requests; software trigger is also supported on each channel. This configuration is done by software.
- Priorities between five requests are software programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (request 0 has priority over request 1, etc.)
- Independent source and destination data zone transfer size (byte, half word, word), emulating packing and unpacking. Source/destination addresses must be aligned on the data transfer size.
- Support for circular buffer management
- 3 event flags (DMA Half Transfer, DMA Transfer Complete and DMA Transfer Error) logically ORed together in a single interrupt request for each channel
- Memory-to-memory transfer
- Peripheral-to-memory and memory-to-peripheral transfers
- Access to SRAM, peripheral SRAM, APB1, APB2 and AHB peripherals as source and destination
- Programmable number of data to be transferred: up to 65536

The functional block diagram is shown as below:

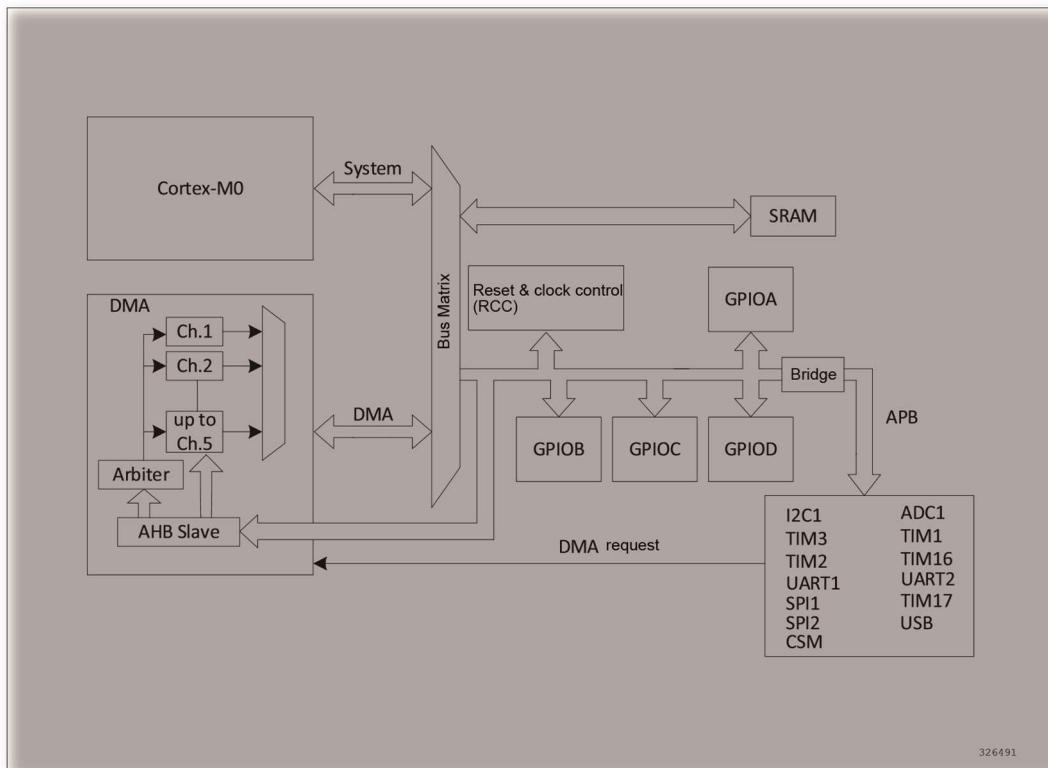


Figure 26. DMA block diagram

## 12.3 DMA functional description

The DMA controller performs direct memory transfer by sharing the system data bus with the CPU core. The DMA request may stop the CPU access to the system bus for some bus cycles, when the CPU and DMA are targeting the same destination (RAM or peripheral). The bus arbiter implements round-robin scheduling, thus ensuring at least half of the system bus bandwidth (both to memory and peripheral) for the CPU.

### 12.3.1 DMA transactions

After an event, the peripheral sends a request signal to the DMA controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA controller accesses the peripheral, an Acknowledge is sent to the peripheral by the DMA controller. The peripheral releases its request as soon as it gets the Acknowledge from the DMA controller. Once the request is released by the peripheral, the DMA controller revokes the Acknowledge. If there are more requests, the peripheral can initiate the next transaction.

In summary, each DMA transfer consists of three operations:

1. The loading of data from the peripheral data register or a designed location in memory unit addressed through the DMA\_CMARx register.
2. The storage of the data loaded to the peripheral data register or a designed location in memory unit addressed through the DMA\_CMARx register.
3. The post-decrementing of the DMA\_CNDTRx register which contains the number of transactions that have still to be performed.

### 12.3.2 Arbiter

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences. The priorities are managed in two stages:

- Software: each channel priority can be configured in the DMA\_CCRx register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: if 2 requests have the same software priority level, the channel with the lower number will get priority versus the channel with the higher number. For example, channel 2 gets priority over channel 4.

### 12.3.3 DMA channels

Each channel can handle DMA transfer between a peripheral register located at a fixed address and a memory address. The amount of data to be transferred (up to 65536) is programmable. The register which contains the amount of data items to be transferred is decremented after each transaction.

### Programmable data sizes

Transfer data sizes of the peripheral and memory are fully programmable through the PSIZE and MSIZE bits in the DMA\_CCRx register.

### Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented after each transaction depending on the PINC and MINC bits in the DMA\_CCRx register. If incremented mode is enabled, the address of the next transfer will be the address of the previous one incremented by 1, 2 or 4 depending on the chosen data size. The first transfer address is stored in the DMA\_CPARx/DMA\_CMARx registers. If the channel is configured in non-circular mode, no DMA request is served after the last transfer (that is once the number of data items to be transferred has reached zero).

### Channel configuration

The following sequence should be followed to configure a DMA channel x (where x is the channel number):

1. Set the peripheral register address in the DMA\_CPARx register. The data will be moved from/to this address to/from the memory after the peripheral data transfer request.
2. Set the memory address in the DMA\_CMARx register. The data transferred will be written to or read from this memory after the peripheral data transfer request.
3. Configure the total number of data to be transferred in the DMA\_CNDTRx register. After each data transfer request, this value will be decremented.
4. Configure the channel priority using the PL[1:0] bits in the DMA\_CCRx register.
5. Configure data transfer direction, circular mode, peripheral & memory incremented mode, peripheral & memory data size, and interrupt after half or full transfer in the DMA\_CCRx register.
6. Activate the channel by setting the ENABLE bit in the DMA\_CCRx register. As soon as the DMA channel is enabled, it can serve any DMA request from the peripheral connected on the channel.

Once half of the bytes are transferred, the half-transfer flag (HTIF) is set and an interrupt request is generated if the Half-Transfer Interrupt Enable bit (HTIE) is set. At the end of the transfer, the Transfer Complete Flag (TCIF) is set and an interrupt request is generated if the Transfer Complete Interrupt Enable bit (TCIE) is set.

### Circular mode

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA\_CCRx register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase if it reaches 0, and the DMA requests continue to be served.

### Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This mode is called Memory to Memory mode.

If the MEM2MEM bit in the DMA\_CCRx register is set, then the DMA channel initiates transfers as soon as it is enabled by software by setting the Enable bit (EN) in the DMA\_CCRx register. The DMA transfer stops once the DMA\_CNDTRx register reaches zero. Memory to Memory mode may not be used at the same time as Circular mode.

#### 12.3.4 Programmable data transfer width, data alignment and endians

When PSIZE and MSIZE are not equal, the DMA module performs some data alignments as described in the following table.

Table 38. Programmable data transfer width & endian behavior (when bits PINC = MINC = 1)

Source port width	Destination port width	Number of data items to transfer	Source (address/data)	Transfer operations	Destination (address/data)
8	8	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: READ B0[7:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B1[7:0] @0x1 then WRITE B1[7:0] @0x1 3: READ B2[7:0] @0x2 then WRITE B2[7:0] @0x2 4: READ B3[7:0] @0x3 then WRITE B3[7:0] @0x3	0x0/B0 0x1/B1 0x2/B2 0x3/B3
8	16	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: READ B0[7:0] @0x0 then WRITE 00B0[15:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 00B1[15:0] @0x2 3: READ B2[7:0] @0x2 then WRITE 00B2[15:0] @0x4 4: READ B3[7:0] @0x3 then WRITE 00B3[15:0] @0x6	0x0/00B0 0x2/00B1 0x4/00B2 0x6/00B3
8	32	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: READ B0[7:0] @0x0 then WRITE 000000B0[31:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 000000B1[31:0] @0x4 3: READ B2[7:0] @0x2 then WRITE 000000B2[31:0] @0x8	0x0/000000B0 0x4/000000B1 0x8/000000B2 0xC/000000B3

Source port width	Destination port width	Number of data items to transfer	Source (address/data)	Transfer operations	Destination (address/data)
				4: READ B3[7:0] @0x3 then WRITE 000000B3[31:0] @0xC	
16	8	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: READ B1B0[15:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B2[7:0] @0x1 3: READ B5B4[15:0] @0x4 then WRITE B4[7:0] @0x2 4: READ B7B6[15:0] @0x6 then WRITE B6[7:0] @0x3	0x0/B0 0x1/B2 0x2/B4 0x3/B6
16	16	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: READ B1B0[15:0] @0x0 then WRITE B1B0[15:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B3B2[15:0] @0x2 3: READ B5B4[15:0] @0x4 then WRITE B5B4[15:0] @0x4 4: READ B7B6[15:0] @0x6 then WRITE B7B6[15:0] @0x6	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6
16	32	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: READ B1B0[15:0] @0x0 then WRITE 0000B1B0[31:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE 0000B3B2[31:0] @0x4 3: READ B5B4[15:0] @0x4 then WRITE 0000B5B4[31:0] @0x8 4: READ B7B6[15:0] @0x6 then WRITE 0000B7B6[31:0] @0xC	0x0/0000B1B0 0x4/0000B3B2 0x8/0000B5B4 0xC/0000B7B6
32	8	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BC[7:0] @0x3	0x0/B0 0x1/B4 0x2/B8 0x3/BC
32	16	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B1B0[15:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B5B4[15:0] @0x2 3: READ BBBAB9B8[31:0] @0x8 then WRITE B9B8[15:0] @0x4 4: READ BFBEBDBC[31:0] @0xC then WRITE BDBC[15:0] @0x6	0x0/B1B0 0x2/B5B4 0x4/B9B8 0x6/BDBC

Source port width	Destination port width	Number of data items to transfer	Source (address/data)	Transfer operations	Destination (address/data)
32	32	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B3B2B1B0[31:0] @0x0  2: READ B7B6B5B4[31:0] @0x4 then WRITE B7B6B5B4[31:0] @0x4  3: READ BBBAB9B8[31:0] @0x8 then WRITE BBBAB9B8[31:0] @0x8  4: READ BFBEBDBC[31:0] @0xC then WRITE BFBEBDBC[31:0] @0xC	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC

### Addressing an AHB peripheral that does not support byte or halfword write operations

When the DMA initiates an AHB byte or halfword write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus. So when the used AHB device does not support byte or halfword write operations (when HSIZE is not used by the module) and does not generate any error, the DMA writes the 32 HWDATA bits as shown in the two examples below:

- To write the halfword “0xABCD”, the DMA sets the HWDATA bus to “0xABCDABCD” with HSIZE = HalfWord.
- To write the byte “0xAB”, the DMA sets the HWDATA bus to “0xABABABAB” with HSIZE = Byte.

Assuming that the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take the HSIZE data into account, it will transform any AHB byte or halfword operation into a 32-bit APB operation in the following manner:

- An AHB byte write operation of the data “0xB0” to the address 0x0 (or to 0x1, 0x2 or 0x3) will be converted to an APB word write operation of the data “0xB0B0B0B0” to the address 0x0.
- An AHB halfword write operation of the data “0xB1B0” to 0x0 (or to 0x2) will be converted to an APB word write operation of the data “0xB1B0B1B0” to the address 0x0.

For instance, to write the APB backup registers (16-bit registers aligned to a 32-bit address boundary), the memory source size (MSIZE) must be configured to “16-bit” and the peripheral destination size (PSIZE) to “32-bit”.

### 12.3.5 Error management

A DMA transfer error can be generated by reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or a write access, the faulty channel is automatically disabled through a hardware clear of its EN bit in the corresponding Channel configuration register (DMA\_CCRx). The channel's transfer error interrupt flag (TEIF) in the DMA\_IFR register is set and an interrupt is generated if the transfer error interrupt enable bit (TEIE) in the DMA\_CCRx register is set.

### 12.3.6 Interrupts

An interrupt can be produced on a Half-transfer, Transfer complete or Transfer error for each DMA channel. Separate interrupt enable bits are available for flexibility.

Table 39. DMA interrupt requests

Interrupt Event	Event Flag	Enable Control Bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

### 12.3.7 DMA request mapping

#### DMA controller

Five requests from the peripherals (TIMx, ADCx, SPIx, I2C1, USB and UARTx) are simply logically ORed before entering the DMA controller.

This means that only one request must be enabled at a time. Refer to the DMA request mapping below.

The peripheral DMA requests can be independently activated/de-activated by programming the DMA control bit in the registers of the corresponding peripheral.

Table 40. Summary of the DMA requests for each channel

Peripheral	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC	ADC1 <sup>(1)</sup>	ADC1 <sup>(2)</sup>			
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX
UART		UART1_TX <sup>(1)</sup>	UART1_RX <sup>(1)</sup>	UART1_TX <sup>(2)</sup> UART2_TX	UART1_RX <sup>(2)</sup> UART2_RX
I2C		I2C1_TX	I2C1_RX		
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP TIM1_CH3 TIM1_CH5
TIM2	TIM2_CH3	TIM2_UP	TIM2_CH2	TIM2_CH4	TIM2_CH1
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP	TIM3_CH1 TIM3_TRIG	

Peripheral	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
TIM16			TIM16_CH1 <sup>(1)</sup> TIM16_UP <sup>(1)</sup>	TIM16_CH1 <sup>(2)</sup> TIM16_UP <sup>(2)</sup>	
TIM17	TIM17_CH1 <sup>(1)</sup> TIM17_UP <sup>(1)</sup>	TIM17_CH1 <sup>(2)</sup> TIM17_UP <sup>(2)</sup>			
USB				USB	USB
CSM		CSM_CH1 <sup>(1)</sup>	CSM_CH2 <sup>(1)</sup>	CSM_CH1 <sup>(2)</sup>	CSM_CH2 <sup>(2)</sup>

1. DMA request mapped on this DMA channel only if the corresponding remapping bit is cleared in the SYSCFG\_CFGR register.
2. DMA request mapped on this DMA channel only if the corresponding remapping bit is set in the SYSCFG\_CFGR register.

## 12.4 DMA register description

Table 41. Overview of DMA registers

Offset	Acronym	Register Name	Reset	Section
0x00	DMA_ISR	DMA interrupt status register	0x00000000	Section 12.4.1
0x04	DMA_IFCR	DMA interrupt flag clear register	0x00000000	Section 12.4.2
0x08 + 20 × (n - 1)	DMA_CCRx	DMA channel x configuration register	0x00000000	Section 12.4.3
0x0C + 20 × (n - 1)	DMA_CNDTRx	DMA channel x number of data register	0x00000000	Section 12.4.4
0x10 + 20 × (n - 1)	DMA_CPARx	DMA channel x peripheral address register	0x00000000	Section 12.4.5
0x14 + 20 × (n - 1)	DMA_CMARx	DMA channel x memory address register	0x00000000	Section 12.4.6

### 12.4.1 DMA interrupt status register (DMA\_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												TEIF5	HTIF5	TCIF5	GIF5
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31:20	Reserved			Reserved, always read as 0.
19,15,11, 7, 3	TEIF <sub>x</sub>	r	0x00	<p>Channel x transfer error flag (<math>x = 1 \dots 5</math>)</p> <p>This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.</p> <p>0: No transfer error (TE) on channel x 1: A transfer error (TE) occurred on channel x</p>
18,14,10, 6, 2	HTIF <sub>x</sub>	r	0x00	<p>Channel x half transfer flag (<math>x = 1 \dots 5</math>)</p> <p>This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.</p> <p>0: No half transfer (HT) event on channel x 1: A half transfer (HT) event occurred on channel x</p>
17,13,9, 5, 1	TCIF <sub>x</sub>	r	0x00	<p>Channel x transfer complete flag (<math>x = 1 \dots 5</math>)</p> <p>This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.</p> <p>0: No transfer complete (TC) event on channel x 1: A transfer complete (TC) event occurred on channel x</p>
16,12,8, 4, 0	GIF <sub>x</sub>	r	0x00	<p>Channel x global interrupt flag (<math>x = 1 \dots 5</math>)</p> <p>This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.</p> <p>0: No TE, HT or TC event on channel x 1: A TE, HT or TC event occurred on channel x</p>

#### 12.4.2 DMA interrupt flag clear register (DMA\_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
												CTE IF5	CHT IF5	CTC IF5	CG IF5
												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTE IF4	CHT IF4	CTC IF4	CG IF4	CTE IF3	CHT IF3	CTC IF3	CG IF3	CTE IF2	CHT IF2	CTC IF2	CG IF2	CTE IF1	CHT IF1	CTC IF1	CG IF1

Bit	Field	Type	Reset	Description
31:20	Reserved			Reserved, always read as 0.
19,15,11, 7, 3	CTEIF <sub>x</sub>	w	0x00	Channel x transfer error clear (x = 1 ...5) These bits are set and cleared by software. 0: No effect 1: Clears the corresponding TEIF flag in the DMA_ISR register
18,14,10, 6, 2	CHTIF <sub>x</sub>	w	0x00	Channel x half transfer clear (x = 1 ...5) These bits are set and cleared by software. 0: No effect 1: Clears the corresponding HTIF flag in the DMA_ISR register
17,13,9, 5, 1	CTCIF <sub>x</sub>	w	0x00	Channel x transfer complete clear (x = 1 ...5) These bits are set and cleared by software. 0: No effect 1: Clears the corresponding TCIF flag in the DMA_ISR register
16,12,8, 4, 0	CGIF <sub>x</sub>	w	0x00	Channel x global interrupt clear (x = 1 ...5) These bits are set and cleared by software. 0: No effect 1: Clears the GIF, TEIF, HTIF and TCIF flags in the DMA_ISR register

### 12.4.3 DMA channel x configuration register (DMA\_CCR<sub>x</sub>) (x = 1...5)

Address offset: 0x08 + 20 × (channel number – 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARE	MEM2 MEM	PL	MSIZE	PSIZE	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:16	Reserved			Reserved, always read as 0.
15	ARE	rw	0x00	Auto reload This bit is set and cleared by software. 1: Auto reload number of data to transfer enabled 0: Auto reload number of data to transfer disabled

## Direct Memory Access Controller (DMA)

UM MM32F013x Ver1.00

14	MEM2MEM	rw	0x00	Memory to memory mode This bit is set and cleared by software. 0: Memory to memory mode disabled 1: Memory to memory mode enabled
13:12	PL	rw	0x00	Channel priority level These bits are set and cleared by software. 00: Low 01: Medium 10: High 11: Very high
11:10	MSIZE	rw	0x00	Memory size These bits are set and cleared by software. 00: 8 bits 01: 16 bits 10: 32 bits 11: reserved
9:8	PSIZE	rw	0x00	Peripheral size These bits are set and cleared by software. 00: 8 bits 01: 16 bits 10: 32 bits 11: reserved
7	MINC	rw	0x00	Memory increment mode This bit is set and cleared by software. 0: Memory increment mode disabled 1: Memory increment mode enabled
6	PINC	rw	0x00	Peripheral increment mode This bit is set and cleared by software. 0: Peripheral increment mode disabled 1: Peripheral increment mode enabled
5	CIRC	rw	0x00	Circular mode This bit is set and cleared by software. 0: Circular mode disabled 1: Circular mode enabled

Bit	Field	Type	Reset	Description
4	DIR	rw	0x00	Data transfer direction This bit is set and cleared by software. 0: Read from peripheral 1: Read from memory

3	TEIE	rw	0x00	Transfer error interrupt enable This bit is set and cleared by software. 0: TE interrupt disabled 1: TE interrupt enabled
2	HTIE	rw	0x00	Half transfer interrupt enable This bit is set and cleared by software. 0: HT interrupt disabled 1: HT interrupt enabled
1	TCIE	rw	0x00	Transfer complete interrupt enable This bit is set and cleared by software. 0: TC interrupt disabled 1: TC interrupt enabled
0	EN	rw	0x00	Channel enable This bit is set and cleared by software. 0: Channel disabled 1: Channel enabled

#### 12.4.4 DMA channel x number of data register (DMA\_CNDTR $x$ ) ( $x = 1 \dots 5$ )

Address offset: 0x0C + 20 × (channel number – 1)

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:16	Reserved			Reserved, always read as 0.

15:0	NDT	rw	0x0000	Number of data to transfer  Number of data to be transferred (0 up to 65535). This register can only be written when the channel is disabled (EN bit in the DMA_CCRx is 0). Once the channel is enabled, this register is read-only, indicating the remaining bytes to be transmitted. This register decrements after each DMA transfer. Once the transfer is completed, this register can either stay at zero or be reloaded automatically by the value previously programmed if the channel is configured in Auto Reload mode.  If this register is zero, no transaction can be served whether the channel is enabled or not.
------	-----	----	--------	---

#### 12.4.5 DMA channel x peripheral address register (DMA\_CPARx) (x = 1...5)

Address offset:  $0x10 + 20 \times (\text{channel number} - 1)$

Reset value: 0x0000 0000

This register must not be written when the channel is enabled (EN bit in the DMA\_CCRx is 1).

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

Bit	Field	Type	Reset	Description
31:0	PA	rw	0x0000 0000	Peripheral address  Base address of the peripheral data register from/to which the data will be read/written.  When PSIZE is 01 (16-bit), the PA[0] bit is ignored. Access is automatically aligned to a half-word address.  When PSIZE is 10 (32-bit), PA[1:0] are ignored. Access is automatically aligned to a word address.

#### 12.4.6 DMA channel x memory address register (DMA\_CMARx) (x = 1...5)

Address offset:  $0x14 + 20 \times (\text{channel number} - 1)$

Reset value: 0x0000 0000

This register must not be written when the channel is enabled (EN bit in the DMA\_CCRx is 1).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA															

Bit	Field	Type	Reset	Description
31:0	MA	rw	0x0000 0000	Memory address Base address of the memory area from/to which the data will be read/written. When MSIZE is 01 (16-bit), the MA[0] bit is ignored. Access is automatically aligned to a half-word address. When MSIZE is 10 (32-bit), MA[1:0] are ignored. Access is automatically aligned to a word address.

# 13 | Analog-to-digital converter (ADC)

## Analog-to-digital converter (ADC)

### 13.1 ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter (SAR A/D converter).

A/D converter supports multiple operation modes: single conversion, continuous conversion, or automatic scan with chosen direction through selected channel. The ADC converter can be started by software, external pin trigger, and individual timers.

The window comparator (analog watchdog) allows the application to detect if the input voltage goes outside the user-defined high or low thresholds.

The ADC input clock is generated from the PCLK2 clock divided by a prescaler and it must not exceed 15MHz.

### 13.2 ADC main features

- SAR ADC with up to 12-bit programmable resolution; no less than 10 multiplexed external input channels and 2 internal channels
- Conversion rate up to 1Msps
- Supports normal operation modes:
  - Single conversion mode: A/D converter does one conversion in the specified channel
  - One-cycle scan mode: A/D converter does one-cycle conversion in all specified channels (from low sequence number channel to high sequence number channel, or from high sequence number channel to low sequence number channel)
  - Continuous scan mode: A/D converter does one-cycle conversions continuously until the software stops A/D conversion. A/D conversion must stop if the conversion channel needs to be changed in the midway. The conversion can restart after configuring the corresponding register.
- Supports arbitrary channel operation modes:
  - Single conversion mode: A/D converter does one conversion in the specified channel
  - One-cycle scan mode: A/D converter does one-cycle conversion in all specified channels (in any sequence)
  - Continuous scan mode: A/D converter does one-cycle conversions continuously until the software stops A/D conversion. A/D conversion doesn't have to stop if the conversion channel needs to be changed. The conversion in a new channel will start in the next scan cycle after configuring the corresponding register.
- Supports the configuration of channel sampling time and resolution via software

- Supports DMA transfer
  - A/D start-of-conversion can be initiated:
    - By software
    - By external triggers with the configuration of delay via software
    - By Timer1/2/3 matching or TRGO signal, external EXTI signal source
- Analog watchdog for comparing the conversion result with the specified value; an interrupt generation can be set by the user when the conversion result matches with the specified value.

### 13.3 System block diagram

The ADC system block diagram is shown below:

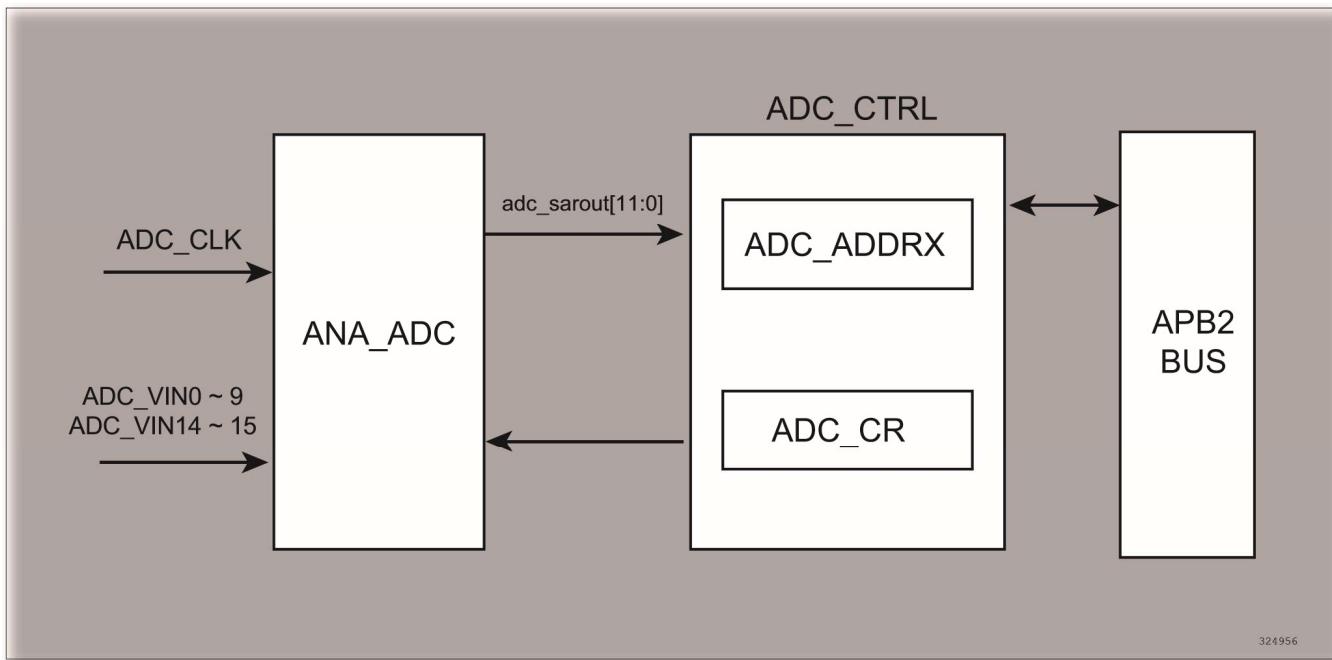


Figure 27. ADC system block diagram

### 13.4 ADC functional description

The figure below shows a single ADC block diagram.

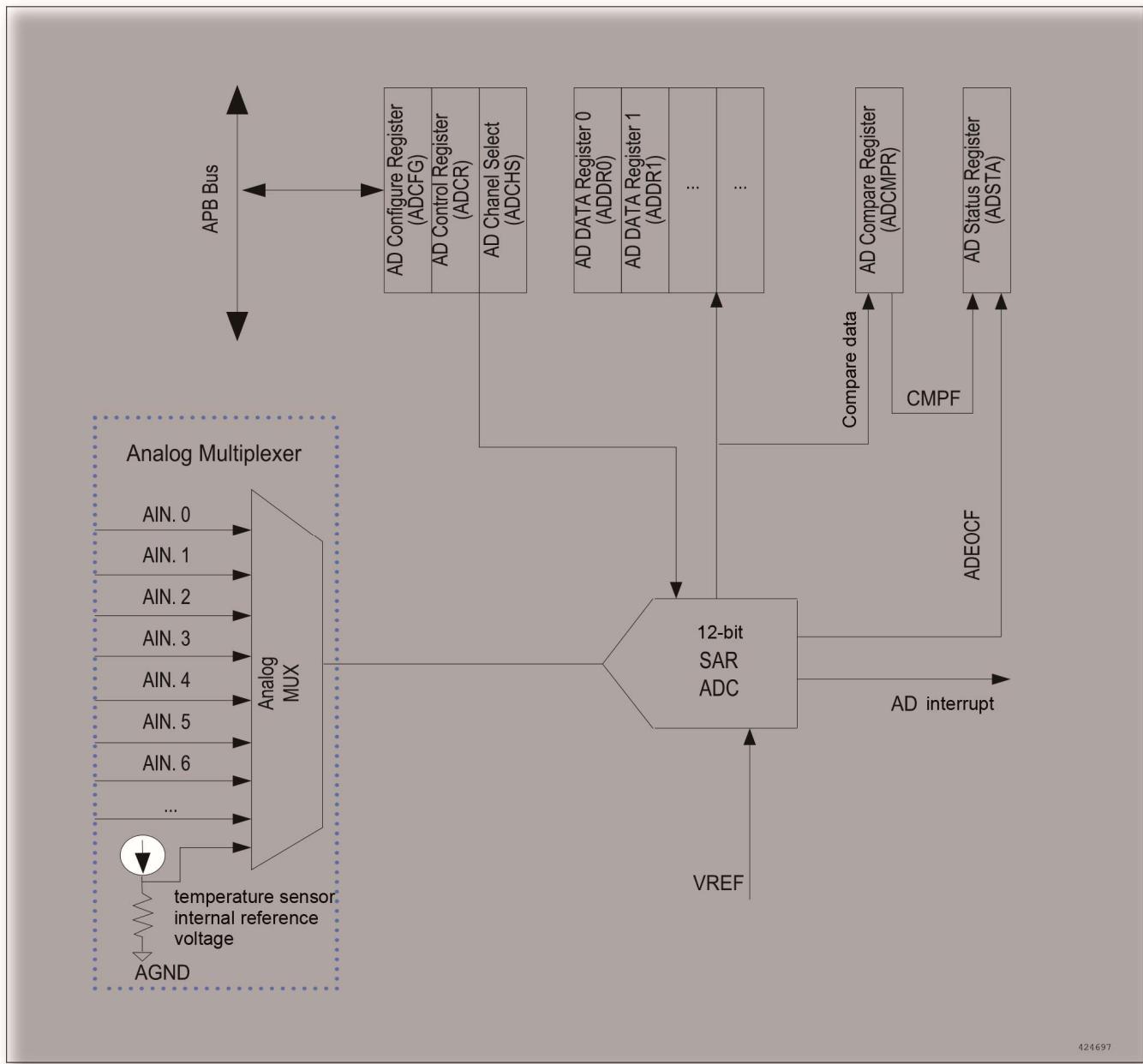


Figure 28. ADC block diagram

Note: T\_SENSOR (temperature sensor) is in the ADC AIN14 channel. V\_SENSOR (internal reference voltage) is in the ADC AIN15 channel.

### 13.4.1 ADC on-off control

The ADC can be powered-on by setting the ADON bit in the ADCFG register. When the ADEN bit is set for the first time, it wakes up the ADC from Power Down mode.

Conversion starts when ADST bit is set for the ADCR register after an ADC power-up delay has elapsed.

The conversion can be stopped by resetting the ADST bit, and the ADC put in power down mode by resetting the ADEN bit.

### 13.4.2 Channel selection

ADC1 has 10 multiplexed channels including external input channel, internal temperature sensor channel and internal 1.2V reference voltage channel. Each external input channel has independent enable bit which can be set by configuring the ADCHS register (in normal operation mode) or the CHANY\_NUM, CHANY\_SEL0 and CHANY\_SEL1 registers (in arbitrary channel operation mode).

## 13.5 Normal operation mode

### 13.5.1 Single conversion mode

In Single conversion mode the A/D converter does one conversion in the corresponding channel. The specific procedures are as follows:

- The A/D conversion is started by setting the ADST bit in the ADCR register via software, external trigger input, or timer overflow.
- After A/D conversion, the converted data is stored in the A/D data registers (ADDATA and ADDRn).
- After A/D conversion, the ADIF bit of the status register ADSTA is set to '1'. Meanwhile, an AD EOC (End Of Conversion) interrupt request is generated if the ADIE bit in the control register ADCR is set to '1'.
- ADST bit remains 1 during the A/D conversion period. Once the A/D channel conversion is completed, the ADST bit is automatically cleared and the A/D converter enters the idle mode.

Note: In Single conversion mode, the channel of lowest sequence is converted while others are ignored, if more than one channel is enabled via software.

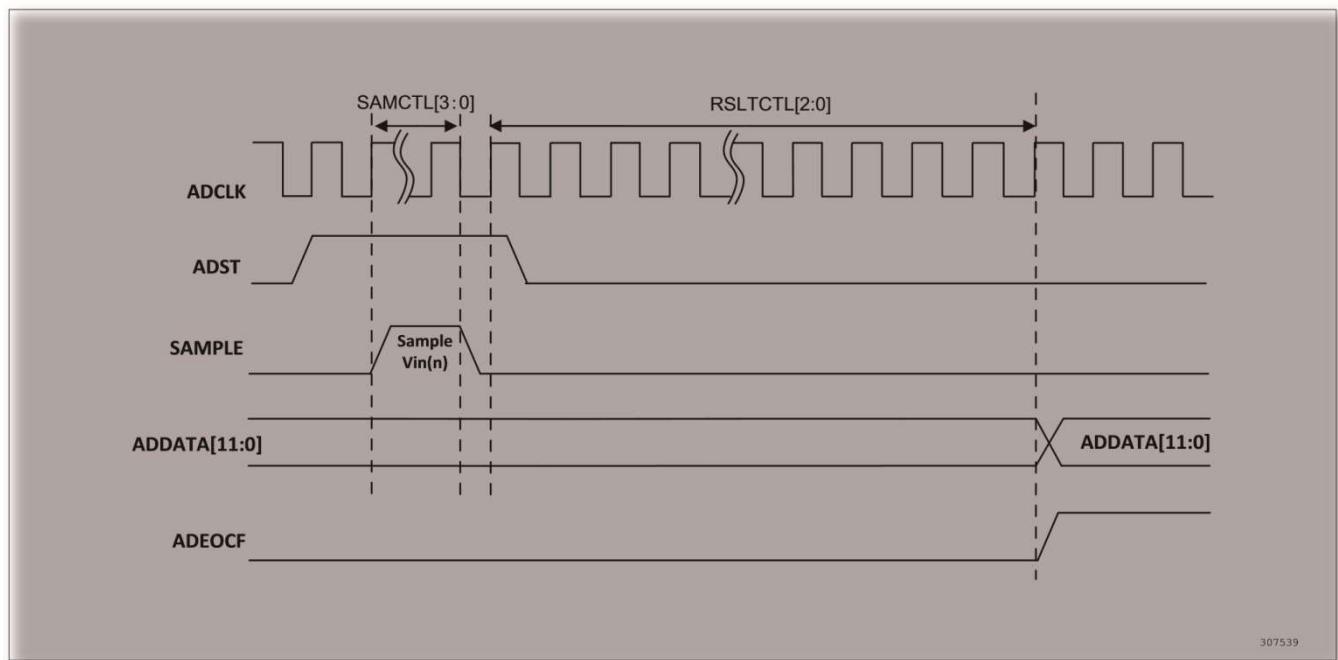


Figure 29. Single conversion mode timing diagram

### 13.5.2 One-cycle scan mode

In One-cycle scan mode, the A/D converter does one conversion for enabled channels in sequence (scan channel direction can be selected by setting the SCAN\_DIR in the register). The specific procedures are as follows:

- The conversion is started by setting the ADST bit by software or external trigger with the configuration of trigger delay via software. The default direction is from the lowest sequence number channel to the highest sequence number channel. It also can be programmed to be from the highest sequence number channel to the lowest sequence number channel.
- After the end of A/D conversion in each channel, the converted values will be loaded to the data register of corresponding channel in sequence. The ADIF EOC flag is set. At this time, an interrupt request is generated after all channels complete the conversion if the EOC interrupt is set.
- Once the final A/D channel sampling is completed, the ADST bit is automatically cleared and the A/D converter enters the idle mode.

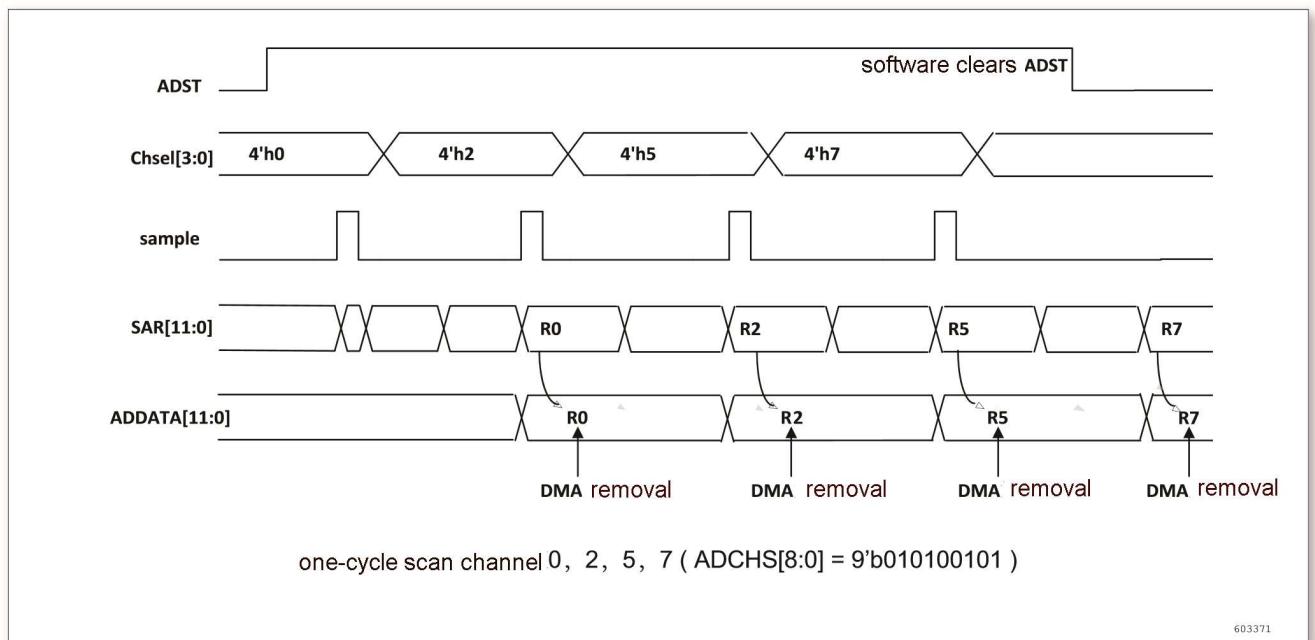


Figure 30. Enable channel conversion timing diagram in One-cycle scan mode (channel direction: from low to high)

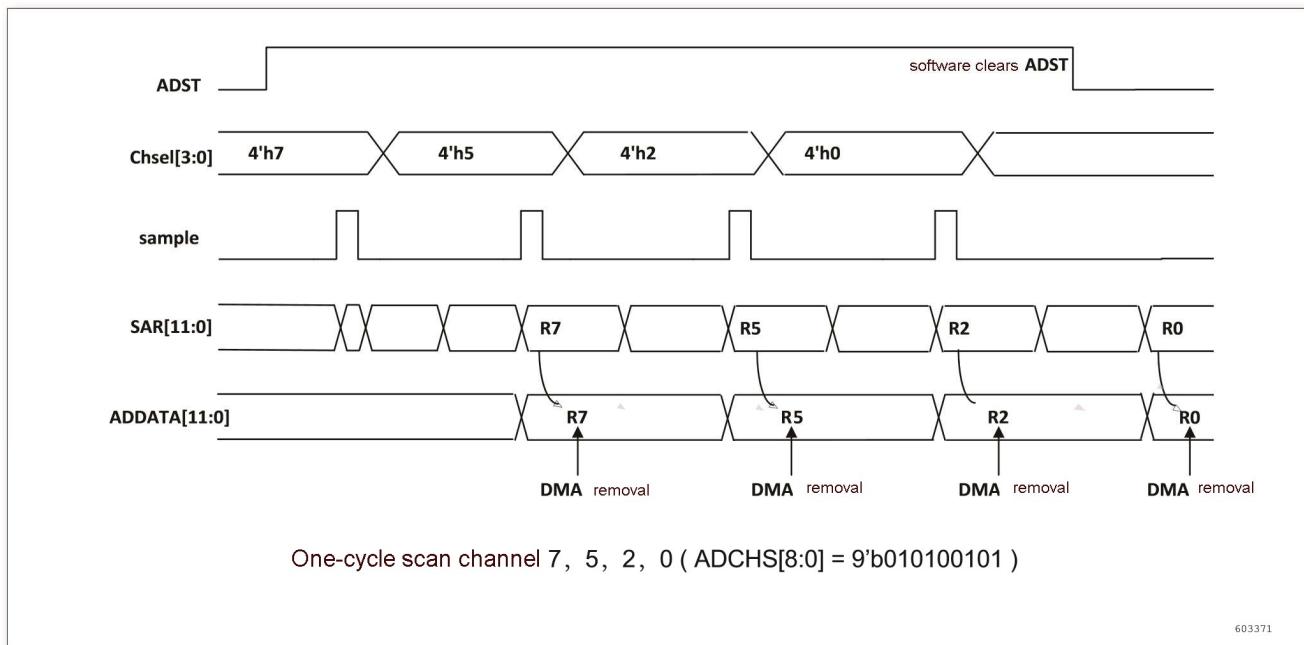


Figure 31. Enable channel conversion timing diagram in One-cycle scan mode (channel direction: from high to low)

### 13.5.3 Continuous scan mode

In continuous scan mode, the A/D converter does continuous conversions on the channel whose CHEN<sub>n</sub> bit is enabled in the ADCHS register (scan channel direction can be selected by setting the SCAN\_DIR in the register). The specific procedures are as follows:

- The conversion is started by setting the ADST bit by software or external trigger with the configuration of trigger delay via software. The default direction is from the lowest sequence number channel to the highest sequence number channel. It also can be programmed to be from the highest sequence number channel to the lowest sequence number channel.
  - After the end of A/D conversion in all channels, the converted values will be loaded to the corresponding data register in sequence. The ADIF EOC flag is set. At this time, an interrupt request is generated after all channels complete the conversion if the EOC interrupt is set.
- The A/D conversion continues as long as the ADST bit remains 1. Once the ADST bit is cleared and the current A/D conversion stops, the A/D converter enters the idle mode. The A/D converter also will complete the current conversion when the ADST bit is cleared.

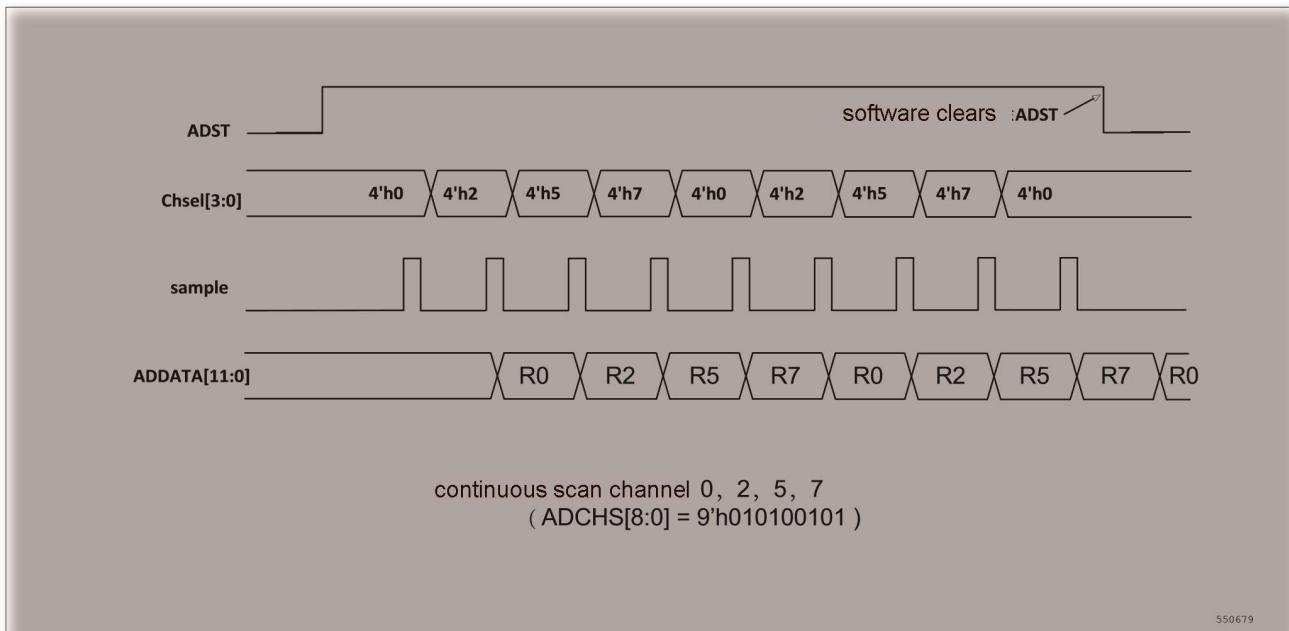


Figure 32. Enable channel conversion timing diagram in continuous scan mode (channel direction: from low to high)

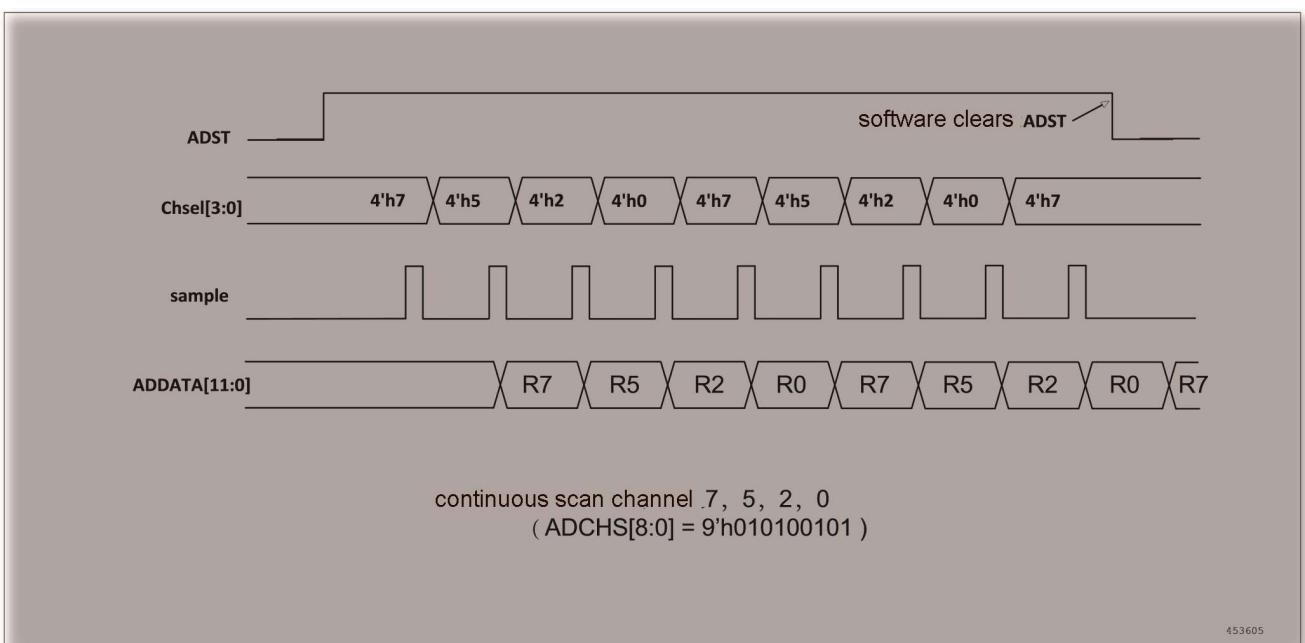


Figure 33. Enable channel conversion timing diagram in continuous scan mode (channel direction: from high to low)

## 13.6 Arbitrary channel operation mode

### 13.6.1 Single conversion mode

In Single conversion mode the A/D converter does one conversion in the corresponding channel. The specific procedures are as follows:

- Set by software the ADC\_ANY\_CFG, ADC\_CHANY0, and ADC\_CHANY1 registers, the conversion channel, and the CHANY\_MDEN bit. (only CHANY\_SEL0 has to be set for the single conversion mode)
- The A/D conversion is started by setting the ADST bit in the ADCR register via software, external trigger input, or timer overflow.
- After A/D conversion, the converted data is stored in the data registers (ADDATA and ADDRn).
- After A/D conversion, the ADIF bit of the status register ADSTA is set to '1'. Meanwhile, an AD EOC (End Of Conversion) interrupt request is generated if the ADIE bit in the control register ADCR is set to '1'.
- ADST bit remains 1 during the A/D conversion period. Once the A/D channel sampling is completed, the ADST is automatically cleared and the A/D converter enters the idle mode.
- If the software updates ADC\_ANY\_CFG, ADC\_CHANY0, and ADC\_CHANY1 during the A/D conversion period, the hardware will not update these settings immediately. The updates will be applied when the conversion of currently configured channels is completed, waiting for setting ADST by software next time.

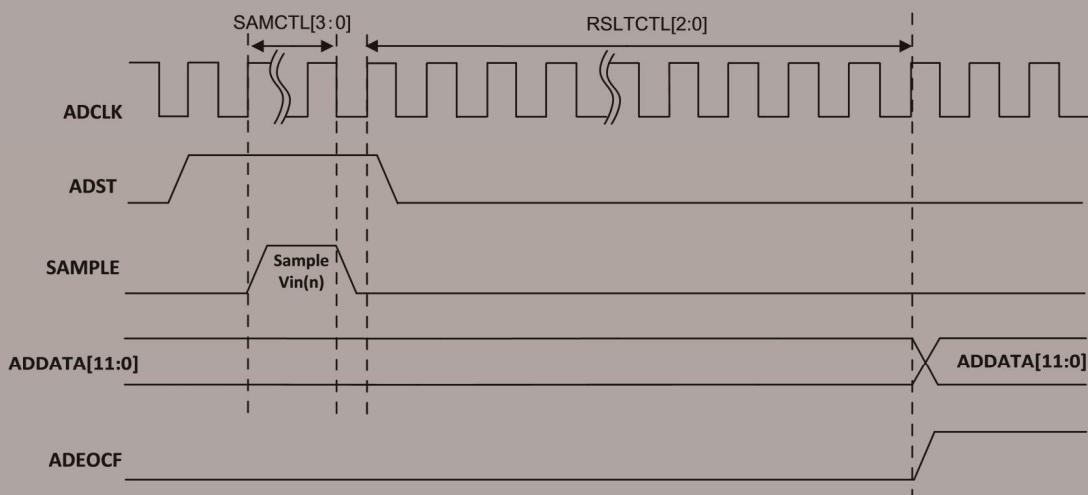


Figure 34. Channel conversion timing diagram in Single conversion mode

### 13.6.2 One-cycle scan mode

In One-cycle scan mode the A/D converter does one conversion according to the software configuration. The specific procedures are as follows:

- Set by software the ADC\_ANY\_CFG, ADC\_CHANY0, and ADC\_CHANY1 registers, desired channels and quantity, and the CHANY\_MDEN bit.
- Set the ADST bit in the ADCR register by software or external trigger with the configuration of trigger delay via software. A/D conversion direction is from CHANY\_SEL0 to CHANY\_SEL15. The number of conversion channels is configured by CHANY\_NUM. The

number is arbitrarily configured from CHANY\_SEL0 to CHANY\_SEL15, which can be identical or not.

- After the end of A/D conversion in each channel, the converted values will be loaded to the data register of corresponding channel in sequence. The ADIF EOC flag is set. At this time, an AD EOC interrupt request is generated if the ADIE bit in the control register ADCR is set to '1'.
- Once the final A/D channel sampling is completed, the ADST is automatically cleared and the A/D converter enters the idle mode.
- If the software updates ADC\_ANY\_CFG, ADC\_CHANY0, and ADC\_CHANY1 during the A/D conversion period, the hardware will not update these settings immediately. The updates will be applied when the conversion of currently configured channels is completed, waiting for setting ADST by software next time.

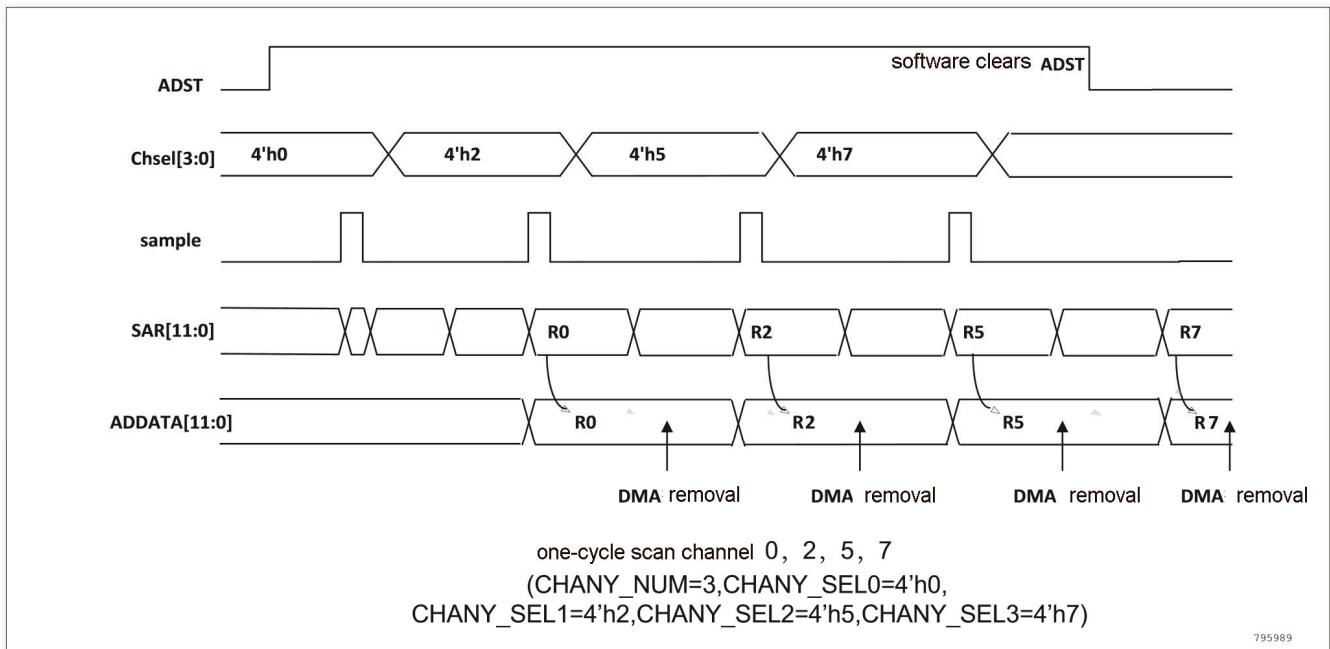


Figure 35. Channel conversion timing diagram In One-cycle scan mode

### 13.6.3 Continuous scan mode

In continuous scan mode the A/D converter does continuous conversions according to the software configuration until the software stops conversion. The specific procedures are as follows:

- Set by software the ADC\_ANY\_CFG, ADC\_CHANY0, and ADC\_CHANY1 registers, desired channels and quantity, and the CHANY\_MDEN bit.
- Set the ADST bit in the ADCR register by software or external trigger with the configuration of trigger delay via software. A/D conversion direction is from CHANY\_SEL0 to CHANY\_SEL15. The number of conversion channels is configured by CHANY\_NUM. The number is arbitrarily configured from CHANY\_SEL0 to CHANY\_SEL15, which can be identical or not.
- After the end of A/D conversion in each channel, the converted values will be loaded to the data register of corresponding channel in order. The ADIF EOC flag is set. At this time, an AD EOC interrupt request is generated if the ADIE bit in the control register ADCR is set to '1'.

- The A/D conversion continues as long as the ADST bit remains 1. Once the ADST bit is cleared and the current A/D conversion is completed, the A/D converter enters the idle mode.
- If the software updates ADC\_ANY\_CFG, ADC\_CHANY0, and ADC\_CHANY1 during the A/D conversion period, the hardware will not update these settings immediately. The updates will be applied when the conversion of current configured channels is completed, i.e. A conversion in the new channel starts in the next scan cycle.

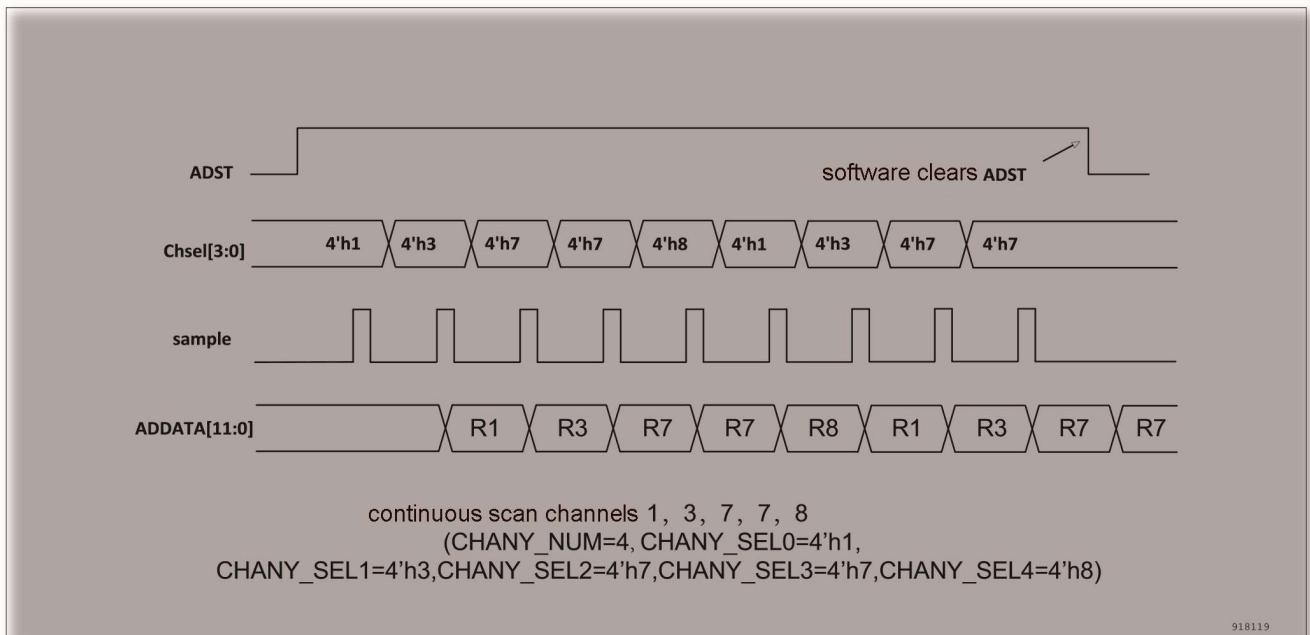


Figure 36. Channel conversion timing diagram in Continuous scan mode

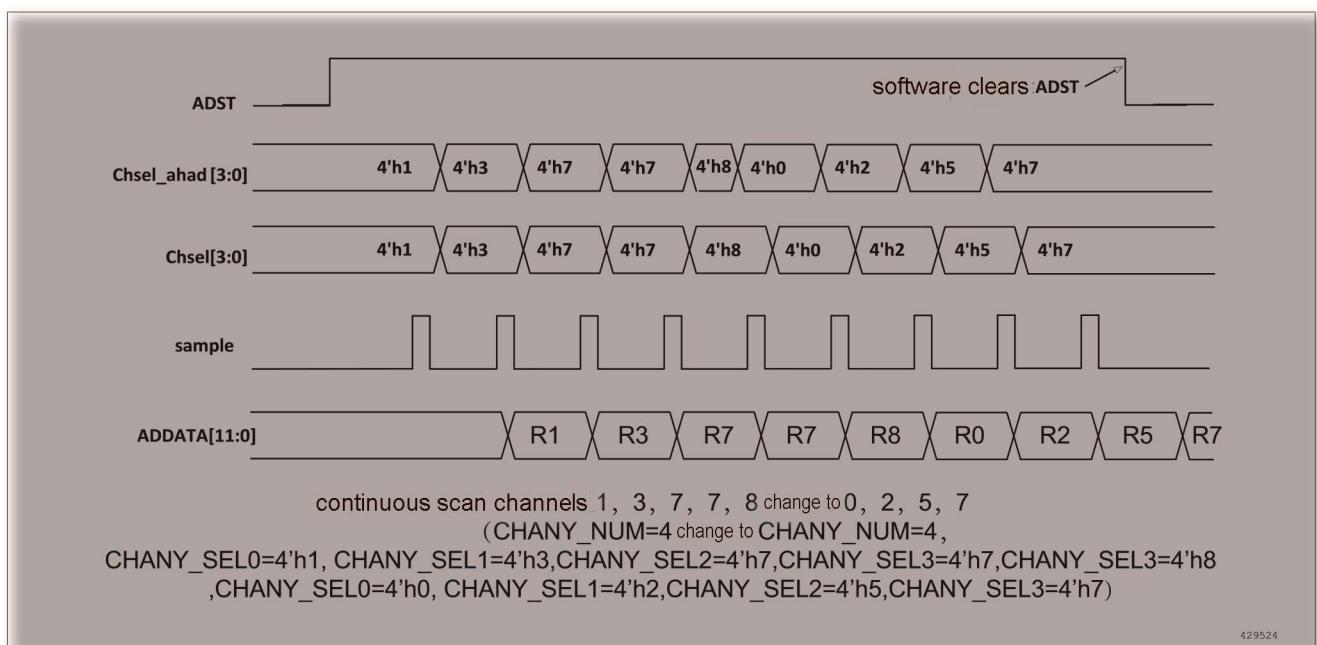


Figure 37. Timing diagram with dynamically updated configuration, in Continuous scan mode

### 13.6.4 DMA request

During one-cycle scan and continuous scan, the converted values are stored in the data register (ADDRn) of corresponding channel. The last conversion result is also stored in the ADDATA register. Transfer of data from one specific channels or from all scanned channels can be selected in the DMA transfer.

## 13.7 Data alignment

ALIGN bit in the ADCR register selects the alignment of data stored after conversion. Data can be left or right aligned as shown in figure below.

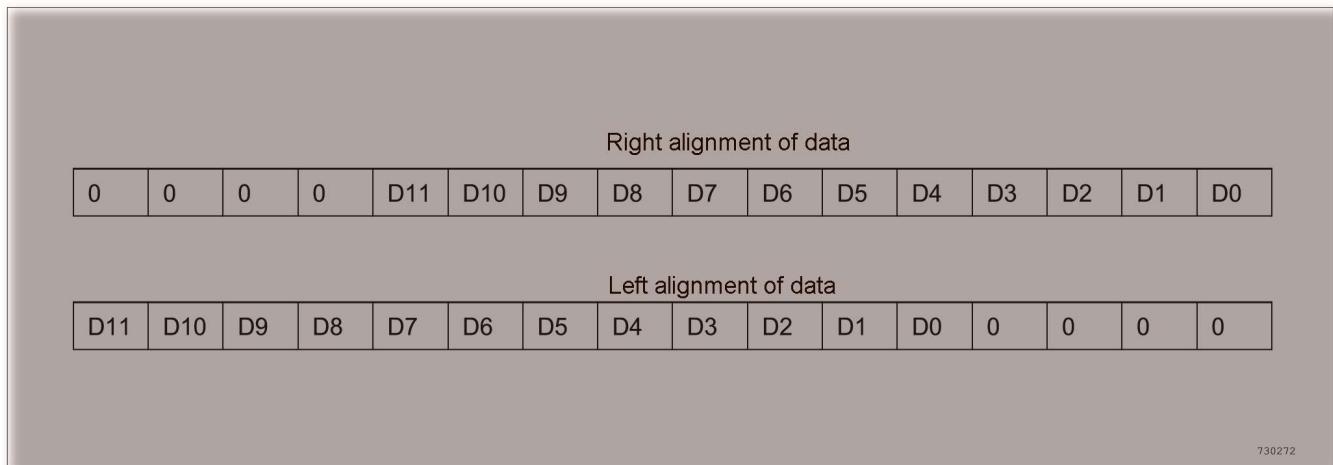


Figure 38. Data alignment style

### 13.7.1 Programmable resolution

The ADC conversion significance can be modified using the RSLTCTL[2:0] bits in the ADC\_CFG register to accelerate the data conversion speed. The significant data bits are aligned from the higher bits of 12-bit data.

### 13.7.2 Programmable sampling time

The ADC clock ADCLK is generated from the PCLK 2 divided by a prescaler. The prescale coefficient can be determined by setting the ADCPRE bit in the ADCFG register, i.e. PCLK 2/(N + 2) divided by a prescaler to be the ADC clock. ADC samples the input voltage for a number of ADC\_CLK cycles which can be modified using the SAMCTL[3:0] bits in the ADC\_CFG registers.

Set the ADC resolution as n bits (n=8,9,10,11,12). The sampling cycle of each channel is m.

The sampling frequency is calculated as follows:

$$F_{\text{sample}} = F_{\text{ADCLK}} / (m + n + 0.5)$$

Suppose the resolution is set to 12 bits, and the sampling cycle of each channel is 2.5T, then  $F_{\text{sample}} = F_{\text{ADCLK}}/15$ .

The total conversion time is calculated as follows:

$T_{CONV} = \text{Sampling time} + 12.5 \text{ conversion cycles}$

Example:

With an ADCCLK = 15MHz and a sampling time of 2.5 cycles:

$$T_{CONV} = 2.5 + 12.5 = 15 \text{ cycles} = 1\mu\text{s}$$

## 13.8 Conversion on external trigger

ADC conversion can be triggered by an external event (e.g. timer capture, EXTI line). If the TRGEN bit in the ADCR register is set then external events are able to trigger a conversion. An external trigger source can be selected by setting the TRGSEL bit. Refer to relevant bit description of AD control register (ADCR.TRGSEL) for details of external trigger source selection. Refer to relevant bit description of AD control register (ADCR.TRGSHIFT) for details of external trigger delay control. After a trigger signal is generated, the sampling starts after a delay of N PCLK2 clock cycles. In case of trigger scan mode, only the sampling in the first channel is delayed and the rest start immediately after the last sampling is completed.

## 13.9 Temperature sensor

The temperature sensor can be used to measure the ambient temperature ( $T_A$ ) of the device.

The temperature sensor is internally connected to the ADC internal signal source channel which is used to convert the sensor output voltage into a digital value. When not in use, this sensor can be turned off individually by setting a relevant register.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variations.

The internal temperature sensor is more suited to applications that detect temperature variations instead of absolute temperatures. If accurate temperature readings are needed, an external temperature sensor part should be used.

The temperature value can be calculated as follows:

$$T(^{\circ}\text{C}) = (V_{SENSE} - V_{25}) / \text{Avg\_Slope} + 25$$

$V_{25}$ : VSENSE value for 25°C

$V_{SENSE}$ : current output voltage of the temperature sensor

$$V_{SENSE} = \text{Value} * V_{dd}/4096 \text{ (Value is the converted result data of ADC)}$$

Avg\_Slope: Average Slope for curve between Temperature vs.  $V_{SENSE}$  (given in mV/°C or  $\mu\text{V}/^{\circ}\text{C}$ )

Refer to the Temperature Sensor section for the actual values of  $V_{25}$  and Avg\_Slope.

## 13.10 Internal reference voltage

The internal signal source channel of ADC is connected to an internal reference voltage of 1.2V. This channel converts the 1.2V reference voltage output to a digital value.

The internal reference voltage has an independent enable bit that can be enabled or disabled by setting the corresponding bit in the register.

## 13.11 Monitoring AD conversion result in the window comparator mode

An upper limit comparison register and a lower limit comparison register are provided in the comparison mode. The monitoring channel can be selected by setting the CMPCH bit via software.

Providing  $CPMHDATA \geq CPMLDATA$ , and the comparison result is greater than or equal to the specified CPMHDATA value or less than the specified CPMLDATA value in the ADCMPR register, the ADWIF bit in the status register ADSTA is set to 1.

Providing  $CPMHDATA < CPMLDATA$ , and the comparison result is equal to the specified CPMHDATA value or between two specified values, the ADWIF bit in the status register ADSTA is set to 1.

An interrupt request is generated if the ADWIE bit in the control register ADCR is set.

## 13.12 ADC register description

Table 42. Overview of ADC registers

Offset	Acronym	Register Name	Reset	Section
0x00	ADC_ADDATA	A/D data register	0x00000000	Section 13.12.1
0x04	ADC_ADCFG	A/D configuration register	0x00000000	Section 13.12.2
0x08	ADC_ADCR	A/D control register	0x00000000	Section 13.12.3
0x0C	ADC_ADCHS	A/D channel selection register	0x00000000	Section 13.12.4

Offset	Acronym	Register Name	Reset	Section
0x10	ADC_ADCMPR	A/D window compare register	0x00000000	Section 13.12.5
0x14	ADC_ADSTA	A/D status register	0x00000000	Section 13.12.6
0x18~0x3C	ADC_ADDR 0 ~ 9	A/D data register	0x00000000	Section 13.12.7
0x50~0x54	ADC_ADDR 14 ~ 15	A/D data register	0x00000000	Section 13.12.7
0x58	ADC_ADSTA_EXT	A/D extended state register	0x00000000	Section 13.12.8
0x5C	ADC_CHANY0	A/D arbitrary channel selection register 0	0x00000000	Section 13.12.9
0x60	ADC_CHANY1	A/D arbitrary channel selection register 1	0x00000000	Section 13.12.10
0x64	ADC_ANY_CFG	A/D arbitrary channel configuration register	0x00000000	Section 13.12.11
0x68	ADC_ANY_CR	A/D arbitrary channel control register	0x00000000	Section 13.12.12

### 13.12.1 A/D data register (ADC\_ADDATA)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										VAILD	OVER RUN	CHANNELSEL			
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31:22	Reserved			Reserved, always read as 0.
21	VALID	r	0x00	Valid flag (read-only) 1 = DATA[11:0] bits valid 0 = DATA[11:0] bits invalid This bit is set after the corresponding analog channel conversion is completed. This bit is cleared by hardware after reading the ADDATA register.
20	OVERRUN	r	0x00	Data overrun flag (read-only) 1 = DATA[11:0] data is overrun 0 = DATA[11:0] data is the last conversion result Before the new conversion result is loaded to the register, OVERRUN is set to 1 if the DATA[11:0] is not read. This bit is cleared by hardware after reading the ADDATA register.
Bit	Field	Type	Reset	Description

19:16	CHANNELSEL r	0x00	These 4 bits show the channel corresponding to the current data (Channel selection) 0000 = conversion data of channel 0 0001 = conversion data of channel 1 0010 = conversion data of channel 2 0011 = conversion data of channel 3 0100 = conversion data of channel 4 0101 = conversion data of channel 5 0110 = conversion data of channel 6 0111 = conversion data of channel 7 1000 = conversion data of channel 8 1001 = conversion data of channel 9 1110 = conversion data of temperature sensor 1111 = conversion data of internal reference voltage Others: invalid
15:0	DATA r	0x00	12-bit A/D conversion result (Transfer data) Data can be left or right aligned as configured.

### 13.12.2 A/D configuration register (ADC\_ADCFG)

Address offset: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	ADCPRE		SAMCTL		RSLTCTL		ADCPRE		VSEN	TSEN	ADWEN	ADEN				

Bit	Field	Type	Reset	Description
31:15	Reserved			Reserved, always read as 0.
14	ADCPRE	rw	0x00	ADC prescaler As the least significant bit of ADCPRE[3:0], used together with Bit[6:4]
13:10	SAMCTL	rw	0x00	Channel x Sample time selection These bits are used to independently select the sample time of each channel. In a sample cycle, the channel selection bit should remain unchanged. 0000: 2.5 cycles      0100: 42.5 cycles 0001: 8.5 cycles      0101: 56.5 cycles 0010: 14.5 cycles      0110: 72.5 cycles

Bit	Field	Type	Reset	Description

	0011: 29.5 cycles	0111: 240.5 cycles
	1000: 3.5 cycles	1001: 4.5 cycles
	1010: 5.5 cycles	1011: 6.5 cycles
	1100: 7.5 cycles	Others: reserved

9:7	RSLTCTL	rw	0x00	ADCx conversion data resolution selection 000: 12 significant bits 010: 10 significant bits 100: 8 significant bits	001: 11 significant bits 011: 9 significant bits
6:4	ADCPRE	rw	0x00	ADC prescaler Set or cleared by software to determine the ADC clock frequency. PCLK2 divided by (([6:4],[14]) + 2) as the ADC clock	
3	VSEN	rw	0x00	Internal reference voltage enable (Voltage Sensor enable) 1: Internal voltage sensor enabled 0: Internal voltage sensor disabled	
2	TSEN	rw	0x00	Temperature sensor enable 1 = Temperature sensor enabled 0 = Temperature sensor disabled	
1	ADWEN	rw	0x00	A/D window comparator enable (ADC window comparison enable) 1= A/D window comparator enabled 0= A/D window comparator disabled	
0	ADEN	rw	0x00	A/D conversion enable (ADC enable) 1 = enabled 0 = disabled	

### 13.12.3 A/D control register (ADC\_ADCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	2	27	26	25	24	23	2:	21	20	19	18	17	16
Reserved				TRG_EDGE		Reserved		TRGSHIFT			TRGSEL		SCANDIR		
15	14	13	1	11	10	9	8	7	6	5	4	3	2	1	0
rw	rw	rw	rw	rw	Rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
CMPCH	ALIGN	ADMD	ADST	Res.	TRGSEL			DMAEN	TRGEN	ADWIE	ADIE				

Bit	Field	Type	Rese t	Description
31:26	Reserved			Reserved, always read as 0.

25:24	TRG_EDGE	rw	0x00	Trigger edge selection 00: Double-edge trigger 01: Falling trigger register 10: Rising edge trigger 11: Trigger blocked
23:22	Reserved			Reserved and always read as 0.
21:19	TRGSHIFT	rw	0x00	External trigger shift sample After a trigger signal is generated, the sampling starts after a delay of N PCLK2 clock cycles. In case of trigger scan mode, the sampling in the rest channels starts immediately after the last sampling is completed. 0: no delay                  1: 4 cycles 2: 16 cycles                3: 32 cycles 4: 64 cycles                5: 128 cycles 6: 256 cycles              7: 512 cycles
18:17	TRGSEL	rw	0x00	External trigger selection Used together with Bit[6:4]
16	SCANDIR	rw	0x00	ADC scan sequence direction This bit is used to select the direction in which the channels will be scanned in the one-cycle scan mode or continuous scan mode. 0: Upward scan from low sequence channel to high sequence channel by the ADC channel selection register 1: Backward scan from high sequence channel to low sequence channel by the ADC channel selection register
15:12	CMPCH	rw	0x00	Window comparison channel selection 0000 = conversion result for selecting comparison channel 0 0001 = conversion result for selecting comparison channel 1 0010 = conversion result for selecting comparison channel 2 0011 = conversion result for selecting comparison channel 3 0100 = conversion result for selecting comparison channel 4 0101 = conversion result for selecting comparison channel 5 0110 = conversion result for selecting comparison channel 6 0111 = conversion result for selecting comparison channel 7 1000 = conversion result for selecting comparison channel 8 1001 = conversion result for selecting comparison channel 9 1110 = conversion result for selecting comparison temperature sensor 1111 = all the other scan channels Others: invalid

Bit	Field	Type	Reset	Description

Bit	Field	Type	Reset	Description
11	ALIGN	rw	0x00	<p>Data alignment 0: right aligned 1: left aligned</p>
10:9	ADMD	rw	0x00	<p>A/D conversion mode (ADC mode) 00: single conversion 01: one-cycle conversion 10: continuous scan  When the conversion mode is changed, the software has to disable the ADST bit.</p>
8	ADST	rw	0x00	<p>A/D conversion start 1 = conversion start 0 = end of conversion or entering the idle mode  There are two ways of setting ADST: In the single or one-cycle mode, ADST is automatically cleared by hardware after conversion. In continuous scan mode the A/D converter does continuous conversions until the software writes '0' to the bit or the system is reset.</p>
7	Reserved			Reserved and always read as 0.
6:4	TRGSEL	rw	0x00	<p>External trigger selection, bits [18:17,6:4] Select external trigger source 00000: TIM1_CC1 00001: TIM1_CC2 00010: TIM1_CC3 00011: TIM2_CC2 00100: TIM3_TRGO 00110: TIM3_CC1 00111: EXTI line 11 01000: TIM1_TRGO 01011: TIM2_CC1 01100: TIM3_CC4 01101: TIM2_TRGO 01111: EXTI line 15 10000: TIM1_CC4 10001: TIM1_CC5 Others: invalid</p>
3	DMAEN	rw	0x00	<p>Direct memory access enable 1 = DMA request enabled 0= DMA request disabled</p>

2	TRGEN	rw	0x00	External hardware trigger source (External trigger enable) 1 = enable A/D conversion by external trigger signal 0 = not enable A/D conversion by external trigger signal
1	ADWIE	rw	0x00	ADC window comparator interrupt enable 1= A/D window comparator interrupt enabled 0= A/D window comparator interrupt disabled
0	ADIE	rw	0x00	ADC interrupt enable 1 = ADC interrupt enabled 0 = ADC interrupt disabled An A/D EOC interrupt request is generated if the ADIF bit is set.

### 13.12.4 A/D channel selection register (ADC\_ADCHS)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHENVS	CHENTS	Reserved		CHEN9	CHEN8	CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHENO		
rw	rw			rw	rw	rw									

Bit	Field	Type	Reset	Description
31:16	Reserved			Reserved, always read as 0.
15	CHENVS	rw	0x00	Internal reference voltage enable (Voltage Sensor enable) 1 = enabled 0 = disabled
14	CHENTS	rw	0x00	Temperature sensor enable 1 = enabled 0 = disabled
13:10	Reserved			Reserved and always read as 0.
9	CHEN9	rw	0x00	Analog input channel 9 enable 1 = enabled 0 = disabled
8	CHEN8	rw	0x00	Analog input channel 8 enable 1 = enabled 0 = disabled

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

7	CHEN7	rw	0x00	Analog input channel 7 enable 1 = enabled 0 = disabled
6	CHEN6	rw	0x00	Analog input channel 6 enable 1 = enabled 0 = disabled
5	CHEN5	rw	0x00	Analog input channel 5 enable 1 = enabled 0 = disabled
4	CHEN4	rw	0x00	Analog input channel 4 enable 1 = enabled 0 = disabled
3	CHEN3	rw	0x00	Analog input channel 3 enable 1 = enabled 0 = disabled
2	CHEN2	rw	0x00	Analog input channel 2 enable 1 = enabled 0 = disabled
1	CHEN1	rw	0x00	Analog input channel 1 enable 1 = enabled 0 = disabled
0	CHEN0	rw	0x00	Analog input channel 0 enable 1 = enabled 0 = disabled

Note: If all channels are reset, then the channel 0 is enabled.

### 13.12.5 A/D window compare register (ADC\_ADCMPR)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:28	Reserved			Reserved, always read as 0.

27:16	CMPHDATA	rw	0x00	Compare data high limit The 12-bit value will be compared with the conversion result from a specified channel.
15:12	Reserved			Reserved and always read as 0.
11:0	CMPLDATA	rw	0x00	Compare data low limit The 12-bit value will be compared with the conversion result from a specified channel.

### 13.12.6 A/D status register (ADC\_ADSTA)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved							OVERRUN								Reserved	VALID
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
						VALID			CHANNEL		Res.	BUSY	ADWIF	ADIF		
	r	r	r	r	r	r	r	r	r	r	r	r	r	rc_w1	rc_w1	

Bit	Field	Type	Reset	Description
31:30	Reserved			Reserved, always read as 0.
29:20	OVERRUN	r	0x00	Data overrun flag for channel 0 to channel 9 (Overrun flag) Read-only.
19:18	Reserved			Reserved and always read as 0.
17:8	VALID	r	0x00	Valid flag for channel 0 to channel 9 (Valid flag) Read-only.
7:4	CHANNEL	r	0x00	Current conversion channel These 4 bits indicate the channel in conversion when BUSY = 1. These 4 bits indicate the channel for the next conversion when BUSY = 0.
3	Reserved			Reserved and always read as 0.
2	BUSY	r	0x00	Busy/Idle 1 = A/D converter is busy 0 = A/D converter is idle

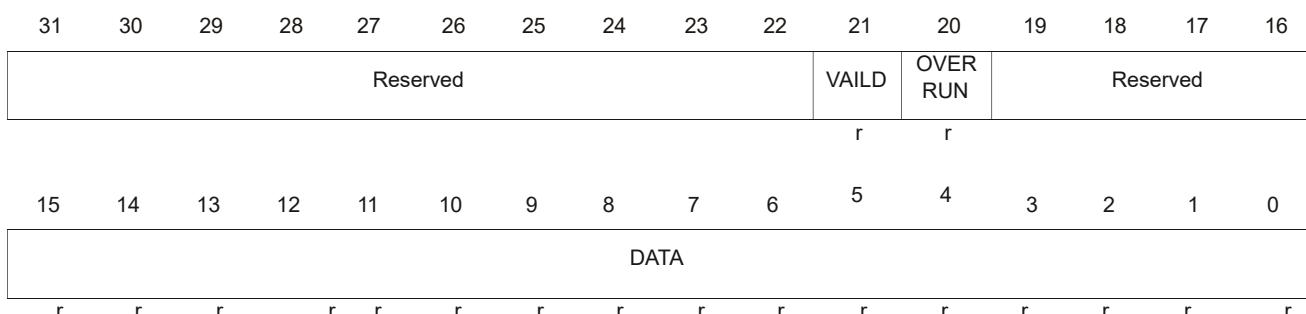
Bit	Field	Type	Reset	Description

1	ADWIF	rc_w1	0x00	ADC window comparator interrupt flag Providing CPMHDATA $\geq$ CPMLDATA, and the selected ADC channel comparison result is greater than or equal to the specified CPMHDATA value or less than the specified CPMLDATA value in the ADCMPR register, the ADWIF bit in the status register ADSTA is set to 1. Providing CPMHDATA $<$ CPMLDATA, and the selected ADC channel comparison result is equal to the specified CPMHDATA value or between two specified values, the ADWIF bit in the status register ADSTA is set to 1. This flag bit is cleared by writing '1'.
0	ADIF	rc_w1	0x00	ADC convert complete flag This bit is set by hardware at the end of channel group conversion, and cleared by software. 1 = A/D conversion complete 0 = A/D converter incomplete This flag bit is cleared by writing '1'.

### 13.12.7 A/D data register (ADC\_ADDR0 ~ 914 ~ 15)

Address offset: 0x18 – 0x3C, 0x50 – 0x54

Reset value: 0x0000 0000



Bit	Field	Type	Reset	Description
31:22	Reserved			Reserved, always read as 0.
21	VALID	r	0x00	Valid flag (read-only) 1 = DATA[11:0] bit valid 0 = DATA[11:0] bit invalid  This bit is set after the corresponding analog channel conversion is completed. This bit is cleared by hardware after reading the ADDATA register.
20	OVERRUN	r	0x00	Data overrun flag (read-only) 1 = DATA [11:0] data is overrun 0 = DATA [11:0] data is the last conversion result Before the new conversion result is loaded to the register, OVERRUN is set to '1' if the DATA[11:0] is not read. This bit is cleared by hardware after reading the ADDATA register.
19:16	Reserved			Reserved and always read as 0.
Bit	Field	Type	Reset	Description

15:0	DATA	r	0x00	12-bit A/D conversion result in the channel (Transfer data) Data can be left or right aligned as configured.
------	------	---	------	---

**13.12.8 A/D extended state register (ADC\_ADSTA\_EXT)**

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OVERRUN	r	r	r	r	r	r	r
Bit	Field	Type	Reset	Description											
31:8	Reserved			Reserved, always read as 0.											
7:6	OVERRUN	r	0x00	Data overrun flag for the channel 10: channel 15 (V_SENSOR) 01: channel 14 (T_SENSOR)											
5:4	Reserved			Reserved and always read as 0.											
3:2	VALID	r	0x00	Valid flag for the channel 10: channel 15 (V_SENSOR) 01: channel 14 (T_SENSOR)											
1:0	Reserved			Reserved and always read as 0.											

**13.12.9 A/D arbitrary channel selection register 0 (ADC\_CHANY0)**

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHANY_SEL7				CHANY_SEL6				CHANY_SEL5				CHANY_SEL4			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANY_SEL3				CHANY_SEL2				CHANY_SEL1				CHANY_SEL0			
rw	rw	rw	rw												

Bit	Field	Type	Reset	Description											
31:28	CHANY_SEL7	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.											

## Analog-to-digital converter (ADC)

UM MM32F013x Ver1.00

27:24	CHANY_SEL6	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.
23:20	CHANY_SEL5	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.
19:16	CHANY_SEL4	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.
15:12	CHANY_SEL3	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.
11:8	CHANY_SEL2	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.
7:4	CHANY_SEL1	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.
3:0	CHANY_SEL0	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.

Note: In the one-cycle or continuous scan modes, the ADC\_CHANY0 shadow register is started by hardware. Before ADC starts to work, writing to ADC\_CHANY0 by software also writes to its shadow register. When ADC is working, only the shadow register is updated if the ADC\_CHANY0 value is changed. Besides that, when ADC starts to convert the last channel, the value of the shadow register will update to ADC\_CHANY1, so as to complete the dynamic channel switch.

### 13.12.10 A/D arbitrary channel selection register 1 (ADC\_CHANY1)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		CHANY_SEL15		CHANY_SEL14				Reserved							
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CHANY_SEL9				CHANY_SEL8			
								rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description											
31:28	CHANY_SEL15	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.											
27:24	CHANY_SEL14	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.											
23:8	Reserved			Reserved and always read as 0.											
7:4	CHANY_SEL9	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.											
3:0	CHANY_SEL8	rw	0x00	can be configured to any channel from channel 0 to channel 9 and from channel 14 to channel 15.											

Note: In the one-cycle or continuous scan modes, the ADC\_CHANY1 shadow register is started by hardware. Before ADC starts to work, writing to ADC\_CHANY1 by software also writes to its shadow register. When ADC is working, only the shadow register is updated if the ADC\_CHANY1 value is changed. Besides that, when ADC starts to convert the last channel, the value of the shadow register will update to ADC\_CHANY1, so as to complete the dynamic channel switch.

### 3.12.11 A/D arbitrary channel configuration register (ADC\_ANY\_CFG)

Address offset: 0x64

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CHANY_NUM			
												rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:4	Reserved			Reserved, always read as 0.
3:0	CHANY_NUM	rw	0x00	Number of channels: 0: 0 channel 1: 0 ~ 1 channels 2: 0 ~ 2 channels ... 14: 0 ~ 14 channels 15: 0 ~ 15 channels

Note: In the one-cycle or continuous scan modes, the ADC\_NUM shadow register is started by hardware. Before ADC starts to work, writing to ADC\_NUM by software also writes to its shadow register. When ADC is working, only the shadow register is updated if the ADC\_NUM value is changed. Besides that, when ADC starts to convert the last channel, the value of the shadow register will update to ADC\_NUM, so as to complete the dynamic channel switch.

### 13.12.12 A/D arbitrary channel control register (ADC\_ANY\_CR)

Address offset: 0x68

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CHANY_MDEN			
												rw			

Bit	Field	Type	Reset	Description
31:1	Reserved			Reserved, always read as 0.

0	CHANY_MDEN	rw	0x00	Arbitrary channel configuration mode enable bit: 1: enabled 0: disabled  When this bit is enabled, the function of ADC channel configuration become changed. Originally, the channels were only controlled by the CHENx in the register ADC_ADCHS. Currently, they are controlled by two parts together. CHANY_NUM is used to configure the number of channels from channel 0 to channel 15. CHANY_SEL0 ~ CHANY_SEL15 are used to configured them to an arbitrary ADC channel.
---	------------	----	------	--

Note: In the arbitrary mode plus the one-cycle/continuous scan mode, the ADST bit in the ADC\_ADCR must be disabled before closing the ADC. Then the user should judge whether the BUSY bit in the ADC\_ADSTA is 0. That is to say, the CHANY\_MDEN bit in the ADC\_ANY\_CR should be disabled when the ADC conversion is completed.

# 14

# Comparator (COMP)

## Comparator (COMP)

### 14.1 COMP introduction

The chip embeds two general-purpose comparators COMP1 and COMP2, that can be used either as standalone devices (all terminal are available on I/Os) or with the timers. The comparators can be used for a variety of functions including:

- Wake-up event from low-power mode triggered by an analog signal
- Analog signal conditioning
- Cycle-by-cycle current control loop when combined with a PWM output from a timer

### 14.2 COMP main features

- Rail-to-rail comparators
- Each comparator has an optional threshold
  - Alternate I/O pins
  - The internal comparison voltage CRV can be AVDD or the partial voltage value of the internal reference voltage
- Programmable hysteresis voltage
- Programmable rate and power consumption
- Comparison result filtering is supported
- The output terminal can be redirected to an I/O port or multiple timer input terminals to trigger the following events:
  - Capture event
  - OCref\_clr event (cycle-by-cycle current control)
  - Brake event for fast PWM shutdowns
- Two comparators can be combined and used in a window comparator.
- Each comparator has interrupt generation capability and supports wake-up from Sleep and Stop modes (through the EXTI controller)
- COMP1/2 has 4 normal phase inputs and 4 inverting inputs with polling function.
  - Supports the polling switch in fixed cycles
  - Supports the control of polling channel 1/2/3 or 1/2
  - Supports fixed inverting input terminal

### 14.3 COMP functional description

#### 14.3.1 Introduction

The block diagram of the comparators is shown below:

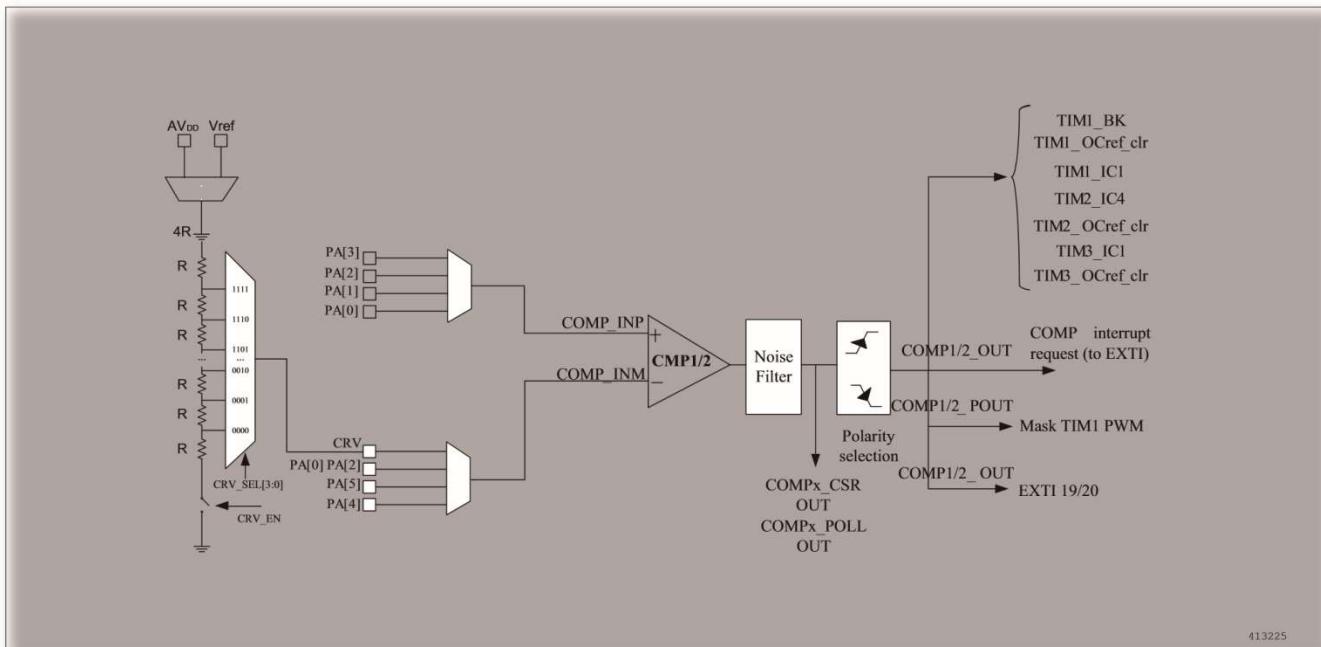


Figure 39. Comparator block diagrams

### 14.3.2 Clock

The COMP clock provided by the clock controller is synchronous with the PCLK (APB2 clock). Clock enable bit in the RCC controller should be set before using the comparator.

### 14.3.3 COMP switch

The COMP is powered-on by setting the EN bit in the COMPx\_CSR register. The COMP is waken up from the power-off state when EN bit is set and stops operation when EN bit is reset.

### 14.3.4 COMP inputs and outputs

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The output can also be internally redirected to a variety of timer inputs for the following purposes:

- BKIN for emergency shut-down of PWM signals
- OCref\_clr inputs for cycle-by-cycle current control
- Input capture for timing measures

### 14.3.5 COMP channel selection

COMP has 4 normal phase input channels to be selected from four external pins alongside 4 inverting input channels to be selected from three internal pins or the partial voltage value of CRV voltage. The CRV voltage can be AVDD or the partial voltage value of the internal reference 1.2V.

In normal operation mode, the input channel of COMP is selected by software. In polling operation mode, comparison results of multiple channels are monitored in a time-shared manner via software polling. Logically, it is similar to simultaneous operation of several

comparators. In normal operation mode, the COMP compares selected signals from INP and INM ports. The specific procedures are as follows:

- Set the INP\_SEL bit and the INM\_SEL bit in the COMPx\_CSR register to select desired signals;
- Set the EN bit in the COMPx\_CSR register, the COMP is powered-on and starts to work;
- The comparison results are stored in the OUT bit of the COMPx\_CSR register.

Moreover, it is required to set the CRV\_SEL bit in the COMP\_CRV register and then set the CRV\_EN bit when the INM\_SEL of COMP selects CRV.

In polling operation mode, the signals from the INP port in the COMP polls periodically. The FIXN bit in the COMPx\_POLL register can be used to configure whether the signals from INM port follow the INP changes. The INM\_SEL bit in the COMPx\_CSR can also be used to configure them. Please note that the INP\_SEL bit in the COMPx\_CSR will be invalid when the polling function is enabled. Similarly, if the FIXN bit in the COMPx\_POLL register determines the INM port follow the INP polling changes, then the INN\_SEL bit in the COMPx\_CSR will be invalid too. The specific procedures are as follows:

- Set the PERIOD bit in the COMPx\_POLL register to select the desired polling wait cycles.
- Set the FIXN bit in the COMPx\_POLL register to determine whether the signals from INM port follow the INP polling changes.
- Set the POLL\_CH bit in the COMPx\_POLL register to determine whether the desired polling channel is 1/2/3 or 1/2.
- Set the POLL\_EN bit in the COMPx\_POLL register to enable the polling function.
- Set the EN bit in the COMPx\_CSR register, the COMP is powered-on and starts to work.
- The polling comparison results are stored in the POUT bit of the COMPx\_POLL register, where the results of polling channel 3/2/1 are individually stored in the POUT[2], POUT[1], and POUT[0].

#### 14.3.6 Interrupt and wakeup

The comparator outputs are internally connected to the external interrupts and events controller. Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from low-power modes.

Refer to Interrupt and events section in the reference manual for more details.

#### 14.3.7 Power mode

The comparator power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The bit MODE in COMPx\_CSR register can be programmed as follows:

- 00: High speed/full power
- 01: Medium speed/medium power
- 10: Low speed/low-power
- 11: Very-low speed/ultra-low-power

#### 14.3.8 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary

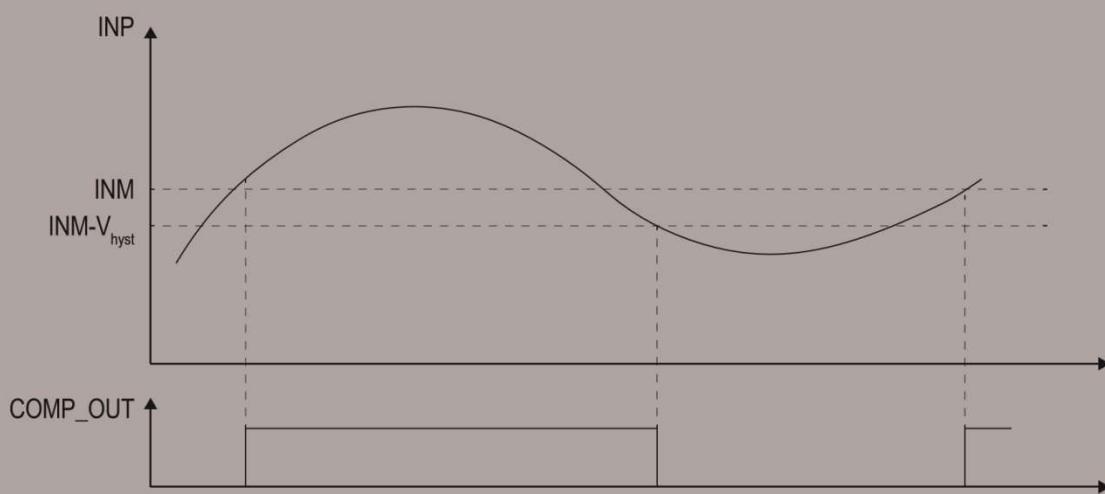
to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

Once the programming is completed, the LOCK bit must be set to 1. This causes the whole COMP<sub>x</sub>\_CSR register to become read-only, including the LOCK bit. The write protection can only be reset by a MCU reset.

### 14.3.9 Hysteresis

The comparator's programmable hysteresis voltage is able to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed to be able to force the hysteresis value.



199472

Figure 40. Comparator hysteresis

## 14.4 COMP register description

Table 43. Overview of COMP registers

Offset	Acronym	Register Name	Reset	Section
0x00, 0x04	COMP <sub>x</sub> _CSR(x=1, 2)	COMP x(x=1, 2) control and status register	0x00000000	Section 14.4.1
0x18	COMP_CRV	COMP external reference voltage register	0x00000000	Section 14.4.2
0x1C, 0x20	COMP <sub>x</sub> _POLL(x=1, 2)	COMP x(x=1, 2) polling register	0x00000000	Section 14.4.3

### 14.4.1 COMP control and status register (COMP<sub>x</sub>\_CSR)(x=1, 2)

Address offset: 0x00, 0x04

Reset value: 0x0000 0000

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

LOCK	OUT	Reserved												OFLT	HYST	
rw	r													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
POL	Res.	OUT_SEL			Res.	INP_SEL		Res.	INM_SEL		MODE		Res.	EN		
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Field	Type	Reset	Description
31	LOCK	rw	0x00	Comparator lock This bit is write-once. It is set by software as '1' and can only be cleared by a system reset. It allows to have all control bits of comparator x as read-only. 1: COMPx_CSR is read-only. 0: COMPx_CSR is read-write.
30	OUT	r	0x00	Comparator x output This read-only bit is a copy of comparator x output state. 1: Output is high (non-inverting input above inverting input). 0: Output is low (non-inverting input below inverting input).
29:21	Reserved			always read as 0
20:18	OFLT	rw	0x00	Comparator x output filter These bits control the comparator x output filter. Continuous unchanged PCLK2 clock COMP outputs are considered effective results, otherwise the configuration remain the same. 111: 128 clock cycles 110: 64 clock cycles 101: 32 clock cycles 100: 16 clock cycles 011: 8 clock cycles 010: 4 clock cycles 001: 2 clock cycles 000: no filter
17:16	HYST	rw	0x00	Comparator x hysteresis These bits control the comparator x hysteresis level. 11: 90mV 10: 30mV 01: 15mV 00: 0mV

Bit	Field	Type	Reset	Description

15	POL	rw	0x00	Comparator x output polarity This bit is used to invert the comparator x output polarity. 1: Output is inverted 0: Output is not inverted
14	Reserved			always read as 0
13:10	OUT_SEL	rw	0x00	Comparator x output selection These bits are used to select the destination of the comparator x output. 0010: Timer 1 break input 0110: Timer 1 Ocrefclear input 0111: Timer 1 input capture 1 1000: Timer 2 input capture 4 1001: Timer 2 Ocrefclear input 1010: Timer 3 input capture 1 1011: Timer 3 Ocrefclear input Other: No selection
9	Reserved			always read as 0.
8:7	INP_SEL	rw	0x00	Comparator x normal phase input selection These bits are used to select the signal source connected to the normal phase input of the comparator x.  Comparator 1: 00: COMP1_INP0(PA0) 01: COMP1_INP3(PA1) 10: COMP1_INP2(PA2) 11: COMP1_INP1(PA3)  Comparator 2: 00: COMP2_INP0(PA0) 01: COMP2_INP3(PA1) 10: COMP2_INP2(PA2) 11: COMP2_INP1(PA3)
6	Reserved			always read as 0

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

5:4	INM_SEL	rw	0x00	Comparator x inverting input selection These bits are used to select the signal source connected to the inverting input of the comparator x. Comparator 1: 00: COMP1_INM0(PA4) 01: COMP1_INM1(PA5) 10: COMP1_INM2(PA0) 11: COMP1_INM3(CRV) Comparator 2: 00: COMP2_INM0(PA4) 01: COMP2_INM1(PA5) 10: COMP2_INM2(PA2) 11: COMP2_INM3(CRV)
3:2	MODE	rw	0x00	Comparator x mode The operating mode control bits of the comparator x allows to adjust the speed/consumption. 11: Very-low speed/ultra-low power 10: Low speed/low-power 01: Medium speed/medium power 00: High speed/full power
1	Reserved			always read as 0
0	EN	rw	0x00	Comparator x enable This bit switches ON/OFF the comparator. 1: Comparator x enabled 0: Comparator x disabled

#### 14.4.2 COMP external reference voltage register (COMP\_CRV)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:6	Reserved			always read as 0.

5	CRV_SRC	rw	0x00	Comparator external reference voltage source select 0: VREF 1: AVDD
4	CRV_EN	rw	0x00	Comparator external reference voltage enable 1: COMP external reference voltage enabled 0: COMP external reference voltage disabled
3:0	CRV_SEL	rw	0x00	Comparator external reference voltage select This bit selects the comparator external reference voltage. 0000: 1/20AVDD / VREF 0001: 2/20AVDD / VREF 0010: 3/20AVDD / VREF 0011: 4/20AVDD / VREF 0100: 5/20AVDD / VREF 0101: 6/20AVDD / VREF 0110: 7/20AVDD / VREF 0111: 8/20AVDD / VREF 1000: 9/20AVDD / VREF 1001: 10/20AVDD / VREF 1010: 11/20AVDD / VREF 1011: 12/20AVDD / VREF 1100: 13/20AVDD / VREF 1101: 14/20AVDD / VREF 1110: 15/20AVDD / VREF 1111: 16/20AVDD / VREF

#### 14.4.3 COMP polling register (COMPx\_POLL)(x=1, 2)

Address offset: 0x1C, 0x20

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					POUT		Res.		PERIOD		Res.	FIXN	POLL_CH	POLL_EN	

Bit	Field	Type	Reset	Description
31:11	Reserved			always read as 0.

10:8	POUT	r	0x00	Polling output This bit is read-only. It reflects the polling channel output state. POUT[0] corresponds to channel 1. POUT[1] corresponds to channel 2. POUT[2] corresponds to channel 3. 1: Output is high (non-inverting input above inverting input). 0: Output is low (non-inverting input below inverting input).
7	Reserved			always read as 0
6:4	PERIOD	rw	0x00	Polling wait cycle Switch to the next polling channel every n PCLK2 cycles. 111: 128 clock cycles 110: 64 clock cycles 101: 32 clock cycles 100: 16 clock cycles 011: 8 clock cycles 010: 4 clock cycles 001: 2 clock cycles 000: 1 clock cycles
3	Reserved			always read as 0
2	FIXN	rw	0x00	Polling inverting input fix 1: Polling channel inverting input is fixed. It is determined by the INM_SEL in the CSR register. 0: Polling channel inverting input is not fixed. It changes simultaneously with the INP channel. At this point, the INM_SEL is invalid.
1	POLL_CH	rw	0x00	Comparator polling channel 1: polling channel 1/2/3. 0: polling channel 1/2. Note: At this point, the INP_SEL is invalid.
0	POLL_EN	rw	0x00	Comparator polling enable 1: Comparator polling enabled. 0: Comparator polling disabled.

# 15

# Advanced-Control Timer (TIM1)

## Advanced-Control Timer (TIM1)

### 15.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1) and general-purpose (TIMx) timers are completely independent, and do not share any resources. They can be synchronized together as described in the general-purpose timer synchronization section.

### 15.2 Main features

TIM1 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in reset state or in a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare

- Break signal input
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

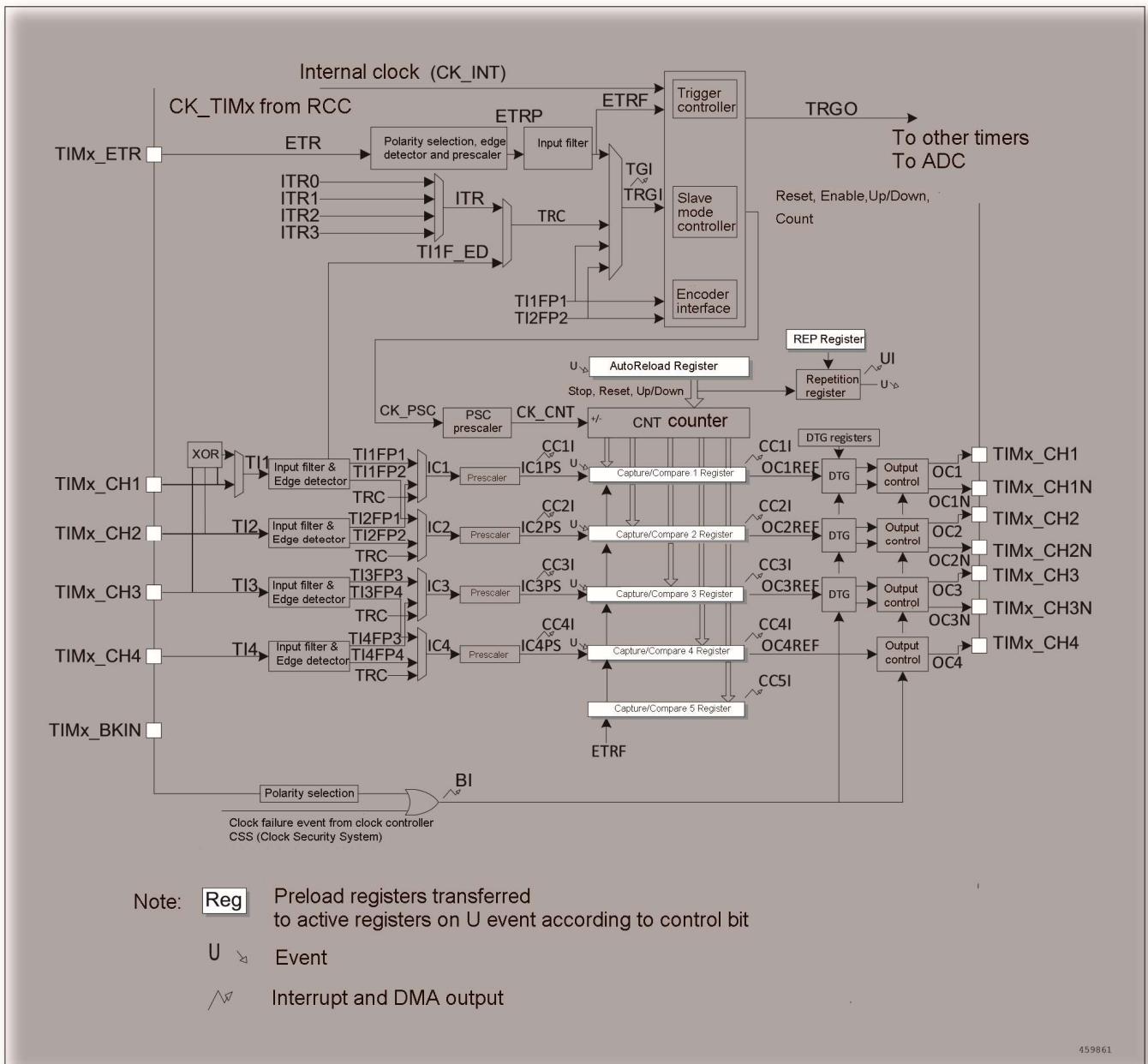


Figure 41. Advanced-control timer block diagram

## 15.3 Functional description

### 15.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)
- Repetition counter register (TIMx\_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register immediately or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx\_CR1 register.

### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

The figures below give some examples of the counter behavior when the prescaler ratio is changed on the fly:

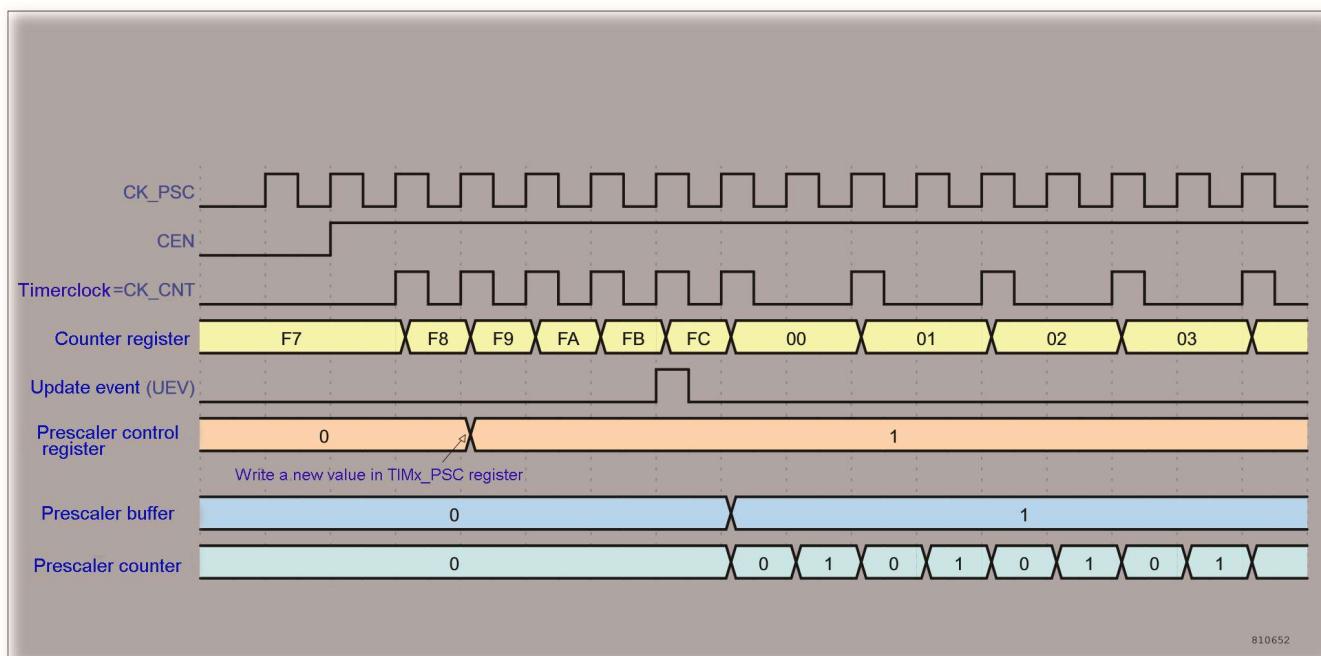


Figure 42. Counter timing diagram with prescaler division change from 1 to 2

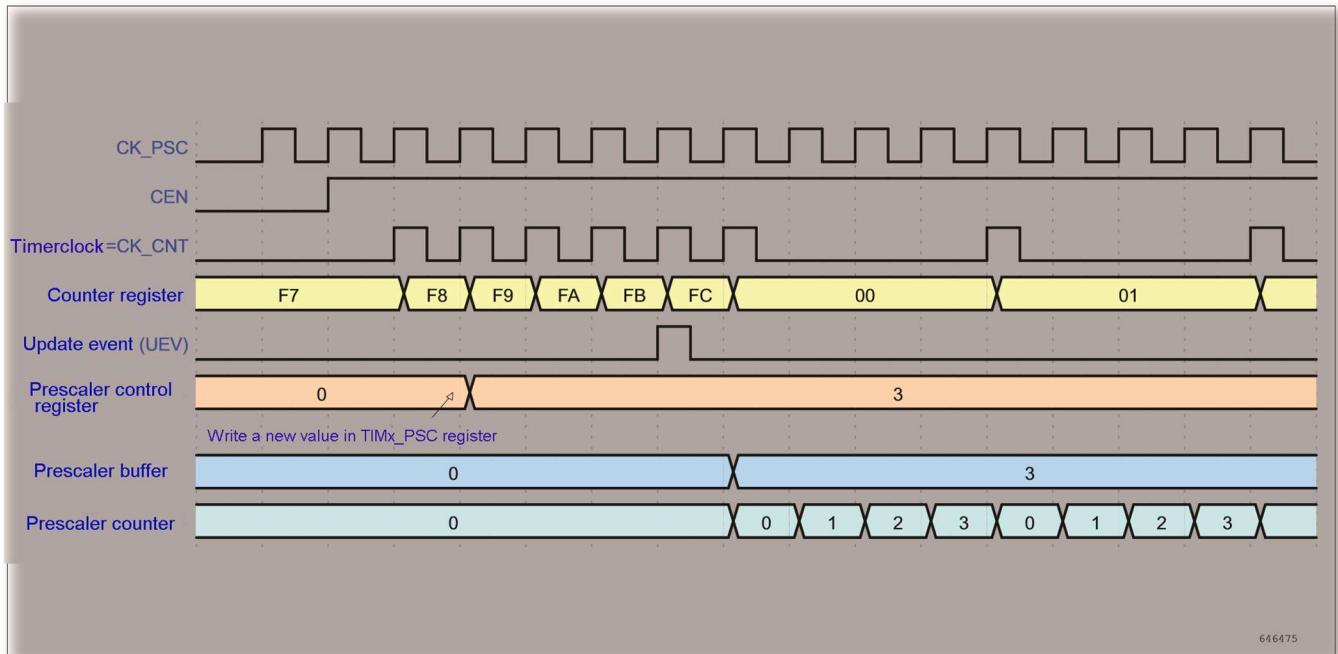


Figure 43. Counter timing diagram with prescaler division change from 1 to 4

### 15.3.2 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx\_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change) when an update event should be generated. In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag by hardware (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture mode.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set by hardware (depending on the URS bit).

- The repetition counter is reloaded with the content of TIMx\_RCR register.
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR = 0x36.

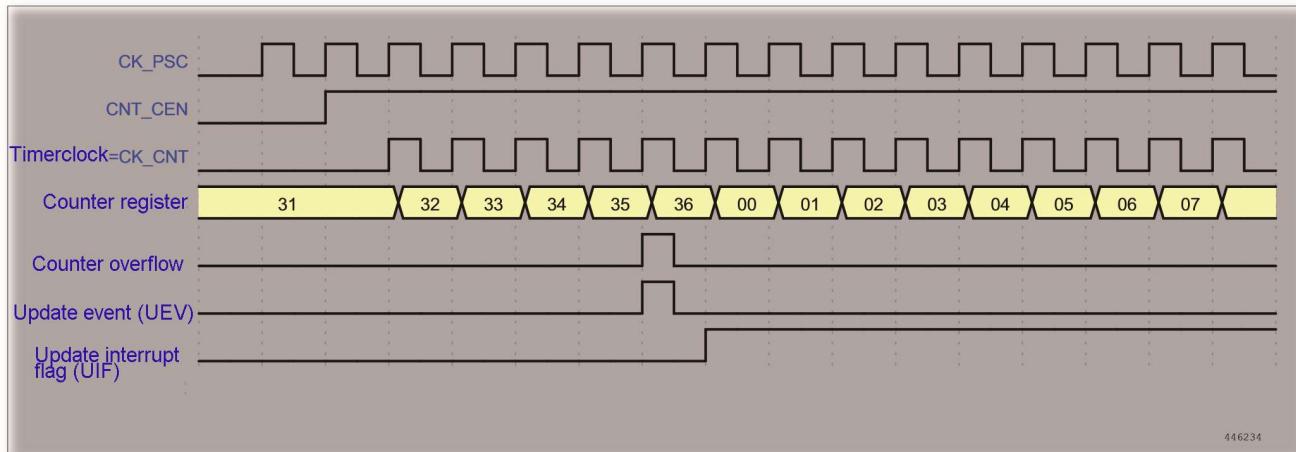


Figure 44. Counter timing diagram, internal clock divided by 1

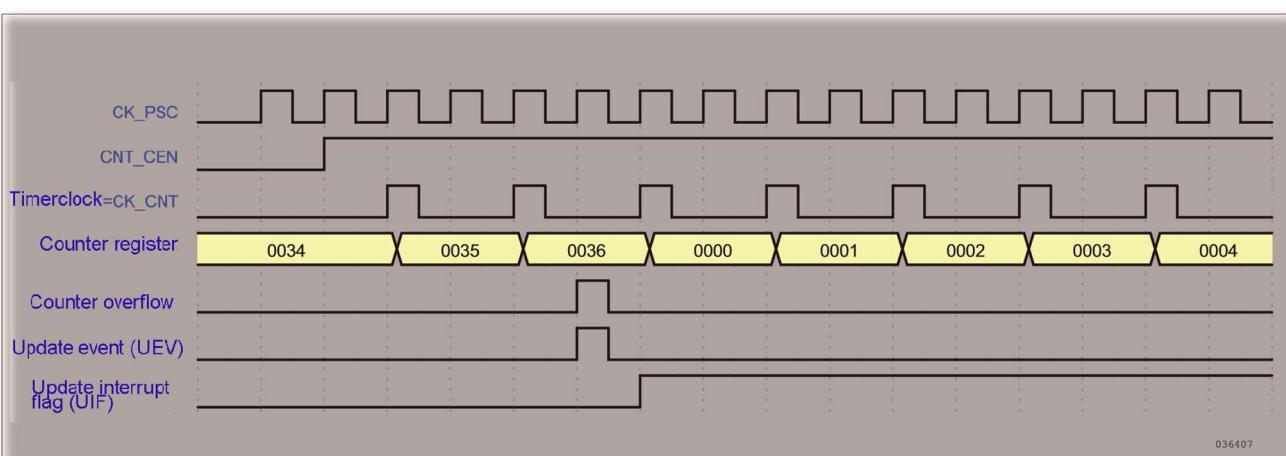


Figure 45. Counter timing diagram, internal clock divided by 2

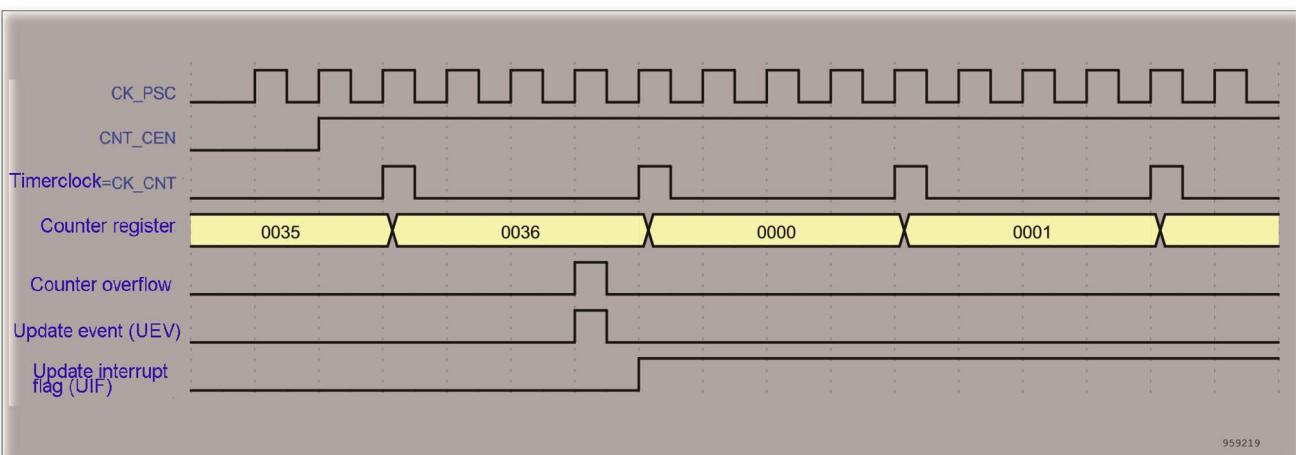


Figure 46. Counter timing diagram, internal clock divided by 4

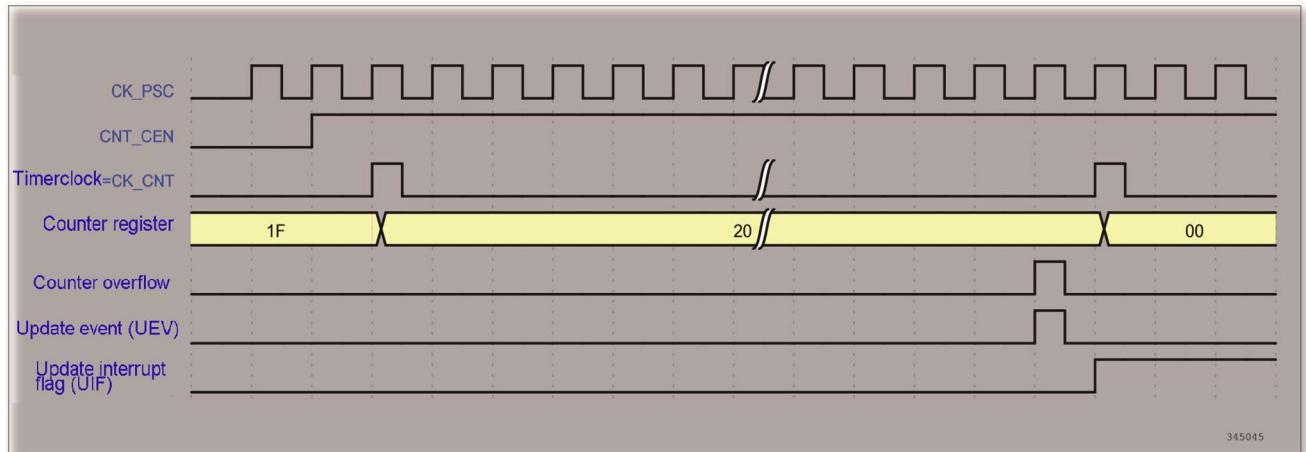


Figure 47. Counter timing diagram, internal clock divided by N

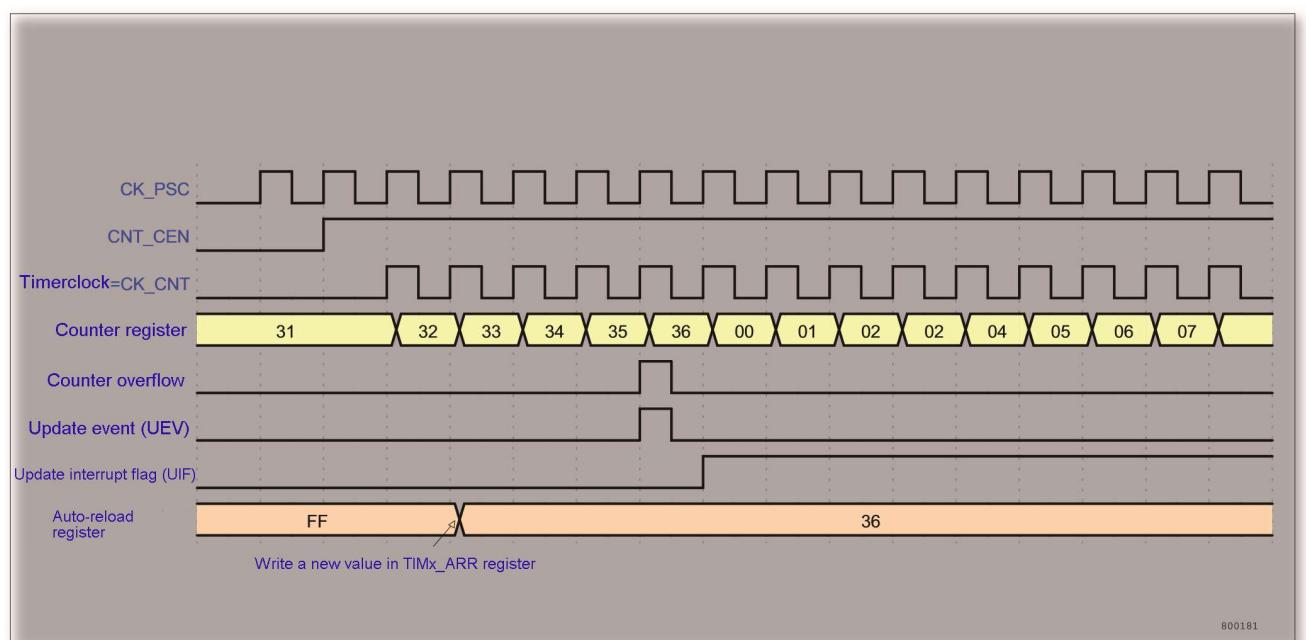


Figure 48. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)

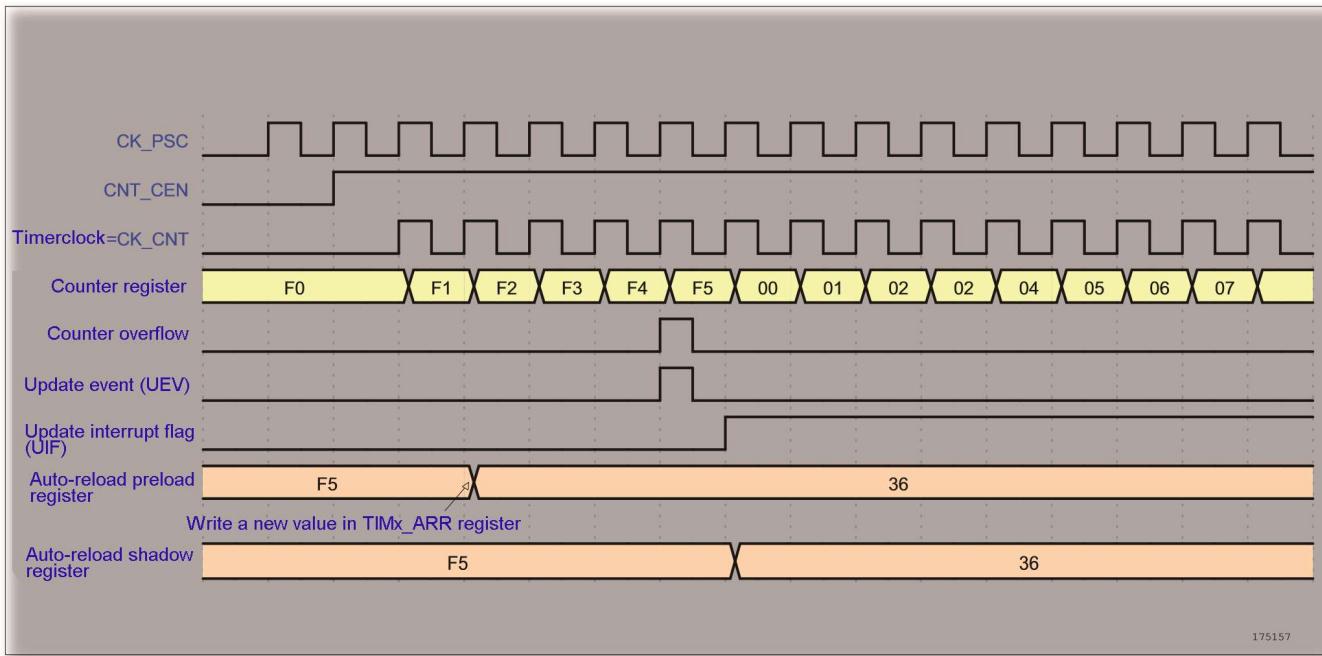


Figure 49. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)

### Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx\_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx\_RCR). Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit).

- The repetition counter is reloaded with the content of TIMx\_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register).

Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when  $\text{TIMx\_ARR} = 0x36$ .

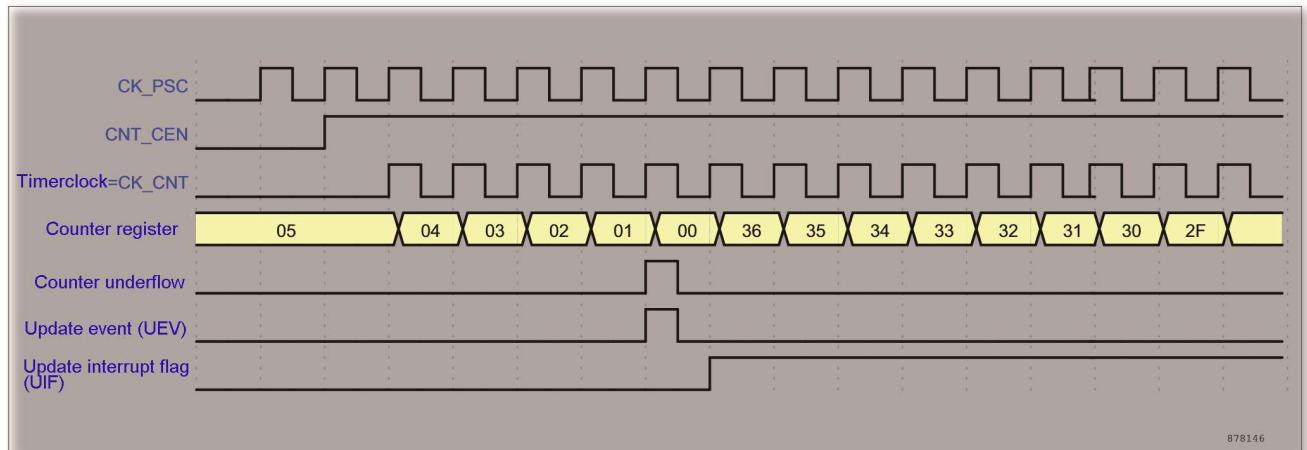


Figure 50. Counter timing diagram, internal clock divided by 1

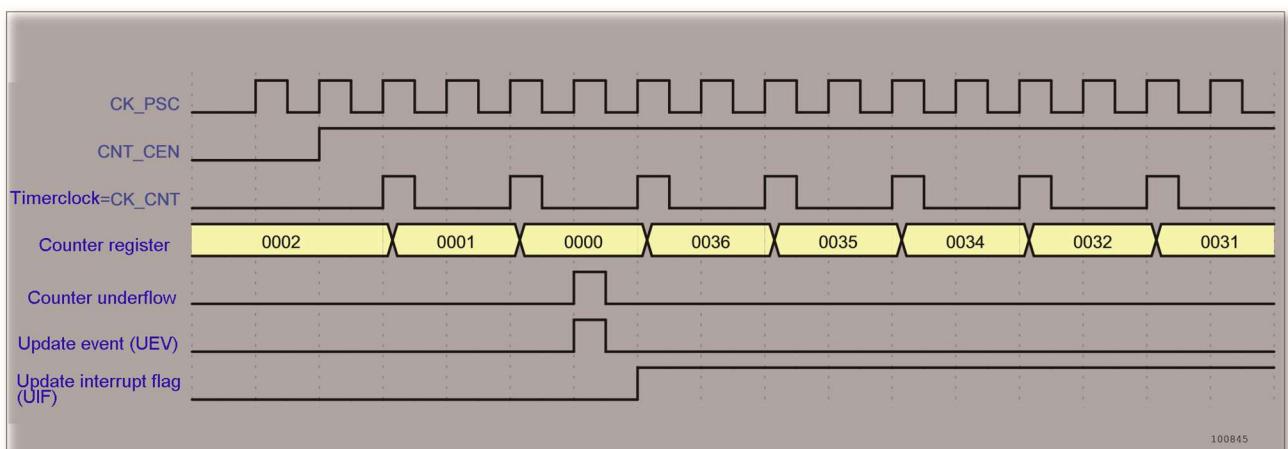


Figure 51. Counter timing diagram, internal clock divided by 2

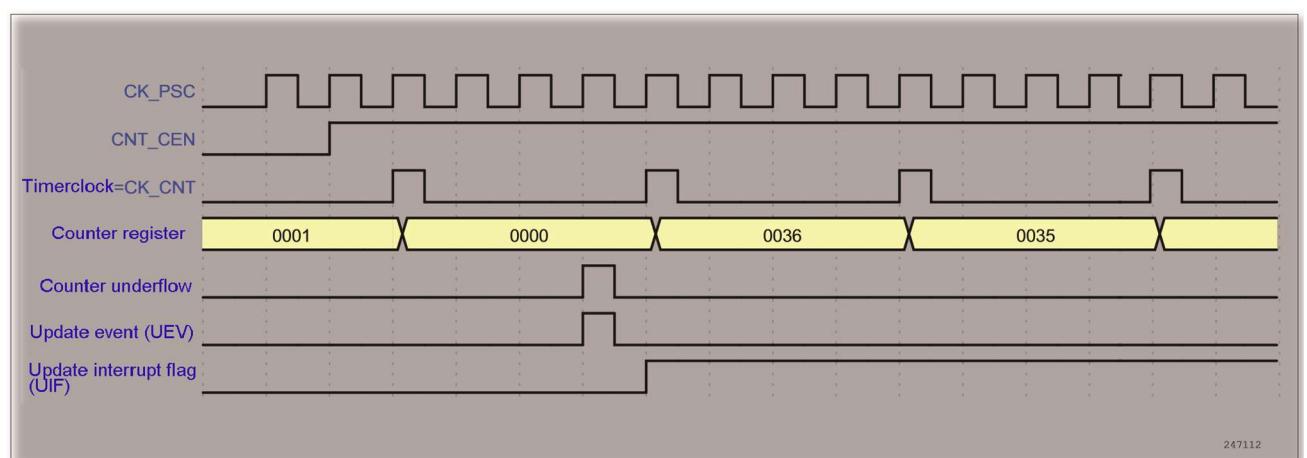


Figure 52. Counter timing diagram, internal clock divided by 4

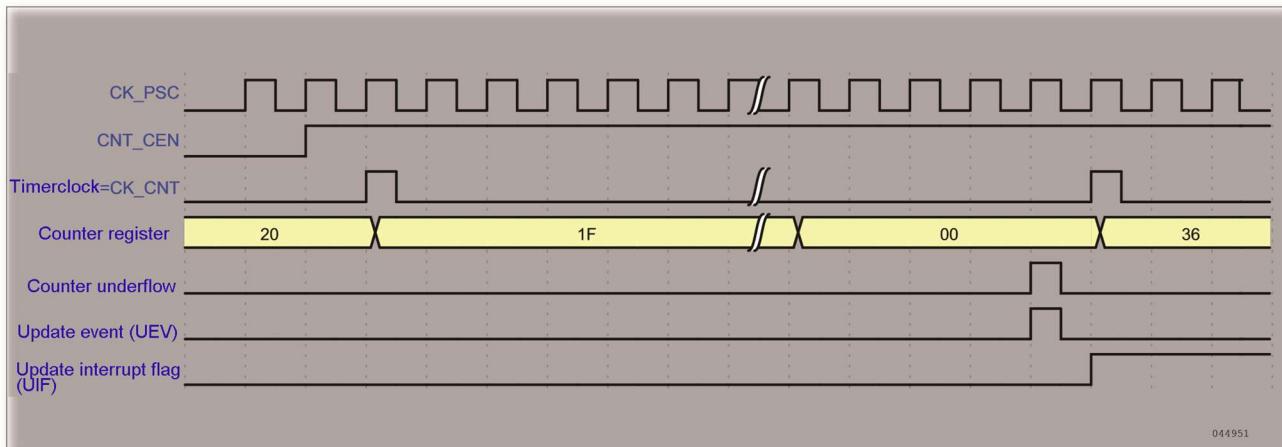


Figure 53. Counter timing diagram, internal clock divided by N

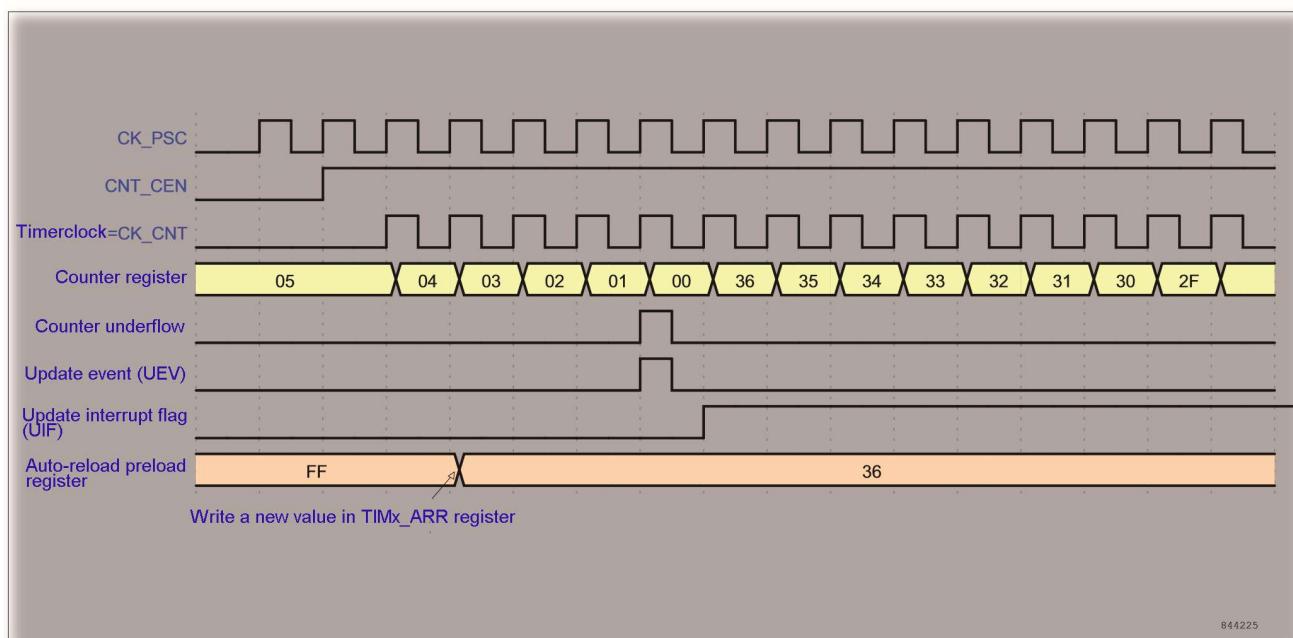


Figure 54. Counter timing diagram, update event when repetition counter is not used

### Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

In this mode, the DIR direction bit in the TIMx\_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller). In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit).

- The repetition counter is reloaded with the content of TIMx\_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register).

Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies:

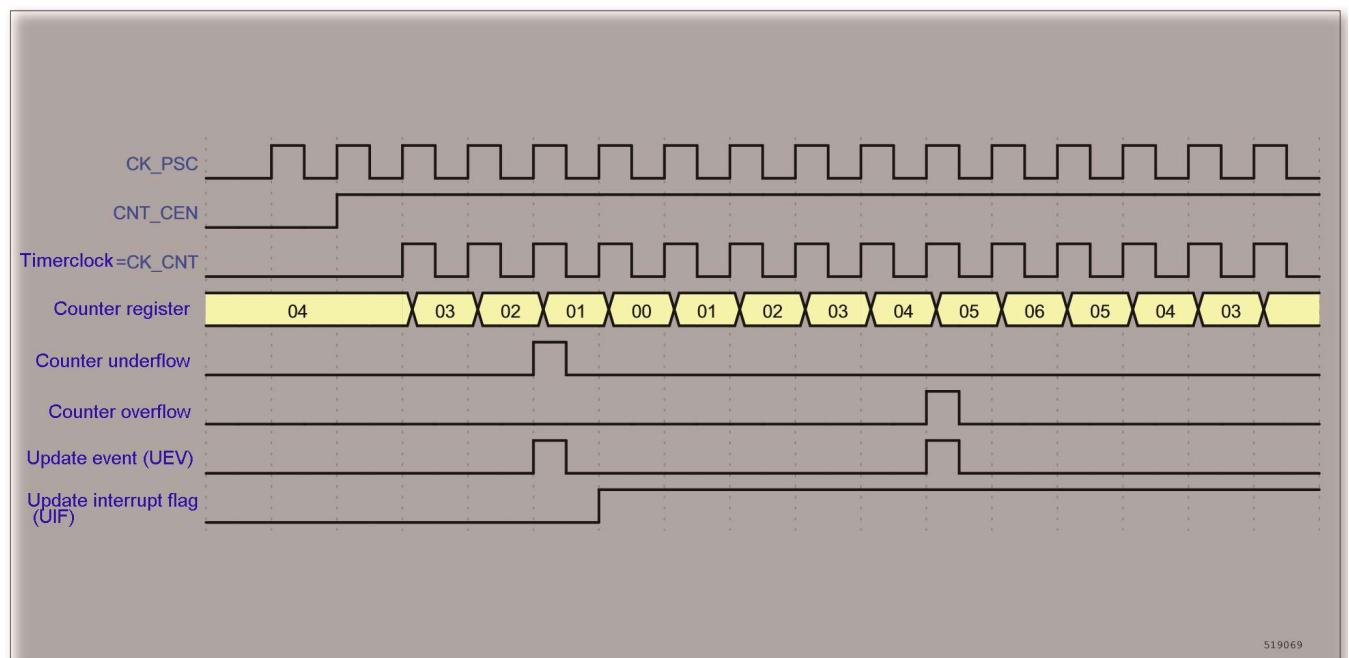


Figure 55. Counter timing diagram, internal clock divided by 1, TIMx\_ARR = 0x6

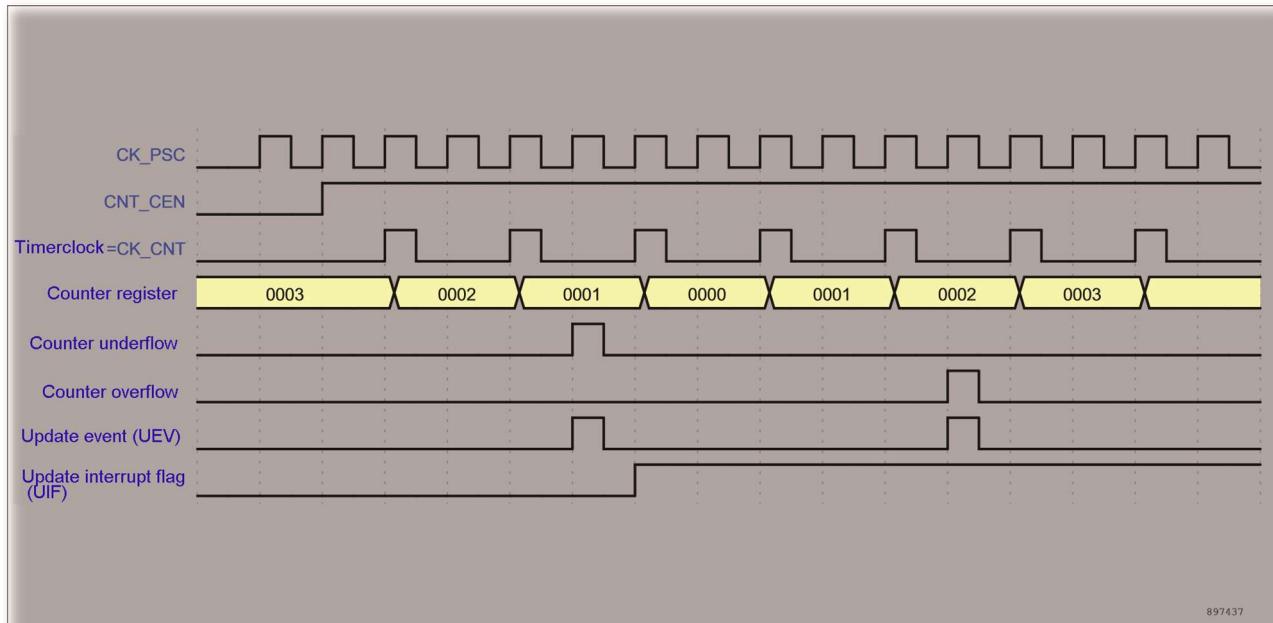


Figure 56. Counter timing diagram, internal clock divided by 2

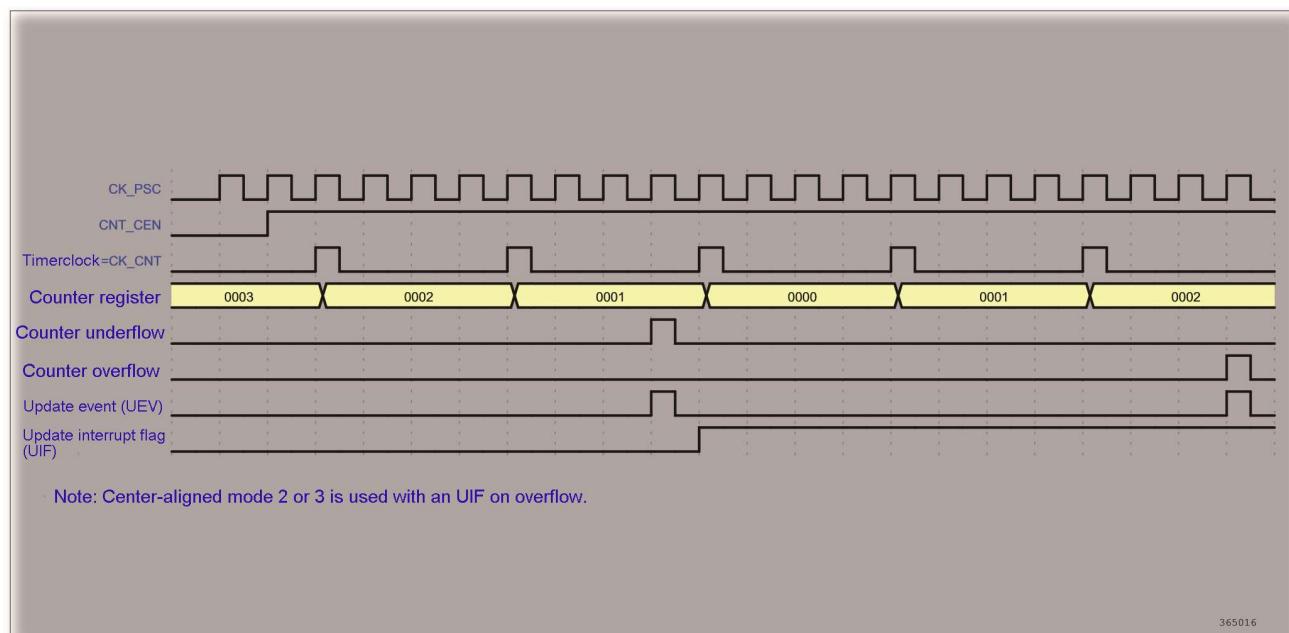


Figure 57. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x03

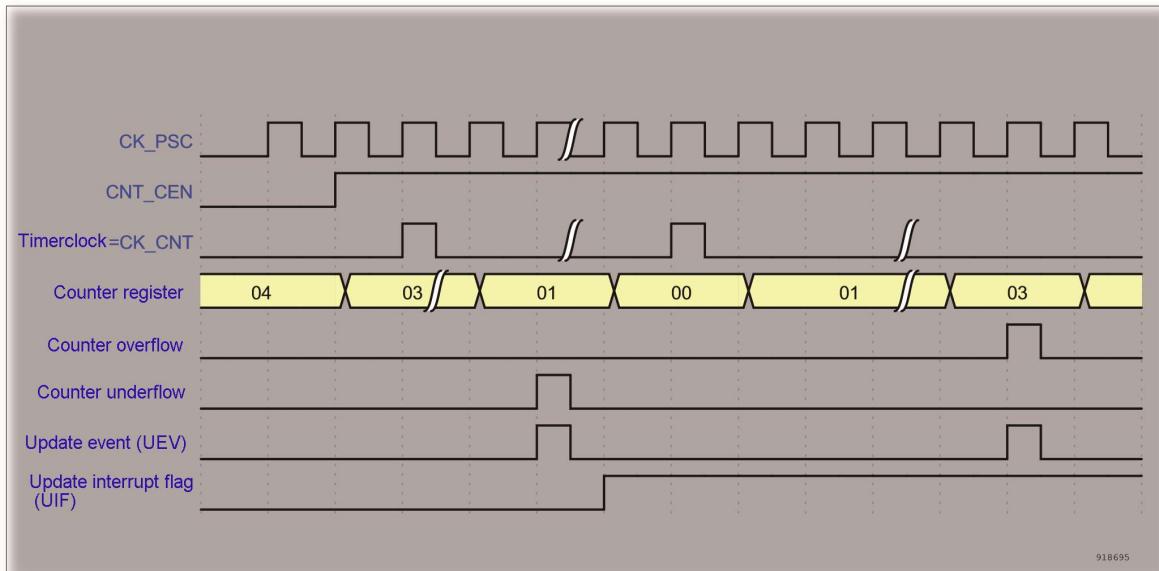


Figure 58. Counter timing diagram, internal clock divided by N

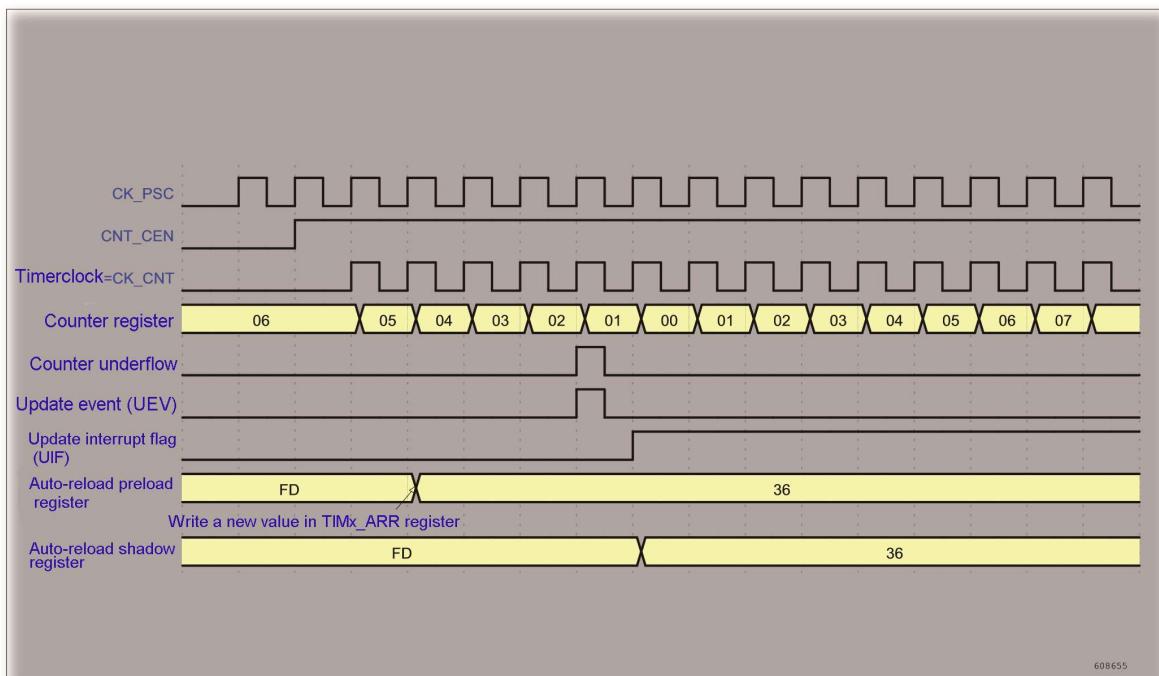


Figure 59. Counter timing diagram, update event with ARPE=1 (counter underflow)

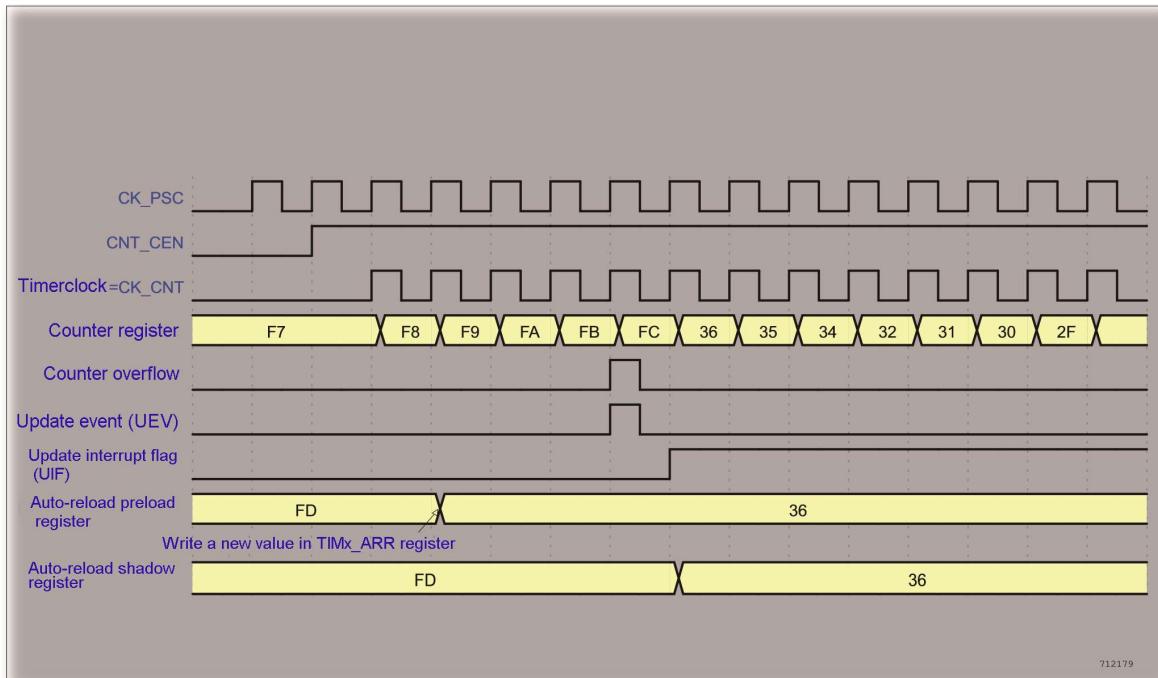


Figure 60. Counter timing diagram, Update event with ARPE=1 (counter overflow)

### 15.3.3 Repetition counter

'Time-base unit' describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

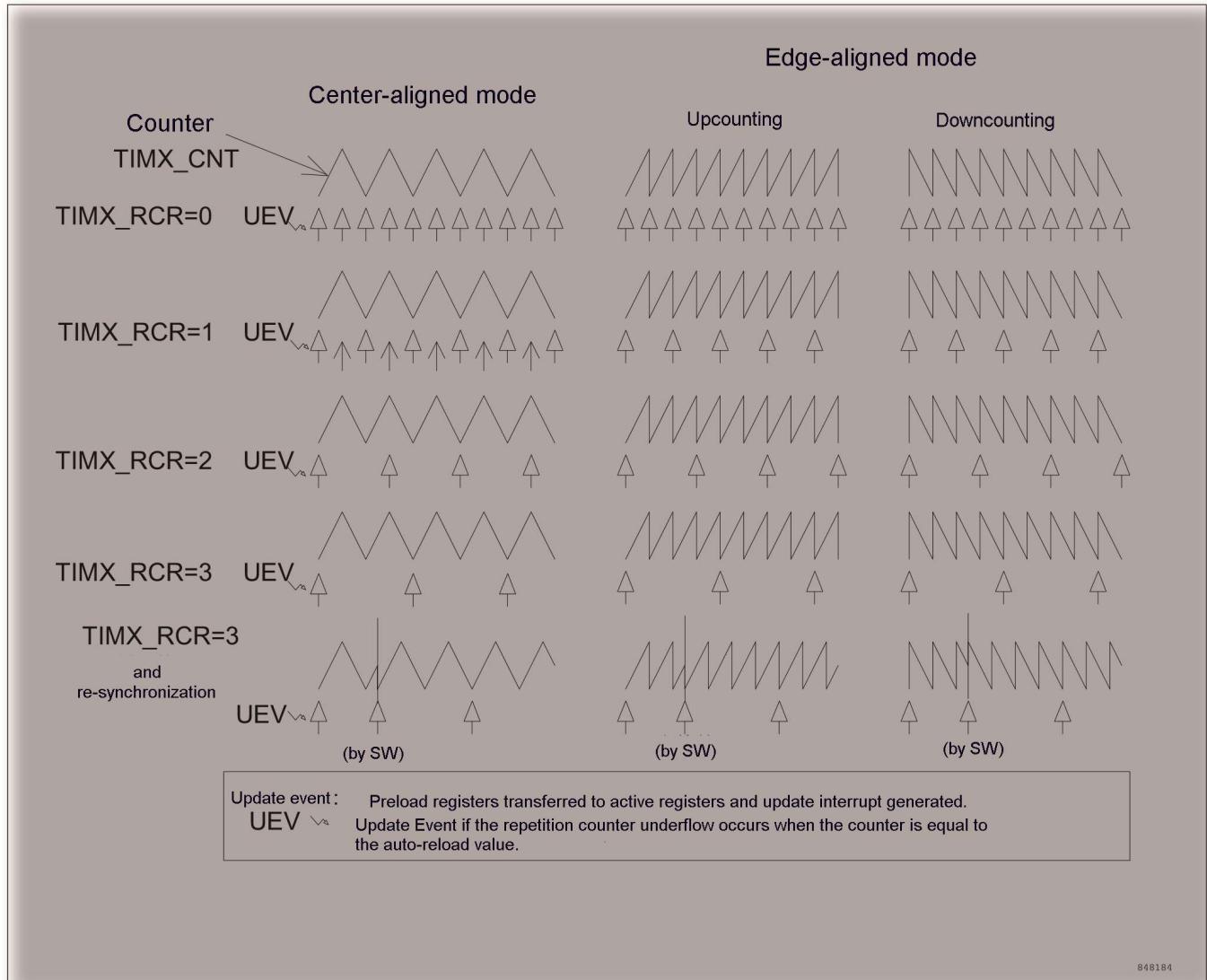
This means that data are transferred from the preload registers to the shadow registers (TIMx\_ARR auto-reload register, TIMx\_PSC preload register, but also TIMx\_CCRx capture/compare registers in compare mode) every N counter overflows or underflows, where N is the value in the TIMx\_RCR repetition counter register.

The repetition counter is decremented:

- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 128 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is  $2 \times T_{ck}$ , due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx\_RCR register value (refer to Figure 61). When the update event is generated by software (by setting the UG bit in TIMx\_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx\_RCR register.

The value of the repetition counter is written to a new REP\_CNT register (Repetition counter value of real-time writing) in real time. This is used in the repetition counter modification mode, to move the update interrupt flag (UIF) to left by (REP-REP\_CNT) phases (to right when the subtracted value is negative) by shifting UIF detection point in real time. These bits should be written after the update event UG is generated (note writing to REP\_CNT before the update event is generated makes the displacement invalid).



848184

Figure 61. Update rate examples depending on mode and TIMx\_RCR register settings

### 15.3.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK\_INT).
- External clock mode 1: external input pin (TIx).
- External clock mode 2: external trigger input (ETR).
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, the user can configure Timer 1 to act as a prescaler for Timer 2.

### Internal clock source (CK\_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

The figure below shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

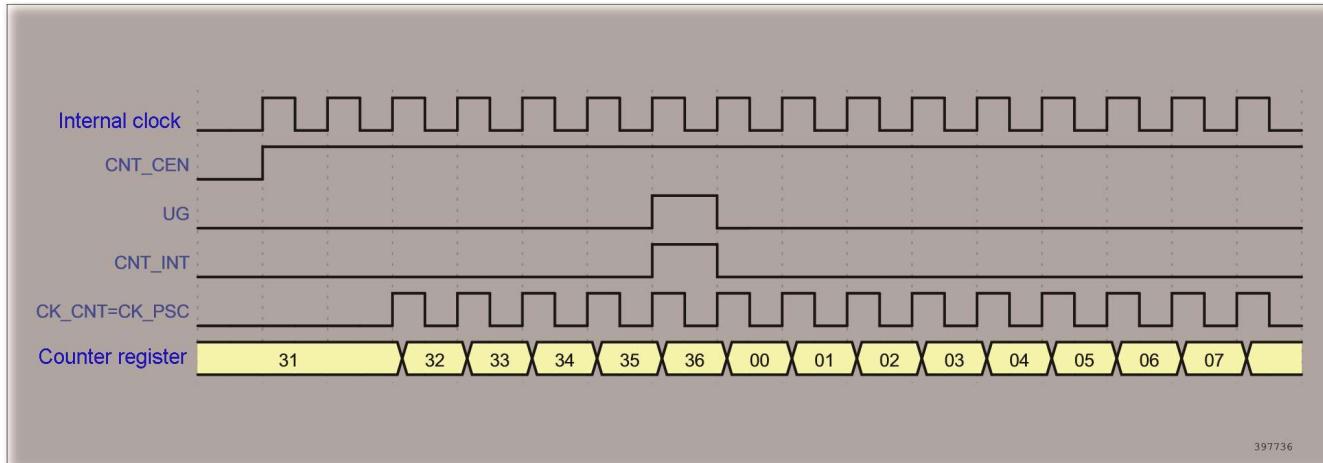


Figure 62. Control circuit in normal mode, internal clock divided by 1

### External clock source mode 1

This mode is selected when SMS=111 in the TIMx\_SMCR register. The counter can count at each rising or falling edge on a selected input.

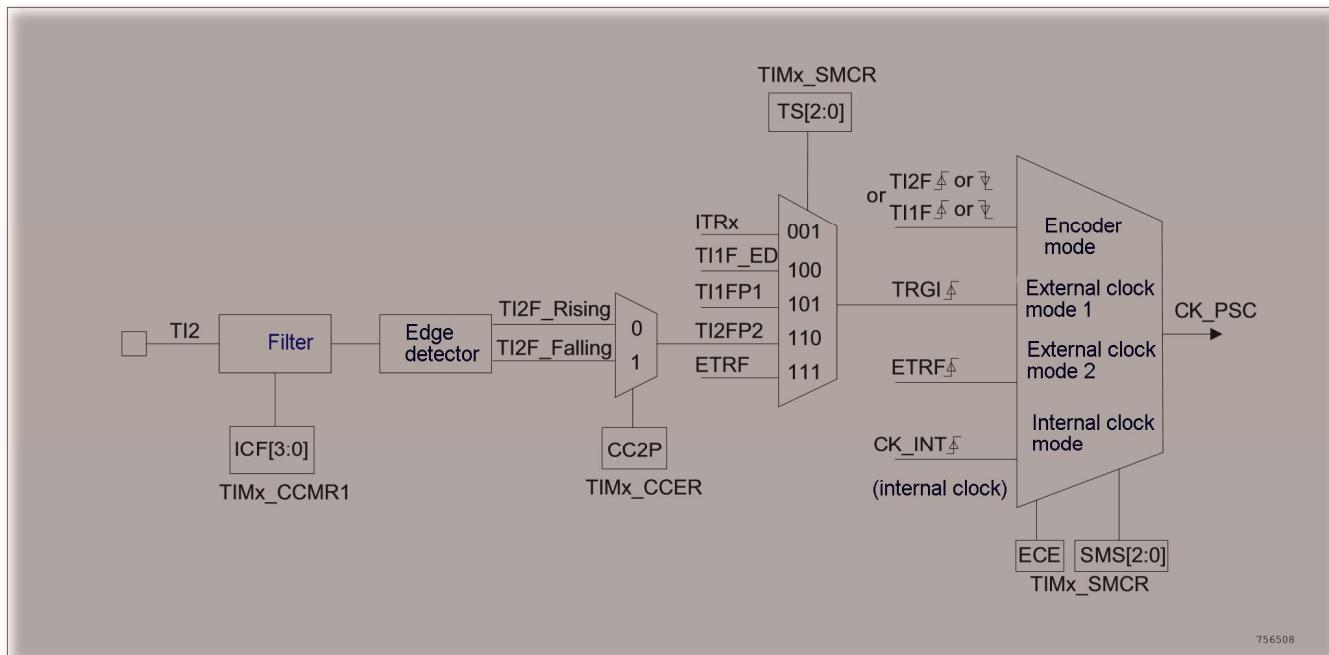


Figure 63. TI2 external clock connection example

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = 01 in the TIMx\_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 in the TIMx\_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx\_SMCR register.
5. Select TI2 as the trigger input source by writing TS = 110 in the TIMx\_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx\_CR1 register.

Note: The capture prescaler is not used for triggering, so the user does not need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set. The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

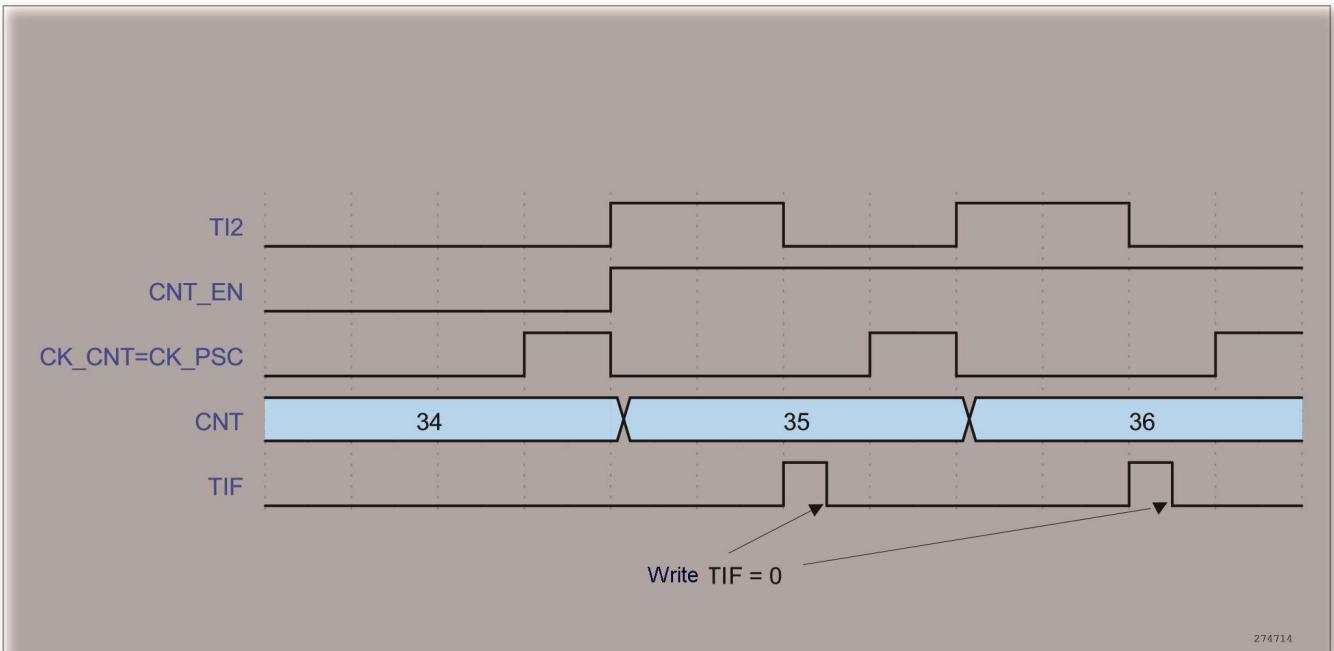


Figure 64. Control circuit in external clock mode 1

## External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx\_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The figure below gives an overview of the external trigger input block.

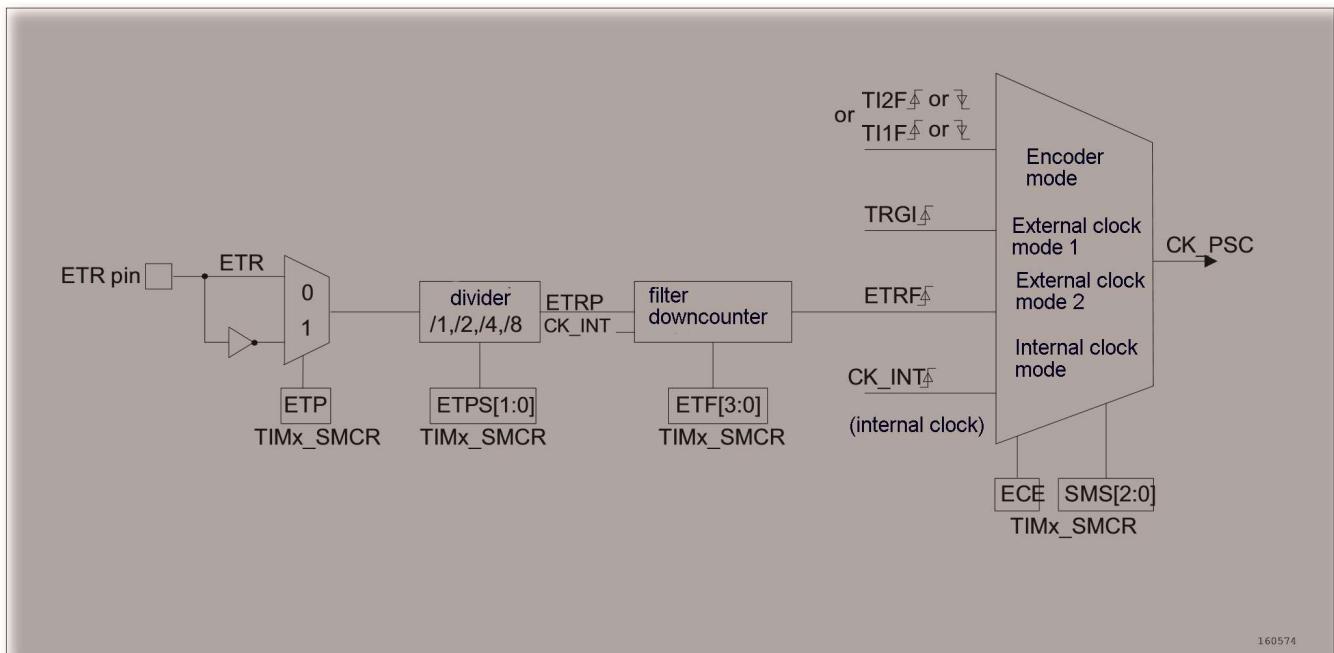


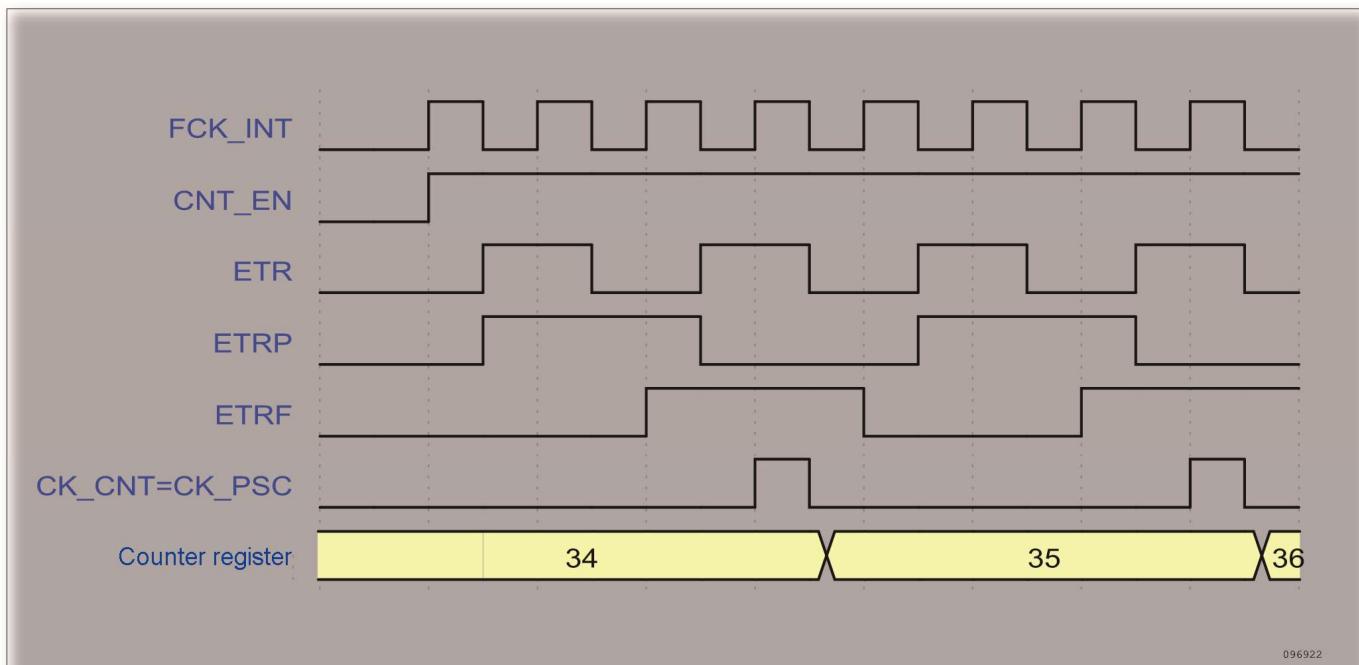
Figure 65. External trigger input block

For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

- As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx\_SMCR register.

- Set the prescaler by writing ETPS[1:0] = 01 in the TIMx\_SMCR register.
- Select rising edge detection on the ETR pin by writing ETP = 0 in the TIMx\_SMCR register.
- Enable external clock mode 2 by writing ECE = 1 in the TIMx\_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx\_CR1 register. The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.



### 15.3.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 67 to Figure 70 give an overview of one Capture/Compare channel.

The input stage samples the corresponding TI<sub>x</sub> input to generate a filtered signal TI<sub>x</sub>F. Then, an edge detector with polarity selection generates a signal (TI<sub>x</sub>FP<sub>x</sub>) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (IC<sub>x</sub>PS).

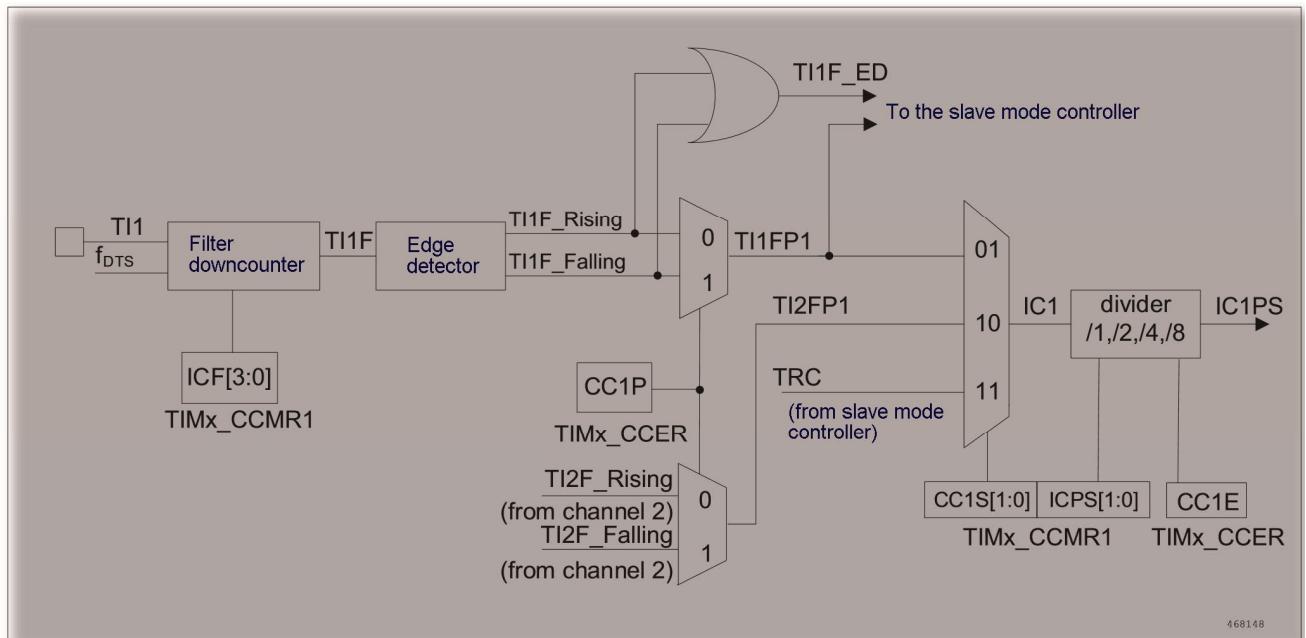


Figure 67. Capture/compare channel (example: channel 1 input stage)

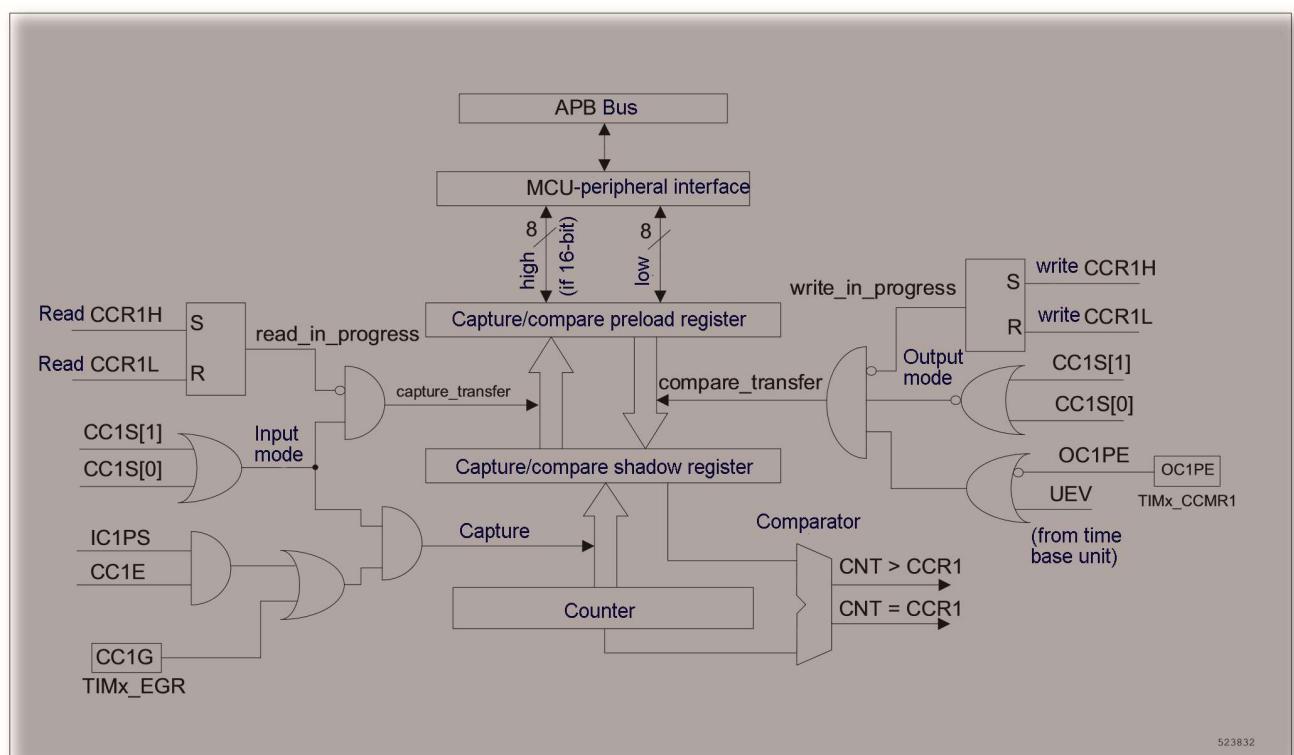


Figure 68. Capture/compare channel 1 main circuit

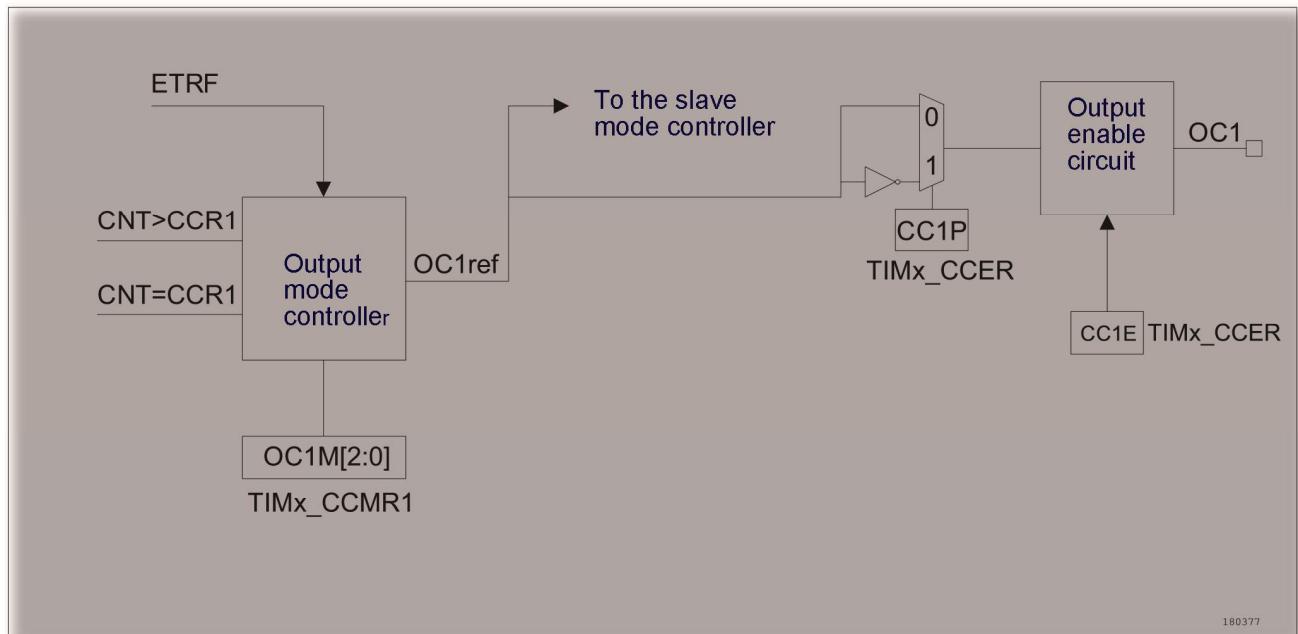


Figure 69. Output stage of capture/compare channel (channel 1 to 3)

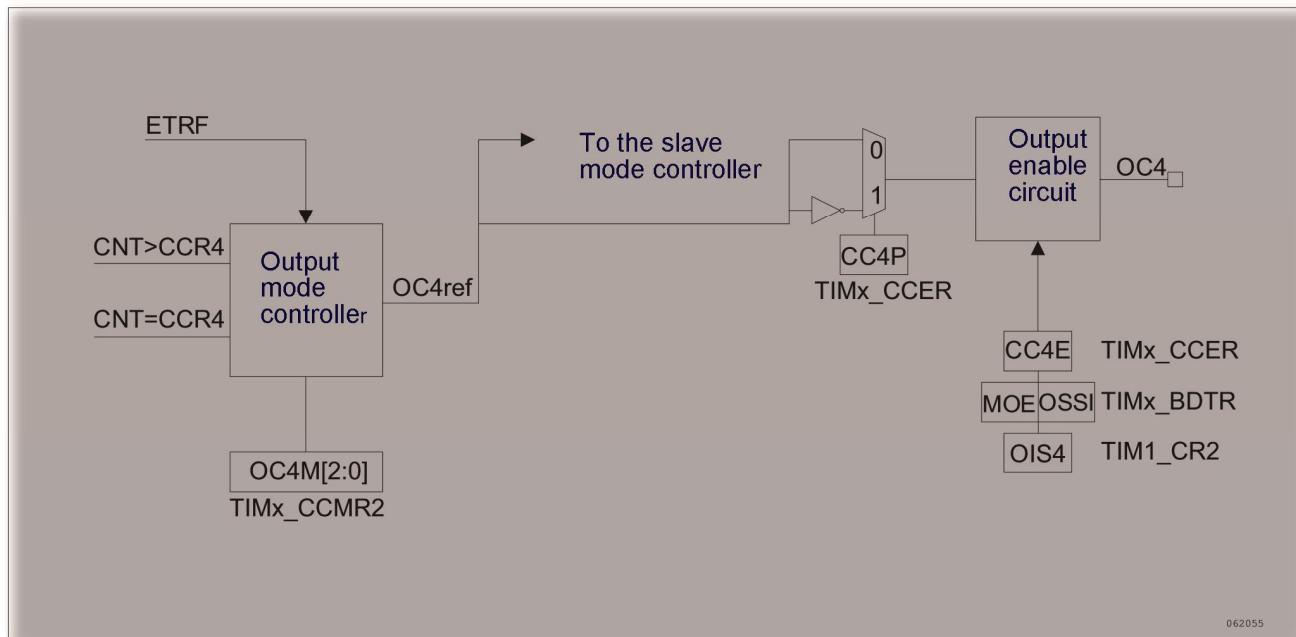


Figure 70. Output stage of capture/compare channel (channel 4)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register. In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 15.3.6 Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx\_CCRx) are used to latch the value of the counter after an edge is detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent

if they are enabled. If a capture occurs while the CC<sub>x</sub>IF flag was already high, then the overcapture flag CC<sub>x</sub>OF (TIM<sub>x</sub>\_SR register) is set. CxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIM<sub>x</sub>\_CCR<sub>x</sub> register. CC<sub>x</sub>OF is cleared when written to '0'.

The following example shows how to capture the counter value in TIM<sub>x</sub>\_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIM<sub>x</sub>\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIM<sub>x</sub>\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIM<sub>x</sub>\_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the input signal (by programming ICxF bits in the TIM<sub>x</sub>\_CCMR<sub>x</sub> register if the input is a TI<sub>x</sub> input). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate an edge transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIM<sub>x</sub>\_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIM<sub>x</sub>\_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIM<sub>x</sub>\_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit to '1' in the TIM<sub>x</sub>\_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIM<sub>x</sub>\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM<sub>x</sub>\_DIER register.

When an input capture occurs:

- The TIM<sub>x</sub>\_CCR1 register gets the value of the counter on the active level transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set to '1' if at least two consecutive captures occurred whereas the CC1IF flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CC<sub>x</sub>G bit in the TIM<sub>x</sub>\_EGR register.

### 15.3.7 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:

- Two IC<sub>x</sub> signals are mapped on the same TI<sub>x</sub> input.
- These 2 IC<sub>x</sub> signals are active on edges with opposite polarity.
- One of the two TI<sub>x</sub>FP signals is selected as trigger input and the slave mode controller is configured in reset mode. For example, user can measure the period (in TIM<sub>x</sub>\_CCR1 register) and the duty cycle (in TIM<sub>x</sub>\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):
  - Select the active input for TIM<sub>x</sub>\_CCR1: write the CC1S bits to 01 in the TIM<sub>x</sub>\_CCMR1 register (TI1 selected).
  - Select the active polarity for TI1FP1 (used both for capture in TIM<sub>x</sub>\_CCR1 and counter clear): write the CC1P bit to '0' (active on rising edge).

- Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P bit to '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx\_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx\_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

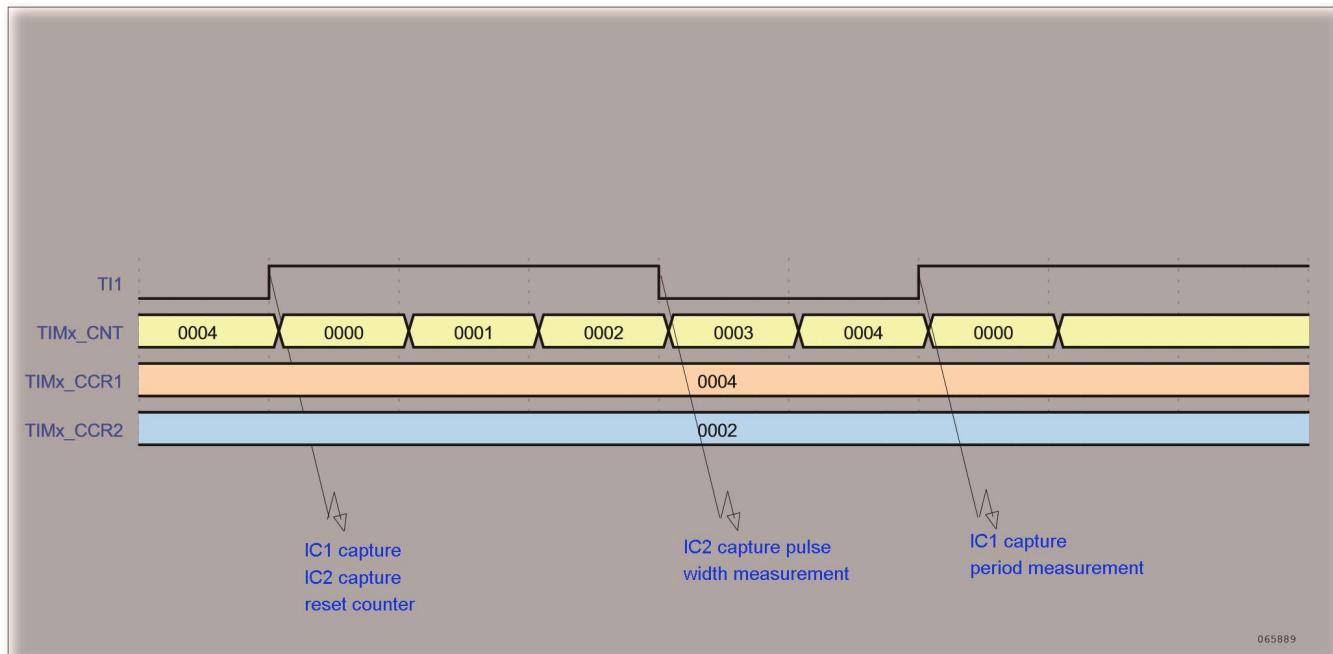


Figure 71. PWM input mode timing

The PWM input mode can be used only with the TIMx\_CH1/TIMx\_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

### 15.3.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCxREF is forced high (OCxREF is always active high) and OCx gets opposite value to CCxP polarity bit.

For example: CCxP = 0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bit to 100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 15.3.9 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bit in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM = 000), be set active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxE bit in the TIMx\_DIER register).
- Generates a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output.

The synchronization resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

The output compare mode is configured as follows:

- Select the counter clock (internal, external, prescaler).
- Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
- Set the CCxE bit if an interrupt request is to be generated.
- Select the output mode. For example:
  - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = 0 to disable preload register
  - Write CCxP = 0 to select active high polarity
  - Write CCxE = 1 to enable the output
- Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in the figure below.

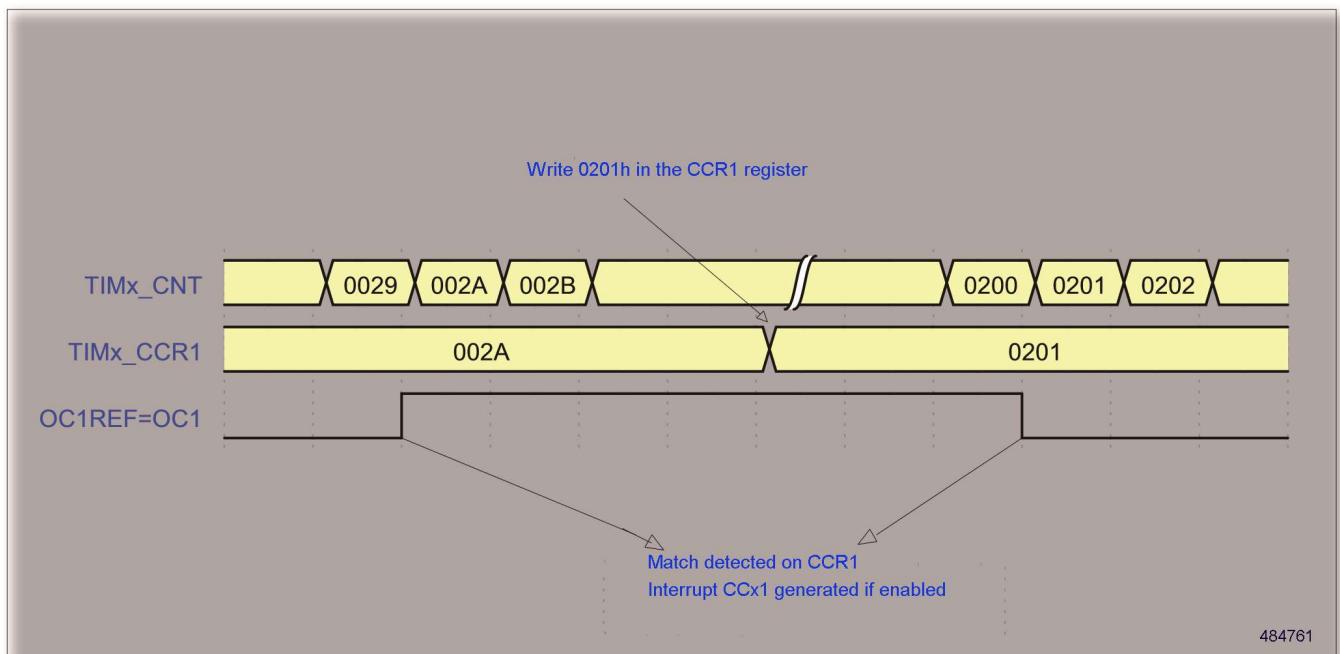


Figure 72. Output compare mode, toggle on OC1

### 15.3.10 PWM mode

Pulse Width Modulation mode allows generating a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bit in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user must initialize all the registers by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSS1 and OSSR bits (TIMx\_CCER and TIMx\_BDTR registers). Refer to the TIMx\_CCER register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether  $\text{TIMx\_CCRx} \leq \text{TIMx\_CNT}$  or  $\text{TIMx\_CNT} \leq \text{TIMx\_CCRx}$  (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx\_CR1 register.

#### PWM edge-aligned mode

##### Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to the section 15.3.2.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as  $\text{TIMx\_CNT} < \text{TIMx\_CCRx}$ ; else it becomes low. If the compare value in TIMx\_CCRx is greater than the auto-reload value (in TIMx\_ARR), then OCxREF is held at '1'. If the compare value is 0 then OCxREF is held at '0'. Figure 196 shows some edge-aligned PWM waveforms in an example where TIMx\_ARR = 8.

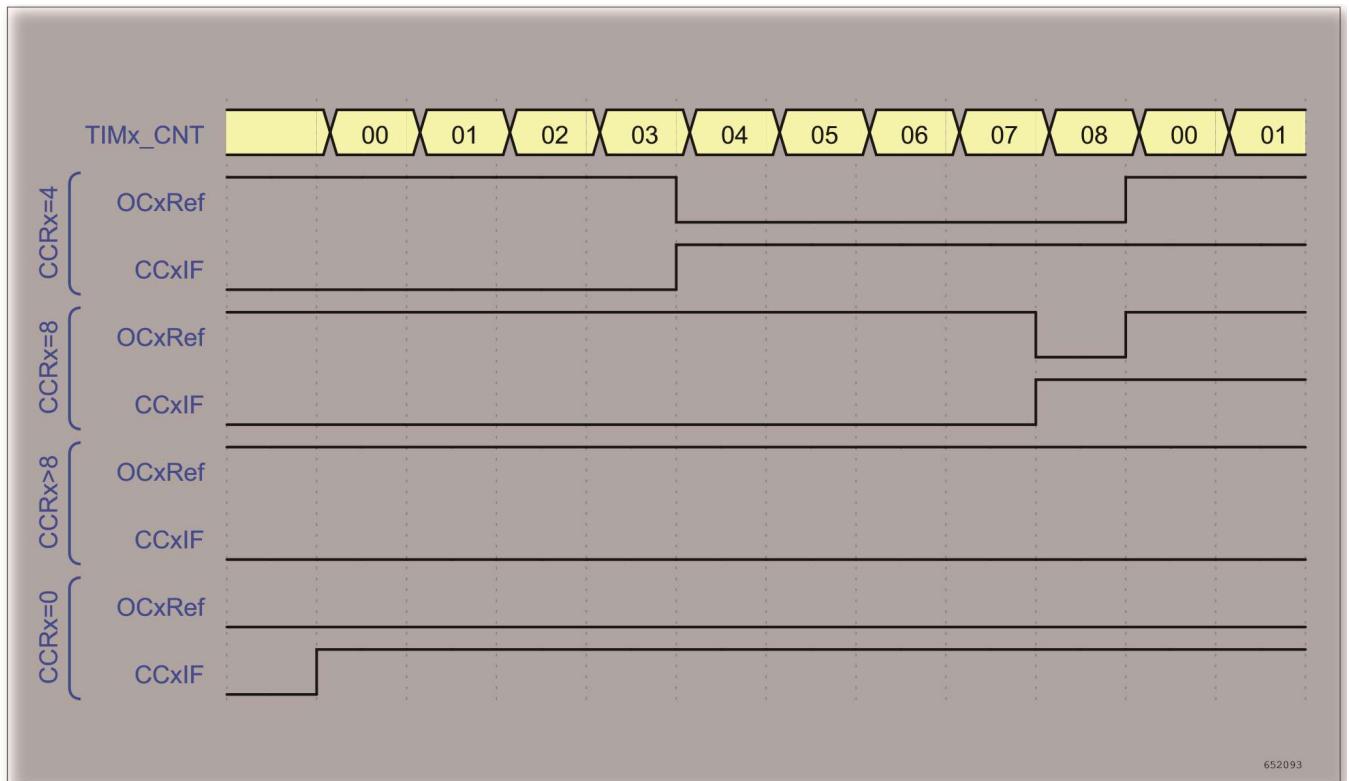


Figure 73. Edge-aligned PWM waveforms (ARR = 8)

### Downcounting configuration

Downcounting is active when DIR bit in TIMx\_CR1 register is high. Refer to the section 15.3.2.

In PWM mode 1, the reference signal OCxREF is low as long as TIMx\_CNT > TIMx\_CCRx; else it becomes high. If the compare value in TIMx\_CCRx is greater than the auto-reload value in TIMx\_ARR, then OCxREF is held at '1'. 0% PWM is not possible in this mode.

### PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxREF/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx\_CR1 register is updated by hardware and must not be changed by software. Refer to Center-aligned mode in section 15.3.2.

The figure below shows some center-aligned PWM waveforms in an example where:

- TIMx\_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx\_CR1 register

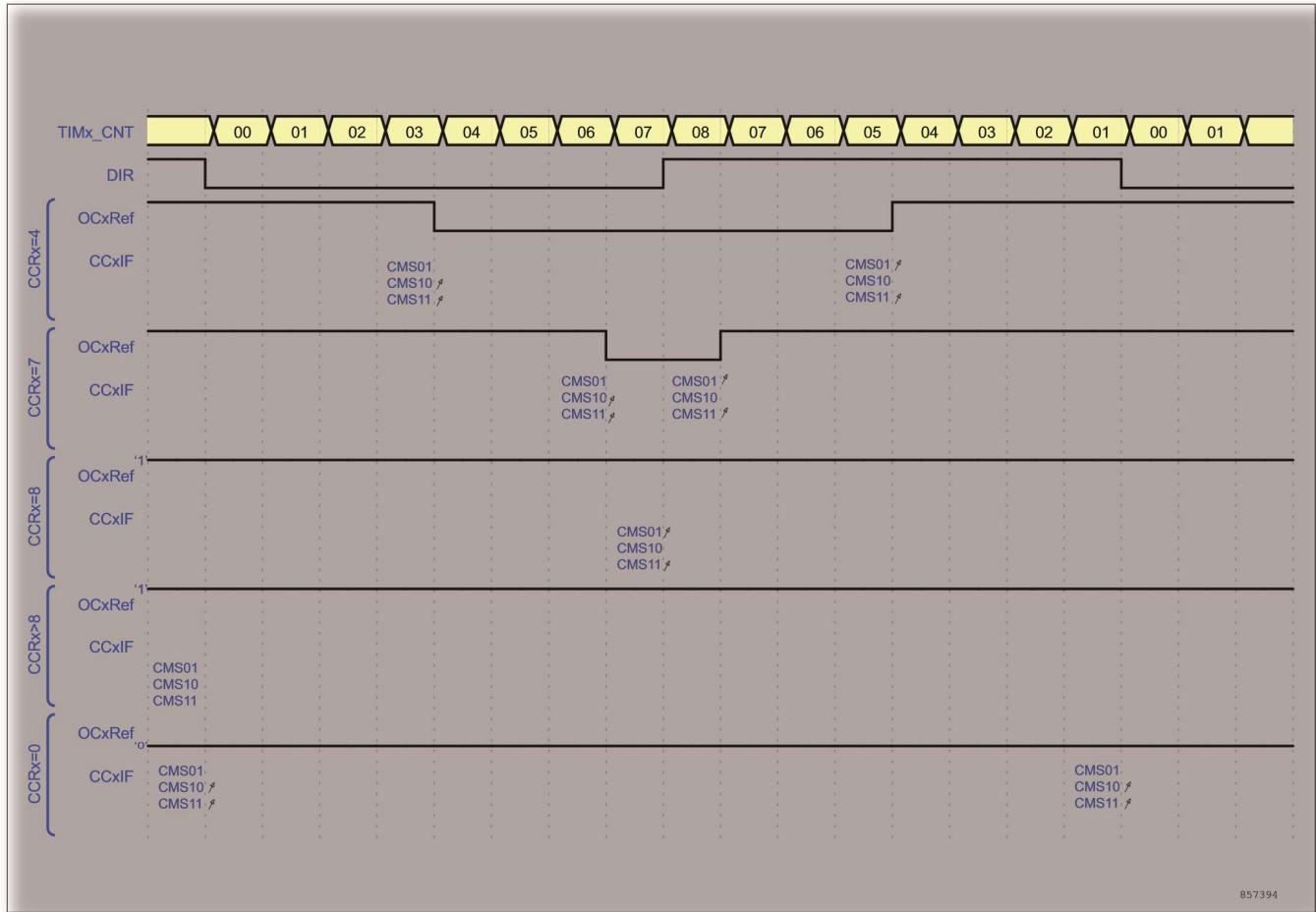


Figure 74. Center-aligned PWM waveforms (ARR = 8)

### Phase shift in the PWM center-aligned mode

PDER (channel x output PWM phase shift enable bit) and CCRxFALL (channel x capture/compare value when downcounting in the PWM center-aligned mode) are added to allow 5 channels to output PWM phase shift. To realize PWM outputs with programmable phase-shift waveform (left shift or right shift as required), the user should enable the PWM phase-shift function in the PDER register and set the CCRxFALL and CCRx.

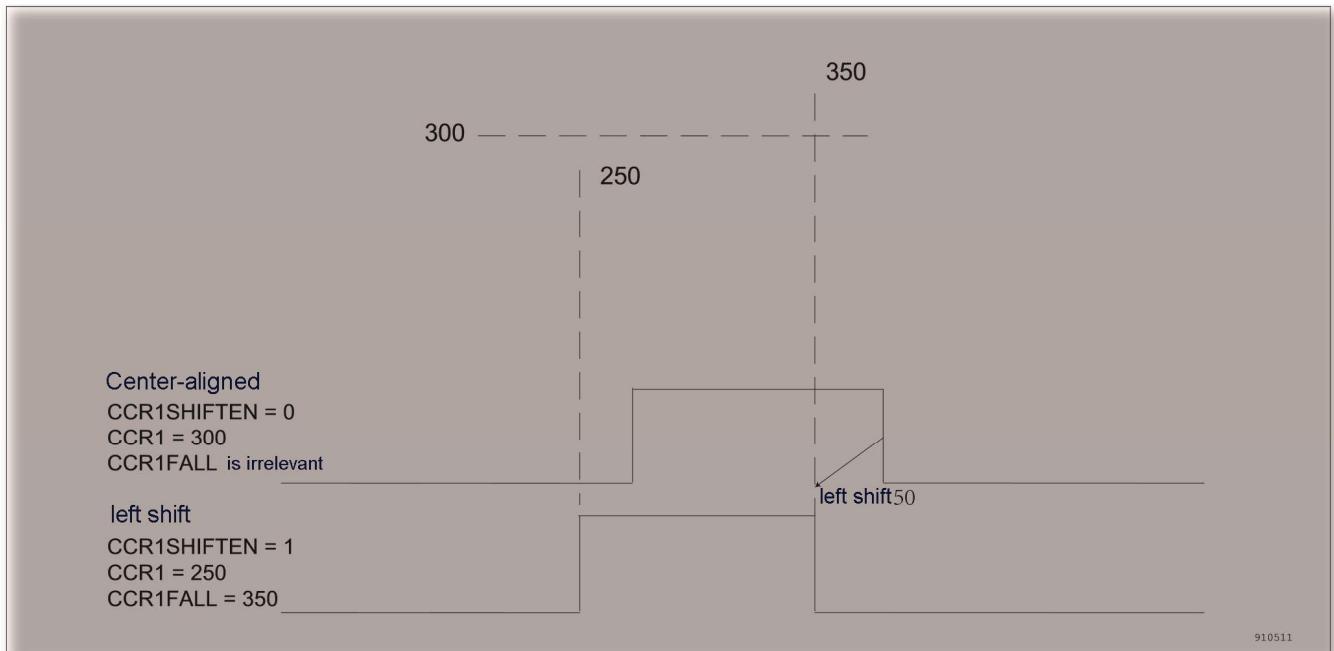


Figure 75. Phase shift diagram

#### Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if the user writes a value in the counter greater than the auto-reload value (TIMx\_CNT > TIMx\_ARR).
  - For example, if the counter was counting up, it will continue to count up.
  - The direction is updated if the user writes 0 or writes the TIMx\_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx\_EGR register) just before starting the counter and not to write the counter while it is running.

#### 15.3.11 Complementary outputs and dead-time insertion

The advanced-control timer (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs. This time is generally known as dead-time and it has to be adjusted depending on the devices connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

User can select the polarity of the outputs (main output OC<sub>x</sub> or complementary OC<sub>xN</sub>) independently for each output. This is done by writing to the CC<sub>xP</sub> and CC<sub>xNP</sub> bits in the TIMx\_CCER register.

The complementary signals OC<sub>x</sub> and OC<sub>xN</sub> are activated by a combination of several control bits: the CC<sub>xE</sub> and CC<sub>xNE</sub> bits in the TIMx\_CCER register and the MOE, OIS<sub>x</sub>, OIS<sub>xN</sub>, OSSI and OSSR bits in the TIMx\_BDTR and TIMx\_CR2 registers. For more details, refer to Table 47: Output control bits for complementary OC<sub>x</sub> and OC<sub>xN</sub> channels with break feature. In particular, the dead-time is activated when switching to the IDLE state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is a 10-bit dead-time generator in each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples)

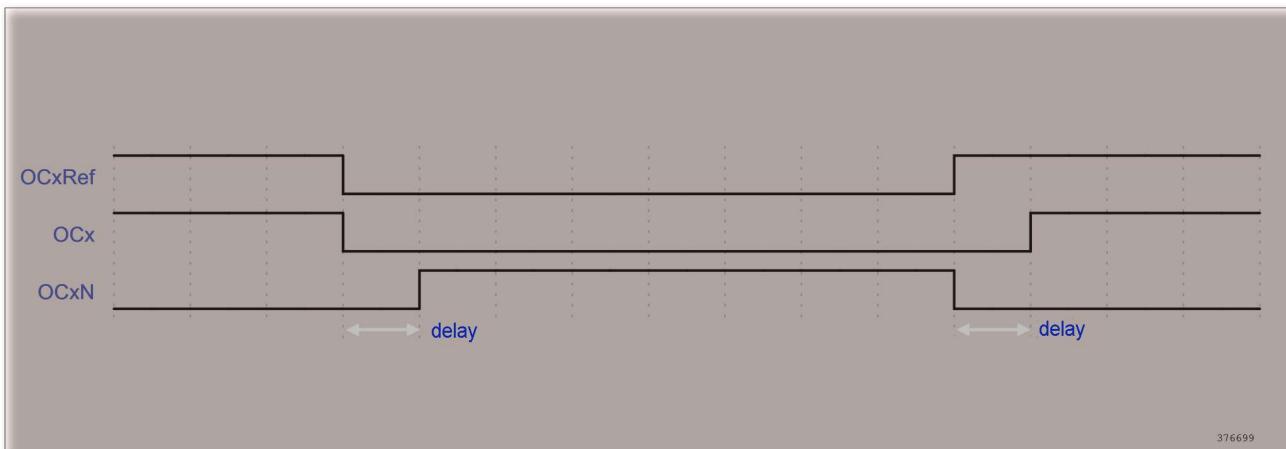


Figure 76. Complementary output with dead-time insertion

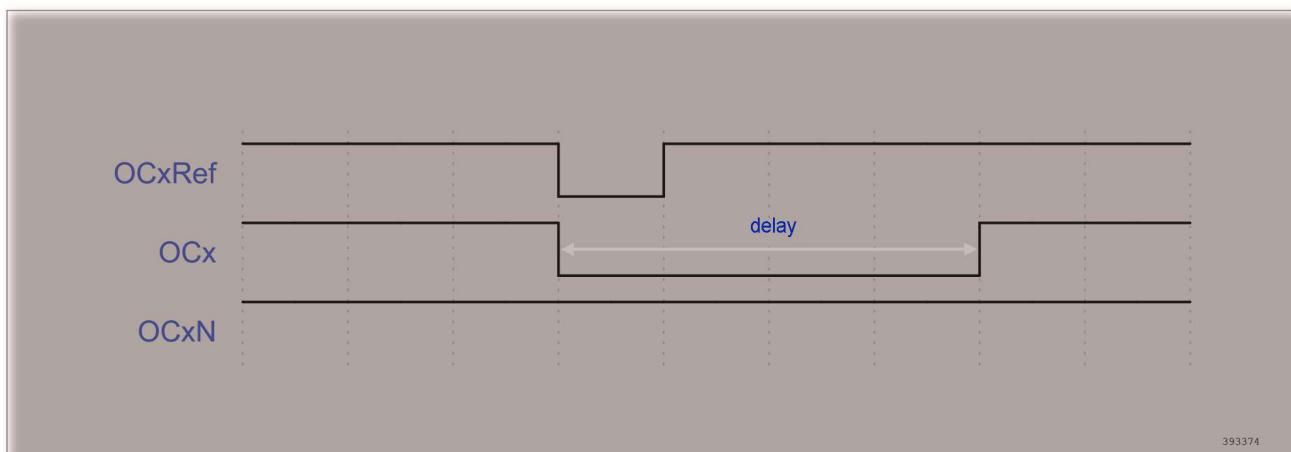


Figure 77. Dead-time waveforms with delay greater than the negative pulse

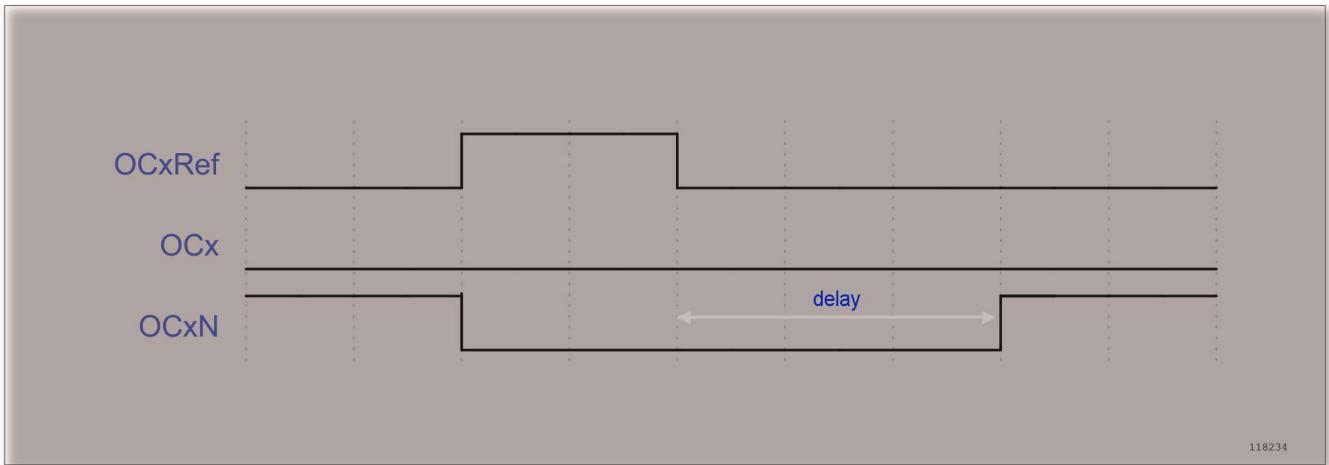


Figure 78. Dead-time waveforms with delay greater than the positive pulse

The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx\_BDTR register. Refer to the delay calculation in the Section 15.4.18.

### Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx\_CCER register.

This allows the user to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP = 0 then OCxN = OCxREF. On the other hand, when both OCx and OCxN are enabled (CCxE = CCxNE = 1), OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

#### 15.3.12 Using the break function

When using the break function, the output enable signals and inactive levels are modified according to corresponding control bits (MOE, OSS1 and OSSR bits in the TIMx\_BDTR register, OISx and OISxN bits in the TIMx\_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to output control bits for complementary OCx and OCxN channels with break feature in the table of registers.

The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller.

When exiting from reset, the break circuit is disabled and the MOE bit is low. User can enable the break function by setting the BKE bit in the TIMx\_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx\_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is written to 1 whereas it was low, a delay (dummy instruction) must be inserted before reading it correctly. This is because the user writes an asynchronous signal, but reads a synchronous signal.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or in reset state (selected by the OSS1 bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx\_CR2 register as soon as MOE = 0. If OSS1 = 0 then the timer releases the enable output else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 CK\_TIM clock cycles).
  - If OSS1 = 0 then the timer releases the enable outputs, else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- An interrupt can be generated if the BIE bit in the TIMx\_DIER register is set and the break status flag (BIF bit in the TIMx\_SR register) is set to '1'. A DMA request can be generated if the BDE bit in the TIMx\_DIER register is set.
- If the AOE bit in the TIMx\_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx\_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows freezing the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The user can choose from three levels of protection selected by the LOCK bits in the TIMx\_BDTR register. Refer to Section 15.4.8. The LOCK bits can be written only once after an MCU reset.

The figure below shows an example of behavior of the outputs in response to a break:

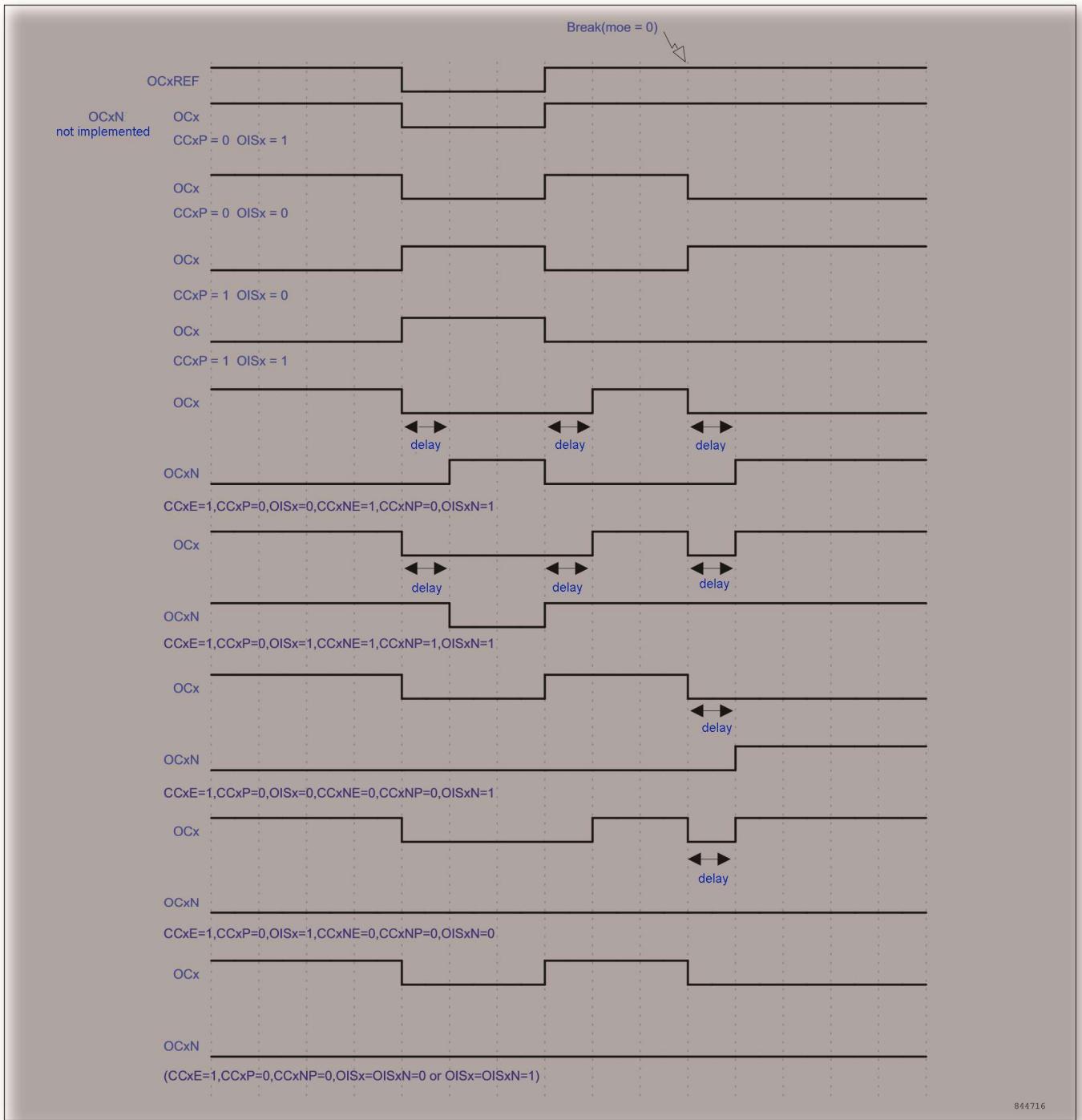


Figure 79. Output behavior in response to a break

### 15.3.13 Clearing the OCxREF signal on an external event

The OCxREF signal can be cleared by setting the OOCS bit in the TIMx\_SMCR register. The OCxREF signal of a given channel can be pulled low when OCCS = 0 and a high level is applied on the ETRF input (OCxCE enable bit in the corresponding TIMx\_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. The OCxREF signal can be pulled low when OCCS = 1 and a high level is applied on the OCREF\_CLR (output from comparator, COMPx\_CSR13:10] = 0110) signal.

This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

For example, the OC<sub>x</sub>REF signal can be connected to an external output. In this case, the ETR must be configured as follow:

- The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIM<sub>x</sub>\_SMCR register set to '00'.
- The external clock mode 2 must be disabled: bit ECE of the TIM<sub>x</sub>\_SMCR register set to '0'.
- The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

The figure below shows the behavior of the OC<sub>x</sub>REF signal corresponding to different OC<sub>x</sub>CE values when the ETRF Input becomes High. In this example, the timer TIM<sub>x</sub> is programmed in PWM mode.

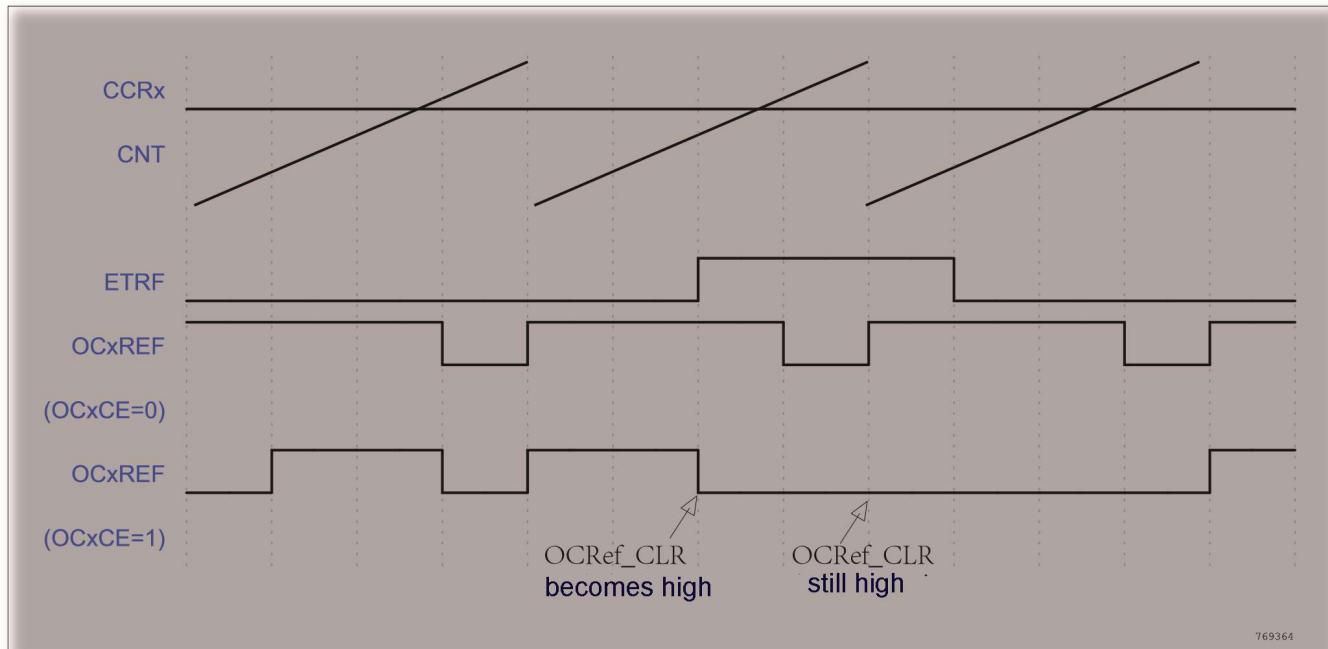


Figure 80. Clearing TIM<sub>x</sub> OC<sub>x</sub>REF

### 15.3.14 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OC<sub>x</sub>M, CC<sub>x</sub>E and CC<sub>x</sub>NE bits. The preload bits are transferred to the shadow bits at the COM commutation event. The user can thus program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIM<sub>x</sub>\_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIM<sub>x</sub>\_SR register), which can generate an interrupt (if the COMIE bit is set in the TIM<sub>x</sub>\_DIER register) or a DMA request (if the COMDE bit is set in the TIM<sub>x</sub>\_DIER register).

The figure below describes the behavior of the OC<sub>x</sub> and OC<sub>x</sub>N outputs when a COM event occurs, in 3 different examples of programmed configurations.

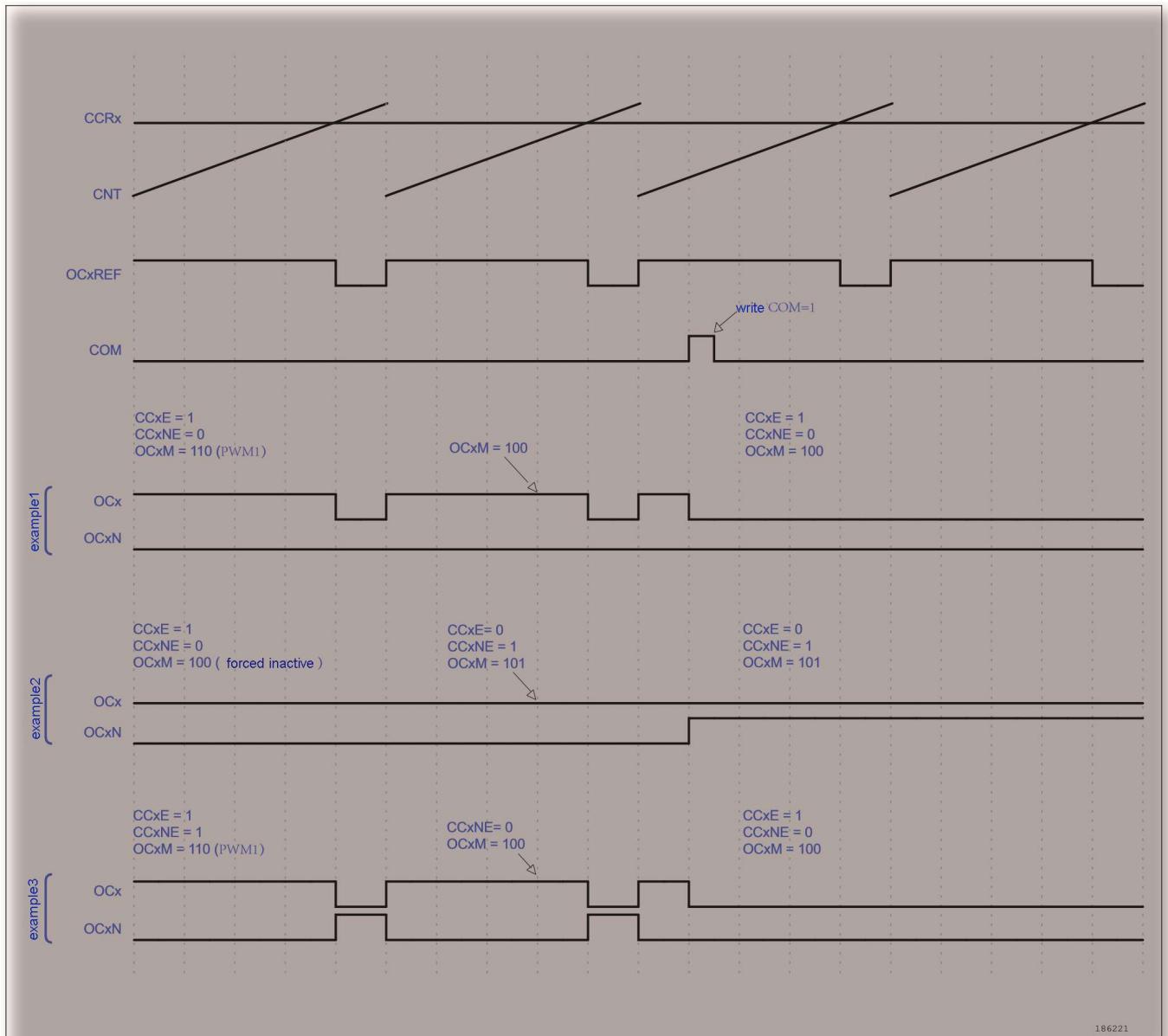


Figure 81. 6-step generation, COM example (OSSR=1)

### 15.3.15 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select One-pulse mode by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

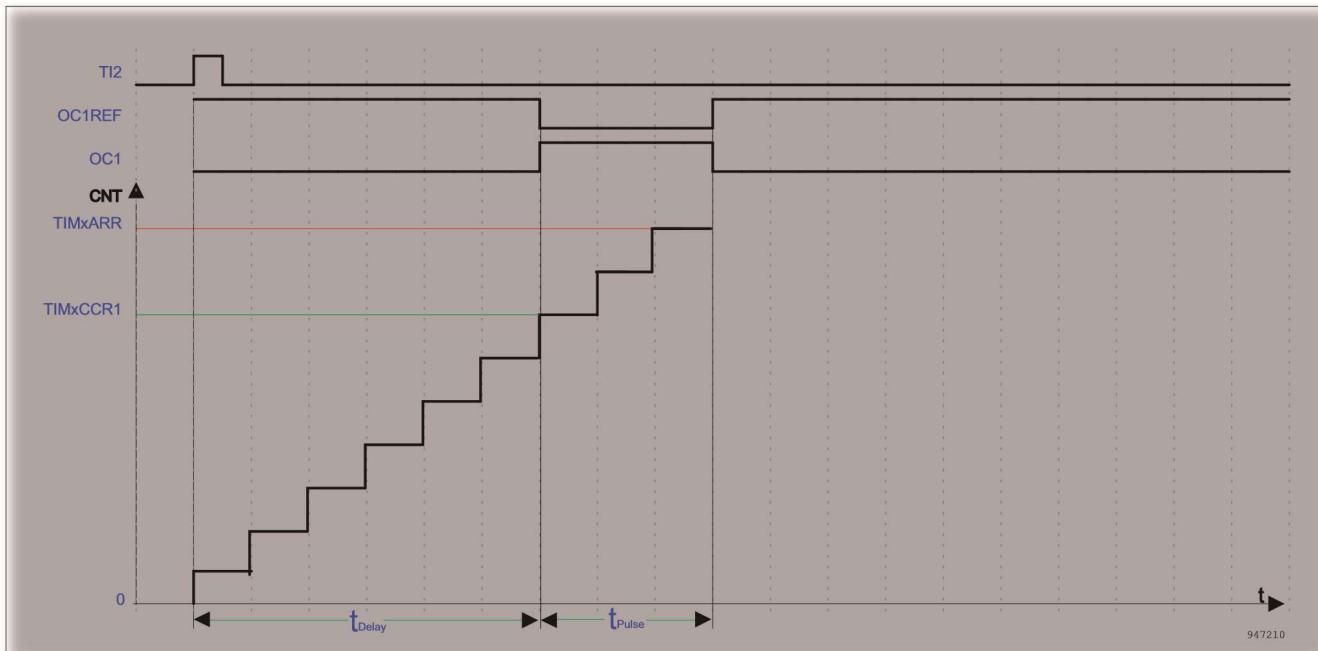


Figure 82. Example of one pulse mode

For example the user may want to generate a positive pulse on OC1 with a length of  $t_{\text{PULSE}}$  and after a delay of  $t_{\text{DELAY}}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S to '01' in the TIMx\_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P to '0' in the TIMx\_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS to '110' in the TIMx\_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{\text{DELAY}}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{\text{PULSE}}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let us say the user wants to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value.

To do this, enable PWM mode 2 by writing OC1M = 111 in the TIMx\_CCMR1 register. The user can optionally enable the preload registers by writing OC1PE = 1 in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case the compare value must be written in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '1' in this example.

In our example, the DIR and CMS bits in the TIMx\_CR1 register should be low.

The user only wants one pulse, so '1' must be written in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

#### Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TI<sub>x</sub> input sets the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{DELAY}$  min we can get.

If the user wants to output a waveform with the minimum delay, the OC<sub>x</sub>FE bit in the TIM<sub>x</sub>\_CCMR<sub>x</sub> register must be set. Then OC<sub>x</sub>Ref (and OC<sub>x</sub>) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OC<sub>x</sub>FE acts only if the channel is configured in PWM1 or PWM2 mode.

### 15.3.16 Encoder interface mode

To select Encoder Interface mode, write SMS to '001' in the TIM<sub>x</sub>\_SMCR register if the counter is counting on TI2 edges only, SMS to '010' if it is counting on TI1 edges only and SMS to '011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIM<sub>x</sub>\_CCER register. When needed, the user can program the input filter as well. The two inputs TI1 and TI2 are used to interface to an incremental encoder.

Refer to Table 44. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM<sub>x</sub>\_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence, the counter counts up or down, the DIR bit in the TIM<sub>x</sub>\_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIM<sub>x</sub>\_ARR register (0 to ARR or ARR down to 0 depending on the direction). So user must configure TIM<sub>x</sub>\_ARR before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The following table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 44 Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset. The following figure gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points.

For this example we assume that the configuration is the following:

- CC1S = '01' (TIMx\_CCMR1 register, IC1FP1 mapped on TI1).
- CC2S = '01' (TIMx\_CCMR2 register, IC2FP2 mapped on TI2).
- CC1P = '0' (TIMx\_CCER register, IC1FP1 non-inverted, IC1FP1 = TI1).
- C2P = '0' (TIMx\_CCER register, IC2FP2 non-inverted, IC2FP2=TI2).
- SMS = '011' (TIMx\_SMCR register, both inputs are active on both rising and falling edges).
- CEN = '1' (TIMx\_CR1 register, counter enabled).

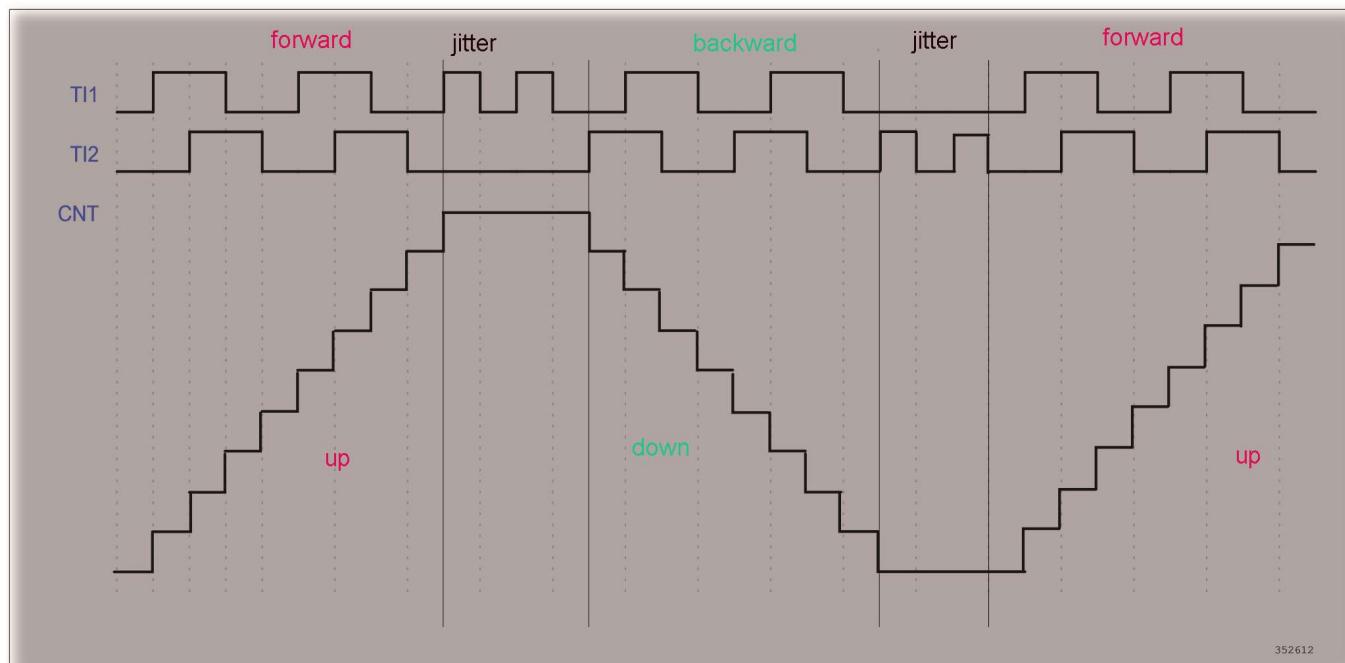


Figure 83. Example of counter operation in encoder interface mode

The following figure gives an example of counter behavior when IC1FP1 polarity is inverted (same configuration as above except CC1P = '1').

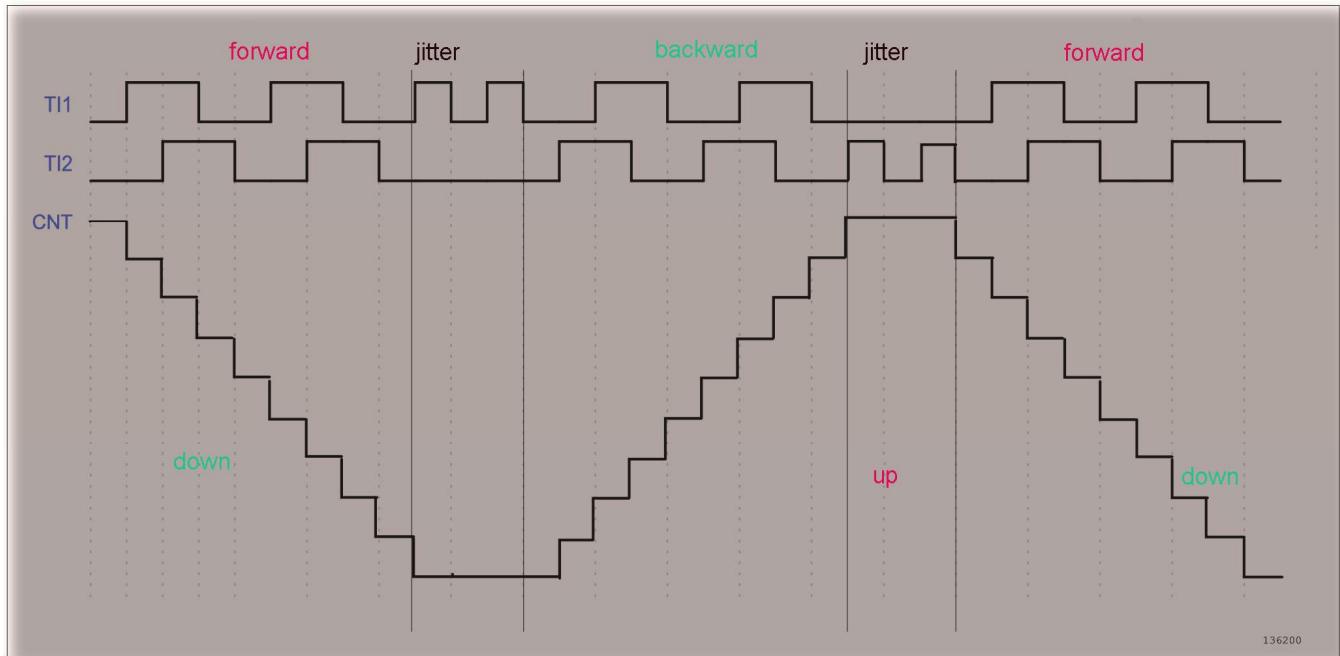


Figure 84. Example of encoder interface mode with IC1FP1 polarity inverted

The timer, when configured in Encoder Interface mode provides information on the sensor's current position. The user can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). It is also possible to read its value through a DMA request generated by a real-time clock.

### 15.3.17 Timer input XOR function

The TI1S bit in the TIMx\_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx\_CH1, TIMx\_CH2 and TIMx\_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. An example of this feature used to interface Hall sensors is given in Section 15.3.18.

### 15.3.18 Interfacing with Hall sensors

This is done using the advanced-control timer (TIM1) to generate PWM signals to drive the motor and another general-purpose timer TIMx (TIM2 or TIM3) referred to as “interfacing timer” in Figure 85.

The “interfacing timer” captures the 3 timer input pins (CC1, CC2, and CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx\_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F\_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (see Figure 67). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the “interfacing timer”

channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1) through the TRGO output.

Example: the user wants to change the PWM configuration of the advanced-control timer TIMx after a programmed delay each time a change occurs on the Hall inputs connected to the TIMx timer.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx\_CR2 register to '1'.
- Program the time base: write the TIMx\_ARR to the max value (the counter must be cleared by the TI1 change). Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors.
- Program channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx\_CCMR1 register to '01'. The user can also program the digital filter if needed.
- Program channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to '111' and the CC2S bits to '00' in the TIMx\_CCMR1 register.
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx\_CR2 register to '101'.

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIMx\_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIMx\_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The following figure describes this example:

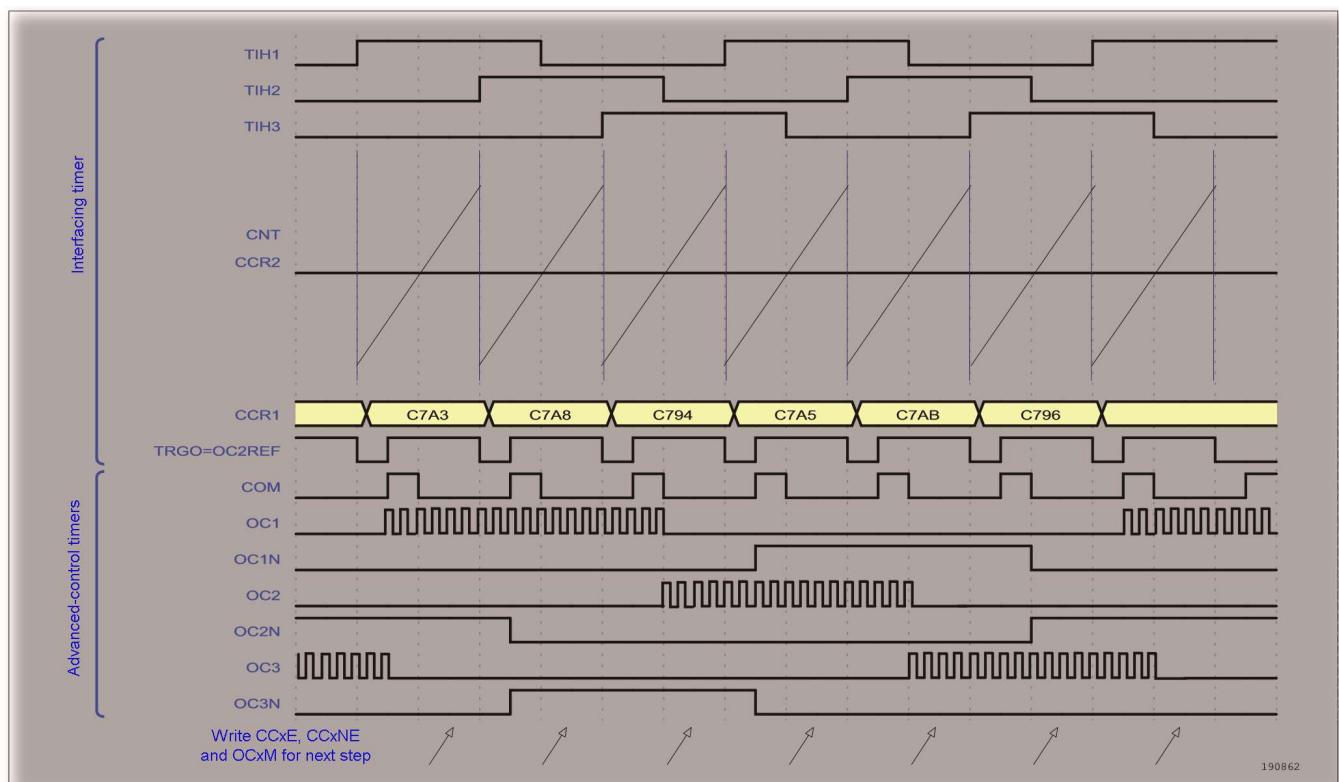


Figure 85. Example of Hall sensor interface

### 15.3.19 TIMx and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

## Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input.

Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC1S bits select the input capture source only, i.e. CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P = 0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS = 101 in TIMx\_SMCR register.
- Start the counter by writing CEN = 1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR = 0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

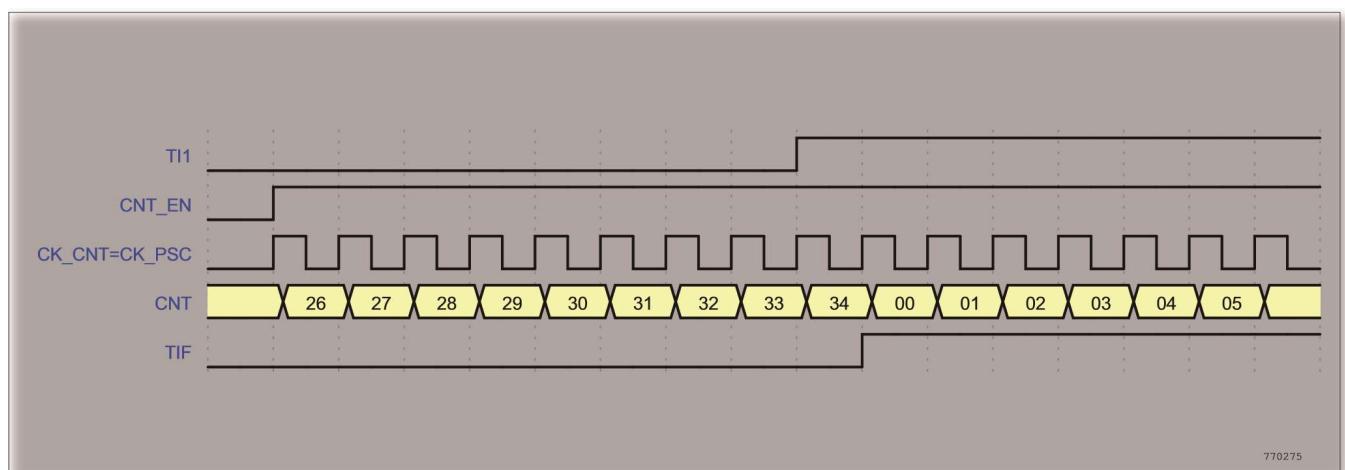


Figure 86. Control circuit in reset mode

## Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so the user does not need to configure it. The CC1S bits select the input capture source only. Write CC1S = 01 in TIMx\_CCMR1 register. Write CC1P = 1 in TIMx\_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS = 101 in TIMx\_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

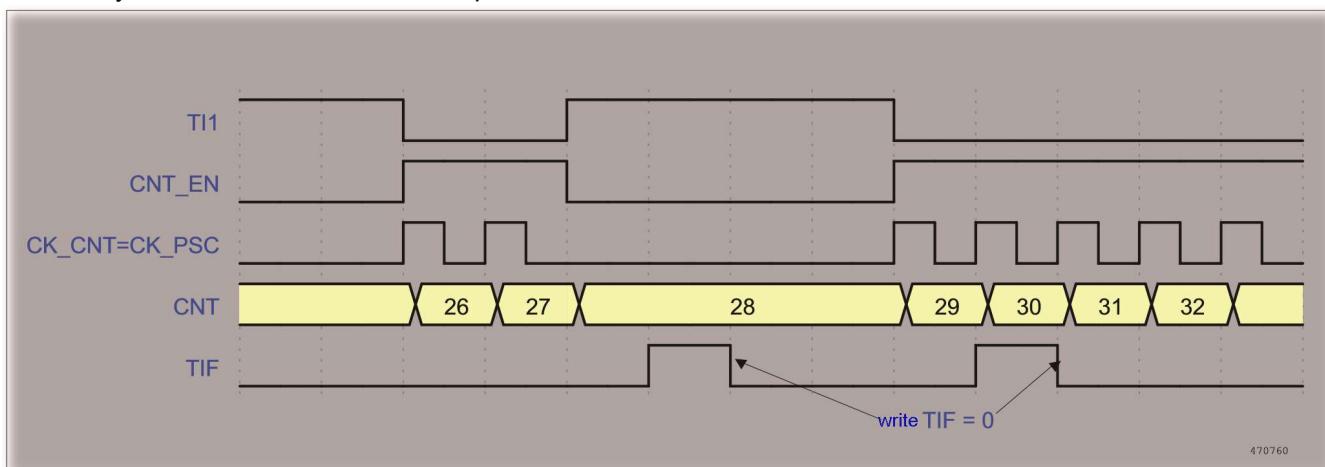


Figure 87. Control circuit in gated mode

### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC2S bits are configured to select the input capture source only. Write CC2P = 1 to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIMx\_SMCR register. Select TI2 as the input source by writing TS = 110 in TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set. The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

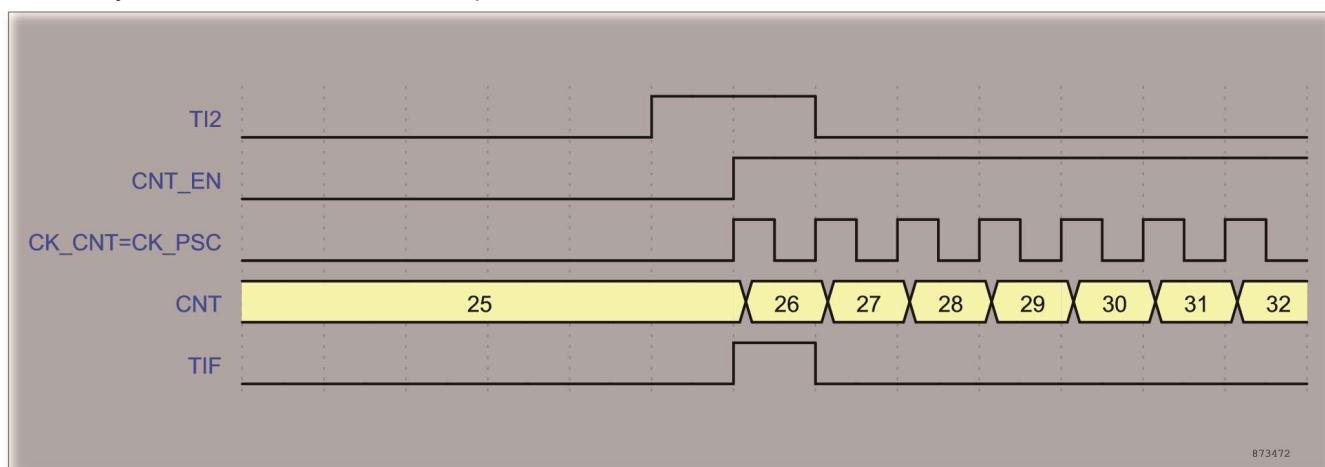


Figure 88. Control circuit in trigger mode

### Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx\_SMCR register. In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

- Configure the external trigger input circuit by programming the TIMx\_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS = 00: prescaler disabled
  - ETP = 0: detection of rising edges on ETR and ECE = 1 to enable the external clock mode 2.
- Configure the channel 1 as follows, to detect rising edges on TI:
  - IC1F = 0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S = 01 in TIMx\_CCMR1 register to select only the input capture source.
  - Write CC1P = 0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in trigger mode by writing SMS = 110 in the TIMx\_SMCR register. Select TI1 as the input source by writing TS = 101 in the TIMx\_SMCR register.

A rising edge on TI1 sets the TIF flag. The counter then counts on ETR rising edges. The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

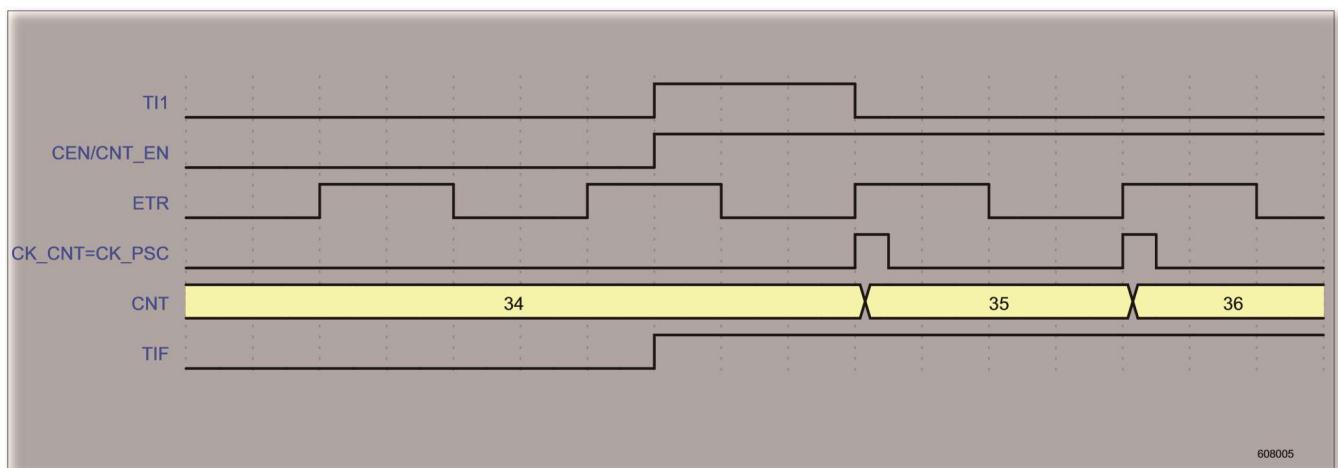


Figure 89. Control circuit in external clock mode 2 + trigger mode

### 15.3.20 Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining. Refer to Section TIMx for details.

### 15.3.21 Debug mode

When the microcontroller enters debug mode (CPU core halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIMx\_STOP configuration bit in DBG module. For more details, refer to subsequent debug section.

## 15.4 Register description

Table 45. Overview of TIM1 registers

Offset	Acronym	Register Name	Reset	Section
--------	---------	---------------	-------	---------

0x00	TIMx_CR1	Control register 1	0x00000000	Section 15.4.1
0x04	TIMx_CR2	Control register 2	0x00000000	Section 15.4.2
0x08	TIMx_SMCR	Slave mode control register	0x00000000	Section 15.4.3
0x0C	TIMx_DIER	DMA/interrupt enable register	0x00000000	Section 15.4.4
0x10	TIMx_SR	Status register	0x00000000	Section 15.4.5
0x14	TIMx_EGR	Event generate register	0x00000000	Section 15.4.6
0x18	TIMx_CCMR1	Capture/compare mode register 1	0x00000000	Section 15.4.7
0x1C	TIMx_CCMR2	Capture/compare mode register 2	0x00000000	Section 15.4.8
0x20	TIMx_CCER	Capture/compare enable register	0x00000000	Section 15.4.9
0x24	TIMx_CNT	Counter	0x00000000	Section 15.4.10
0x28	TIMx_PSC	Prescaler	0x00000000	Section 15.4.11
0x2C	TIMx_ARR	Auto-reload register	0x00000000	Section 15.4.12
0x30	TIMx_RCR	Repetition counter register	0x00000000	Section 15.4.13
0x34	TIMx_CCR1	Capture/compare register 1	0x00000000	Section 15.4.14
0x38	TIMx_CCR2	Capture/compare register 2	0x00000000	Section 15.4.15
0x3C	TIMx_CCR3	Capture/compare register 3	0x00000000	Section 15.4.16
0x40	TIMx_CCR4	Capture/compare register 4	0x00000000	Section 15.4.17

Offset	Acronym	Register Name	Reset	Section
0x44	TIMx_BDTR	Brake and dead-time register	0x00000000	Section 15.4.18
0x48	TIMx_DCR	DMA control register	0x00000000	Section 15.4.19
0x4C	TIMx_DMAR	DMA address for full transfer	0x00000000	Section 15.4.20
0x54	TIMx_CCMR3	Capture/compare mode register 3	0x00000000	Section 15.4.21
0x58	TIMx_CCR5	Capture/compare register 5	0x00000000	Section 15.4.22
0x5C	TIMx_PDER	PWM phase-shift/DMA repeat update request enable register	0x00000000	Section 15.4.23
0x60 ~ 0x70	CCRxFALL	PWM phase-shift downcounting capture/compare register	0x00000000	Section 15.4.24

### 15.4.1 Control register 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD	ARPE	CMS	DIR	OPM	URS	UDIS	CEN		

Bit	Field	Type	Reset	Description
15:10	Reserved			Reserved, always read as 0
9:8	CKD	rw	0x00	Clock division These 2 bits indicate the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock used by the dead-time generators and the digital filters (ETR, TIx). 00: $t_{DTS} = t_{CK\_INT}$ 01: $t_{DTS} = 2 \times t_{CK\_INT}$ 10: $t_{DTS} = 4 \times t_{CK\_INT}$ 11: Reserved, do not program this value
7	ARPE	rw	0x00	Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered

Bit	Field	Type	Reset	Description
6:5	CMS	rw	0x00	<p>Center-aligned mode selection</p> <p>00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).</p> <p>01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.</p> <p>10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.</p> <p>11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.</p> <p>Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1). See center-aligned PWM waveforms for details.</p>
4	DIR	rw	0x00	<p>Direction</p> <p>0: Counter used as upcounter</p> <p>1: Counter used as downcounter</p> <p>Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.</p>
3	OPM	rw	0x00	<p>One pulse mode</p> <p>0: Counter is not stopped at update event.</p> <p>1: Counter stops counting at the next update event (clearing the bit CEN).</p>
2	URS	rw	0x00	<p>Update request source</p> <p>This bit is set by software to select the UEV event sources.</p> <p>0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:</p> <ul style="list-style-type: none"> <li>- Counter overflow/underflow</li> <li>- Setting the UG bit</li> <li>- Update generation through the slave mode controller</li> </ul> <p>1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.</p>

Bit	Field	Type	Reset	Description

1	UDIS	rw	0x00	Update disable This bit is set by software to enable/disable UEV event generation. 0: UEV enabled. The Update (UEV) event is generated by one of the following events: - Counter overflow/underflow - Setting the UG bit to '1' - Update generation through the slave mode controller with the subsequent update of shadow registers 1: UEV disabled. The Update event is not generated, shadow registers keep their values (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set to '1' or if a hardware reset is received from the slave mode controller.
0	CEN	rw	0x00	Counter enable 0: Counter disabled 1: Counter enabled Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

### 15.4.2 Control register 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															OIS5
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS	CCDS	CCUS	Res.	CCPC		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31: 17	Reserved			Reserved, always read as 0.
16	OIS5	rw	0x00	Output Idle state 5 (OC5 output). Refer to OIS1 bit.
15	Reserved			Reserved and always read as 0.
14	OIS4	rw	0x00	Output Idle state 4 (OC4 output). Refer to OIS1 bit.
13	OIS3N	rw	0x00	Output Idle state 3 (OC3N output). Refer to OIS1N bit.
12	OIS3	rw	0x00	Output Idle state 3 (OC3 output). Refer to OIS1 bit.
11	OIS2N	rw	0x00	Output Idle state 2 (OC2N output). Refer to OIS1N bit.

Bit	Field	Type	Reset	Description
10	OIS2	rw	0x00	Output Idle state 2 (OC2 output). Refer to OIS1 bit.

## Advanced-Control Timer (TIM1)

UM MM32F013x Ver1.00

9	OIS1N	rw	0x00	<p>Output Idle state 1 (OC1N output)</p> <p>0: OC1N = 0 after a dead-time when MOE = 0</p> <p>1: OC1N = 1 after a dead-time when MOE = 0</p> <p>Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).</p>
8	OIS1	rw	0x00	<p>Output Idle state 1 (OC1 output)</p> <p>0: OC1 = 0 (after a dead-time if OC1N is implemented) when MOE = 0</p> <p>1: OC1 = 1 (after a dead-time if OC1N is implemented) when MOE = 0</p> <p>Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).</p>
7	TI1S	rw	0x00	<p>TI1 selection</p> <p>0: The TIMx_CH1 pin is connected to TI1 input</p> <p>1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)</p>

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

6:4	MMS	rw	0x00	<p>Master mode selection</p> <p>These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:</p> <ul style="list-style-type: none"> <li>000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.</li> <li>001: Enable - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).</li> <li>010: Update - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.</li> <li>011: Compare Pulse - The trigger output (TRGO) send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.</li> <li>100: Compare - OC1REF signal is used as trigger output (TRGO)</li> <li>101: Compare - OC2REF signal is used as trigger output (TRGO)</li> <li>110: Compare - OC3REF signal is used as trigger output (TRGO)</li> <li>111: Compare - OC4REF signal is used as trigger output (TRGO)</li> </ul>
3	CCDS	rw	0x00	<p>Capture/compare DMA selection</p> <p>0: CCx DMA request sent when CCx event occurs 1: CCx DMA requests sent when update event occurs</p>
2	CCUS	rw	0x00	<p>Capture/compare control update selection</p> <p>0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COM bit only 1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COM bit or when a rising edge occurs on TRGI Note: This bit acts only on channels that have a complementary output.</p>
1	Reserved			Reserved and always read as 0.

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

0	CCPC	rw	0x00	Capture/compare preloaded control 0: CCxE, CCxNE and OCxM bits are not preloaded 1: CCxE, CCxNE and OCxM bits are preloaded; after having been written, they are updated only when the COM bit is set. Note: This bit acts only on channels that have a complementary output.
---	------	----	------	--

### 15.4.3 Slave mode control register (TIMx\_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF		MSM		TS		OCCS		SMS			

Bit	Field	Type	Reset	Description
15	ETP	rw	0x00	External trigger polarity This bit selects whether ETR or inverted ETR is used for trigger operations. 0: ETR is non-inverted, active at high level or rising edge 1: ETR is inverted, active at low level or falling edge
14	ECE	rw	0x00	External clock enable This bit enables External clock mode 2. 0: External clock mode 2 disabled 1: External clock mode 2 enabled. The counter is clocked by any active rising edge on the ETRF signal. Note 1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS = 111 and TS = 111). Note 2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111). Note 3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.
13:12	ETPS	rw	0x00	External trigger prescaler External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks. 00: Prescaler OFF 01: ETRP frequency divided by 2 10: ETRP frequency divided by 4 11: ETRP frequency divided by 8
Bit	Field	Type	Reset	Description

11:8	ETF	rw	0x00	<p>External trigger filter</p> <p>This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:</p> <ul style="list-style-type: none"> <li>0000: No filter, sampling is done at <math>f_{DTS}</math></li> <li>0001: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 2</li> <li>0010: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 4</li> <li>0011: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 8</li> <li>0100: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 6</li> <li>0101: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 8</li> <li>0110: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 6</li> <li>0111: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 8</li> <li>1000: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 6</li> <li>1001: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 8</li> <li>1010: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 5</li> <li>1011: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 6</li> <li>1100: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 8</li> <li>1101: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 5</li> <li>1110: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 6</li> <li>1111: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 8</li> </ul>
7	MSM	rw	0x00	<p>Master/slave mode</p> <p>0: No action</p> <p>1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.</p>
6:4	TS	rw	0x00	<p>Trigger selection</p> <p>This bit-field selects the trigger input to be used to synchronize the counter.</p> <ul style="list-style-type: none"> <li>000: Internal Trigger 0 (ITR0)</li> <li>001: Internal Trigger 1 (ITR1)</li> <li>010: Internal Trigger 2 (ITR2)</li> <li>011: Internal Trigger 3 (ITR3)</li> <li>100: TI1 Edge Detector (TI1F_ED)</li> <li>101: Filtered Timer Input 1 (TI1FP1)</li> <li>110: Filtered Timer Input 2 (TI2FP2)</li> <li>111: External Trigger input (ETRF)</li> </ul> <p>See the table below for more details on ITRx.</p> <p>Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.</p>

Bit	Field	Type	Reset	Description
3	OCCS	rw	0x00	Output compare clear selection

This bit is used to clear the comparator output in the PWM mode.

1: The comparator output is considered as the clear signal

0: The external trigger input is considered as the clear signal.

2:0	SMS	rw	0x00	Slave mode selection  When external signals are selected, the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control register description).  000: Slave mode disabled - if CEN = 1, then the prescaler is clocked directly by the internal clock. 001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level. 010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level. 011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input. 100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers. 101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled. 110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled. 111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.  Note: The gated mode must not be used if TI1F_EN is selected as the trigger input (TS = 100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.
-----	-----	----	------	---

Table 46. TIMx internal trigger connection

Slave TIM	ITR0	ITR1	ITR2	ITR3
TIM1	None	TIM2	TIM3	None
TIM2	TIM1	None	TIM3	None
TIM3	TIM1	TIM2	None	None

#### 15.4.4DMA/Interrupt enable register (TIMX\_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														CC5 DE	CC5 IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COM DE	CC4 DE	CC3 DE	CC2 DE	CC1 DE	UDE	BIE	TIE	COM IE	CC4 IE	CC3 IE	CC2 IE	CC1 IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:18	Reserved			Reserved, always read as 0.
17	CC5DE	rw	0x00	Capture/Compare 5 DMA request enable 0: Capture/Compare 5 DMA request disabled 1: Capture/Compare 5 DMA request enabled
16	CC5IE	rw	0x00	Capture/Compare 5 interrupt enable 0: Capture/Compare 5 interrupt disabled 1: Capture/Compare 5 interrupt enabled
15	Reserved			Reserved and always read as 0.
14	TDE	rw	0x00	Trigger DMA request enable 0: Trigger DMA request disabled 1: Trigger DMA request enabled
13	COMDE	rw	0x00	COM DMA request enable 0: COM DMA request disabled 1: COM DMA request enabled
12	CC4DE	rw	0x00	Capture/Compare 4 DMA request enable 0: Capture/Compare 4 DMA request disabled 1: Capture/Compare 4 DMA request enabled
11	CC3DE	rw	0x00	Capture/Compare 3 DMA request enable 0: Capture/Compare 3 DMA request disabled 1: Capture/Compare 3 DMA request enabled
10	CC2DE	rw	0x00	Capture/Compare 2 DMA request enable 0: Capture/Compare 2 DMA request disabled 1: Capture/Compare 2 DMA request enabled
9	CC1DE	rw	0x00	Capture/Compare 1 DMA request enable 0: Capture/Compare 1 DMA request disabled 1: Capture/Compare 1 DMA request enabled
8	UDE	rw	0x00	Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled

Bit	Field	Type	Reset	Description
7	BIE	rw	0x00	Break interrupt enable 0: Break interrupt disabled 1: Break interrupt enabled

### Advanced-Control Timer (TIM1)

UM MM32F013x Ver1.00

6	TIE	rw	0x00	Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
5	COMIE	rw	0x00	COM interrupt enable 0: COM interrupt disabled 1: COM interrupt enabled
4	CC4IE	rw	0x00	Capture/Compare 4 interrupt enable 0: Capture/Compare 4 interrupt disabled 1: Capture/Compare 4 interrupt enabled
3	CC3IE	rw	0x00	Capture/Compare 3 interrupt enable 0: Capture/Compare 3 interrupt disabled 1: Capture/Compare 3 interrupt enabled
2	CC2IE	rw	0x00	Capture/Compare 2 interrupt enable 0: Capture/Compare 2 interrupt disabled 1: Capture/Compare 2 interrupt enabled
1	CC1IE	rw	0x00	Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt disabled 1: Capture/Compare 1 interrupt enabled
0	UIE	rw	0x00	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

#### 15.4.5 Status register (TIMx\_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved															CC5 IF	

rc\_w0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved			CC4 OF	CC3 OF	CC2 OF	CC1 OF	Res.	BIF	TIF	COM IF	CC4 IF	CC3 IF	CC2 IF	CC1 IF	UIF	

rc\_w0 rc\_w0

Bit	Field	Type	Reset	Description
31:17	Reserved			Reserved, always read as 0
16	CC5IF	rc_w0	0x00	Capture/Compare 5 interrupt flag Refer to CC1IF description.
15:13	Reserved			Reserved and always read as 0.

12	CC4OF	rc_w0	0x00	Capture/Compare 4 overcapture flag Refer to CC1OF description.
11	CC3OF	rc_w0	0x00	Capture/Compare 3 overcapture flag Refer to CC1OF description.
10	CC2OF	rc_w0	0x00	Capture/Compare 2 overcapture flag refer to CC1OF description.
9	CC1OF	rc_w0	0x00	Capture/Compare 1 overcapture flag This flag is set to '1' by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'. 0: No overcapture has been detected. 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set to 1.
8	Reserved			Reserved and always read as 0.
7	BIF	rc_w0	0x00	Break interrupt flag This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active. 0: No break event occurred. 1: An active level has been detected on the break input.
6	TIF	rc_w0	0x00	Trigger interrupt flag This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software. 0: No trigger event occurred. 1: Trigger interrupt pending.
5	COMIF	rc_w0	0x00	COM interrupt flag This flag is set by hardware on COM event (when capture/compare control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software. 0: No COM event occurred 1: COM interrupt pending.

Bit	Field	Type	Reset	Description
4	CC4IF	rc_w0	0x00	Capture/Compare 4 interrupt flag Refer to CC1IF description.
3	CC3IF	rc_w0	0x00	Capture/Compare 3 interrupt flag Refer to CC1IF description.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
2		CC2IF		rc_w0		0x00		Capture/Compare 2 interrupt flag Refer to CC1IF description.											
1		CC1IF		rc_w0		0x00		Capture/Compare 1 interrupt flag If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software. 0: No match. 1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register. 0: No input capture occurred 1: The counter value has been captured in (copied to) TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)											
0		UIF		rc_w0		0x00		Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update occurred. 1: Update event pending. This bit is set by hardware when the registers are updated: - At overflow or underflow regarding the repetition downcounter value (update if REP_CNT = 0) and if the UDIS = 0 in the TIMx_CR1 register. - When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register. - When CNT is reinitialized by a trigger event (refer to the synchronous control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.											

#### 15.4.6 Event generate register (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	W	W	W	W	W	W	W	W

Bit	Field	Type	Reset	Description
31:17	Reserved			Reserved, always read as 0
16	CC5G	w	0x00	Capture/Compare 5 generation Refer to CC1G description.
15:8	Reserved			Reserved and always read as 0.
7	BG	w	0x00	Break generation  This bit is set by software in order to generate a brake event, it is automatically cleared by hardware. 0: No action 1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA can occur if enabled.
6	TG	w	0x00	Trigger generation  This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA can occur if enabled.
5	COMG	w	0x00	Capture/Compare control update generation  This bit can be set by software, it is automatically cleared by hardware. 0: No action 1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits Note: This bit acts only on channels that have a complementary output.
4	CC4G	w	0x00	Capture/Compare 4 generation Refer to CC1G description.
3	CC3G	w	0x00	Capture/Compare 3 generation Refer to CC1G description.
2	CC2G	w	0x00	Capture/Compare 2 generation Refer to CC1G description.
Bit	Field	Type	Reset	Description

1	CC1G	w	0x00	Capture/Compare 1 generation  This bit is set by software in order to generate a capture/compare event, it is automatically cleared by hardware.  0: No action 1: A capture/compare event is generated on channel CC1: If channel CC1 is configured as output: CC1IF flag is set, corresponding interrupt or DMA request is sent if enabled. If channel CC1 is configured as input: The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.
0	UG	w	0x00	Update generation  This bit can be set by software, it is automatically cleared by hardware.  0: No action 1: Reinitialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR = 1 (downcounting).

#### 15.4.7 Capture/compare mode register 1 (TIMx\_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So the user must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
OC2 CE	OC2M		OC2 PE	OC2 FE	CC2S	OC1 CE	OC1M		OC1 PE	OC1 FE	CC1S					
IC2F		IC2PSC				IC1F		IC1PSC								
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

#### Output Compare mode:

Bit	Field	Type	Reset	Description
15	OC2CE	rw	0x00	Output compare 2 clear enable
14:12	OC2M	rw	0x00	Output compare 2 mode
11	OC2PE	rw	0x00	Output compare 2 preload enable
10	OC2FE	rw	0x00	Output compare 2 fast enable
9:8	CC2S	rw	0x00	Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register). Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).
7	OC1CE	rw	0x00	Output compare 1 clear enable 0: OC1REF is not affected by the ETRF input 1: OC1REF is cleared as soon as a high level is detected on ETRF input

Bit	Field	Type	Reset	Description
6:4	OC1M	rw	0x00	<p>Output compare 1 mode</p> <p>These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.</p> <p>000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the OC1REF.</p> <p>001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).</p> <p>010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).</p> <p>011: Toggle. OC1REF level toggles when TIMx_CCR1=TIMx_CNT.</p> <p>100: Force inactive level. OC1REF is forced low.</p> <p>101: Force active level. OC1REF is forced high.</p> <p>110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT &lt; TIMx_CCR1, else inactive. In downcounting, channel 1 is inactive (OC1REF = 0) as long as TIMx_CNT &gt; TIMx_CCR1, else active (OC1REF = 1).</p> <p>111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT &lt; TIMx_CCR1, else active. In downcounting, channel 1 is active as long as TIMx_CNT &gt; TIMx_CCR1, else inactive.</p> <p>Note 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).</p> <p>Note 2: In PWM mode 1 or 2, the OC1REF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.</p>
3	OC1PE	rw	0x00	<p>Output compare 1 preload enable</p> <p>0: Preload function of TIMx_CCR1 register disabled. TIMx_CCR1 register can be written at anytime, the new value is taken in account immediately.</p> <p>1: Preload function of TIMx_CCR1 register enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.</p> <p>Note 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).</p> <p>Note 2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.</p>

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

Advanced-Control Timer (TIM1)				UM_MM32F013x_Ver1.00
2	OC1FE	rw	0x00	<p>Output compare 1 fast enable</p> <p>This bit is used to accelerate the effect of an event on the trigger in input on the CC output.</p> <p>0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.</p> <p>1: An active edge on the trigger input acts like a compare match. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles.</p> <p>OCFE acts only if the channel is configured in PWM1 or PWM2 mode.</p>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC1 channel is configured as output</li> <li>01: CC1 channel is configured as input, IC1 is mapped on TI1</li> <li>10: CC1 channel is configured as input, IC1 is mapped on TI2</li> <li>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)</li> </ul> <p>Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).</p>

### Input capture mode:

Bit	Field	Type	Reset	Description
15 : 12	IC2F	rw	0x00	Input capture 2 filter
11:10	IC2PSC	rw	0x00	Input capture 2 prescaler
9:8	CC2S	rw	0x00	<p>Capture/Compare 2 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC2 channel is configured as output.</li> <li>01: CC2 channel is configured as input, IC2 is mapped on TI2.</li> <li>10: CC2 channel is configured as input, IC2 is mapped on TI1.</li> <li>11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register).</li> </ul> <p>Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).</p>

7:4	IC1F	rw	0x00	<p>Input capture 1 filter</p> <p>This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to generate a transition on the output:</p> <ul style="list-style-type: none"> <li>0000: No filter, sampling is done at <math>f_{DTS}</math></li> <li>1000: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 6</li> <li>0001: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 2</li> <li>1001: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 8</li> <li>0010: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 4</li> <li>1010: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 5</li> <li>0011: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 8</li> <li>1011: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 6</li> <li>0100: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 6</li> <li>1100: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 8</li> <li>0101: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 8</li> <li>1101: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 5</li> <li>0110: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 6</li> <li>1110: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 6</li> <li>0111: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 8</li> <li>1111: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 8</li> </ul>
3:2	IC1PSC	rw	0x00	<p>Input capture 1 prescaler</p> <p>This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).</p> <p>The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register). 00: no prescaler, capture is done each time an edge is detected on the capture input</p> <ul style="list-style-type: none"> <li>01: capture is done once every 2 events</li> <li>10: capture is done once every 4 events</li> <li>11: capture is done once every 8 events</li> </ul>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC1 channel is configured as output</li> <li>01: CC1 channel is configured as input, IC1 is mapped on TI1</li> <li>10: CC1 channel is configured as input, IC1 is mapped on TI2</li> <li>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)</li> </ul> <p>Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).</p>

#### 15.4.8 Capture/compare mode register 2 (TIMx\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
OC4 CE	OC4M		OC4 PE	OC4 FE	CC4S	OC3 CE	OC3M			OC3 PE	OC3 FE	CC3S	IC3PSC	IC3F	IC4PSC	IC4F

## Output Compare mode

Bit	Field	Type	Reset	Description
15	OC4CE	rw	0x00	Output compare 4 clear enable
14:12	OC4M	rw	0x00	Output compare 4 mode
11	OC4PE	rw	0x00	Output compare 4 preload enable
10	OC4FE	rw	0x00	Output compare 4 fast enable
9:8	CC4S	rw	0x00	Capture/Compare 4 selection  This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC4 channel is configured as output 01: CC4 channel is configured as input, IC4 is mapped on TI4 10: CC4 channel is configured as input, IC4 is mapped on TI3 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).
7	OC3CE	rw	0x00	Output compare 3 clear enable
6:4	OC3M	rw	0x00	Output compare 3 mode
3	OC3PE	rw	0x00	Output compare 3 preload enable
2	OC3FE	rw	0x00	Output compare 3 fast enable
1:0	CC3S	rw	0x00	Capture/Compare 3 selection  This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI3 10: CC3 channel is configured as input, IC3 is mapped on TI4 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

## Input Compare mode

Bit	Field	Type	Reset	Description
15:12	IC4F	rw	0x00	Input capture 4 filter
11:10	IC4PSC	rw	0x00	Input capture 4 prescaler
9:8	CC4S	rw	0x00	Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC4 channel is configured as output 01: CC4 channel is configured as input, IC4 is mapped on TI4 10: CC4 channel is configured as input, IC4 is mapped on TI3 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).
7:4	IC3F	rw	0x00	Input capture 3 filter
3:2	IC3PSC	rw	0x00	Input capture 3 prescaler
1:0	CC3S	rw	0x00	Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI3 10: CC3 channel is configured as input, IC3 is mapped on TI4 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

### 15.4.9 Capture/compare enable register (TIMx\_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved														CC5 P	CC5 E	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	CC4P	CC4E	CC3 NP	CC3 NE	CC3P	CC3E	CC2 NP	CC2 NE	CC2P	CC2E	CC1 NP	CC1 NE	CC1P	CC1 E	rw	rw

Bit	Field	Type	Reset	Description
31:18	Reserved			Reserved, always read as 0

## Advanced-Control Timer (TIM1)

UM MM32F013x Ver1.00

17	CC5P	rw	0x00	Capture/Compare 5 output polarity Refer to CC1P description.
16	CC5E	rw	0x00	Capture/Compare 5 output enable Refer to CC1E description.
15:14	Reserved		Reserved and always read as 0.	
13	CC4P rw 0x00 Capture/Compare 4 output polarity			Refer to CC1P description.
12	CC4E	rw	0x00	Capture/Compare 4 output enable Refer to CC1E description.
11	CC3NP	rw	0x00	Capture/Compare 3 complementary output polarity Refer to CC1NP description.
10	CC3NE	rw	0x00	Capture/Compare 3 complementary output enable Refer to CC1NE description.
9	CC3P rw 0x00 Capture/Compare 3 output polarity			Refer to CC1P description.
8	CC3E	rw	0x00	Capture/Compare 3 output enable Refer to CC1E description.
7	CC2NP	rw	0x00	Capture/Compare 2 complementary output polarity Refer to CC1NP description.
6	CC2NE	rw	0x00	Capture/Compare 2 complementary output enable Refer to CC1NE description.
5	CC2P rw 0x00 Capture/Compare 2 output polarity			Refer to CC1P description.
4	CC2E	rw	0x00	Capture/Compare 2 output enable Refer to CC1E description.
3	CC1NP	rw	0x00	Capture/Compare 1 complementary output polarity 0: OC1N active high 1: OC1N active low  Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit	Field	Type	Reset	Description

2	CC1NE	rw	0x00	<p>Capture/Compare 1 complementary output enable</p> <p>0: Off - OC1N is not active. OC1N level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.</p> <p>1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.</p>
1	CC1P	rw	0x00	<p>Capture/Compare 1 output polarity</p> <p>CC1 channel configured as output:</p> <p>0: OC1 active high 1: OC1 active low</p> <p>CC1 channel configured as input:</p> <p>This bit selects whether the inverted signal of IC1 or IC1 is used for trigger or capture operations.</p> <p>0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted.</p> <p>1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted.</p> <p>Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).</p>
0	CC1E	rw	0x00	<p>Capture/Compare 1 output enable</p> <p>CC1 channel configured as output:</p> <p>0: Off - OC1 is not active. OC1 level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits.</p> <p>1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits.</p> <p>CC1 channel configured as input:</p> <p>This bit determines if a capture of the counter value can actually be done into the TIMx_CCR1 register or not.</p> <p>0: capture disabled 1: capture enabled</p>

Table 47. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states <sup>(1)</sup>	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	0	0	0	Output Disabled (not driven by the timer), OCx = 0, OCx_EN = 0	Output Disabled (not driven by the timer), OCxN = 0, OCxN_EN = 0
		0	0	1	Output Disabled (not driven by the timer), OCx = 0, OCx_EN = 0	OCxREF + Polarity, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		0	1	0	OCxREF + Polarity, OCx = OCxREF xor CCxP, OCx_EN = 1	Output Disabled (not driven by the timer), OCxN = 0, OCxN_EN = 0
		0	1	1	OCxREF + Polarity + dead-time, OCx_EN = 1	Complementary to OCxREF + Polarity + dead-time, OCxN_EN = 1
		1	0	0	Output Disabled (not driven by the timer), OCx = CCxP, OCx_EN = 0	Output Disabled (not driven by the timer), OCxN = CCxNP, OCxN_EN = 0
		1	0	1	Off-State (output enabled with inactive state) OCx = CCxP, OCx_EN = 1	OCxREF + Polarity, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		1	1	0	OCxREF + Polarity, OCx = OCxREF xor CCxP, OCx_EN = 1	Off-State (output enabled with inactive state) OCxN = CCxNP, OCxN_EN = 1
		1	1	1	OCxREF + Polarity + dead-time, OCx_EN = 1	Complementary to OCxREF + Polarity + dead-time, OCxN_EN = 1
0	X	0	0	0	Output Disabled (not driven by the timer), Asynchronously: OCx = CCxP, OCx_EN = 0, OCxN = CCxNP, OCxN_EN = 0;	
		0	0	1	Then if the clock is present: OCx = OISx and OCxN = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state.	
		0	1	0		
		0	1	1		
		1	0	0		Off-State (output enabled with inactive state)
		1	0	1		Asynchronously: OCx = CCxP, OCx_EN = 1, OCxN = CCxNP, OCxN_EN = 1;
		1	1	0		Then if the clock is present: OCx = OISx and OCxN = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state.
		1	1	1		

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OC<sub>x</sub> and OC<sub>xN</sub> channels depends on the OC<sub>x</sub> and OC<sub>xN</sub> channel state and the GPIO registers.

#### 15.4.10 Counter (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w

Bit	Field	Type	Reset	Description
15:0	CNT	rw	0x0000	Counter value

#### 15.4.11 Prescaler (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description
15:0	PSC	rw	0x0000	<p>Prescaler value</p> <p>The counter clock frequency (CK_CNT) is equal to <math>f_{CK\_Psc}/(PSC + 1)</math>.</p> <p>PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through slave controller when configured in “reset mode”).</p>

#### 15.4.12 Auto-reload register (TIMx\_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description
15:0	ARR	rw	0x0000	<p>Auto-reload value</p> <p>ARR is the value to be loaded in the actual auto-reload register. Refer to Section 15.3.1: about ARR update and behavior.</p> <p>The counter is blocked while the auto-reload value is null.</p>

#### 15.4.13 Repetition counter register (TIMx\_RCR)

Address offset: 0x30

Reset value: 0x0000

Advanced-Control Timer (TIM1)										UM_MM32F013x_Ver1.00						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
REP_CNT								rw	REP							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:8	REP_CNT	rw	0x00	<p>Repetition counter value of real-time writing</p> <p>These bits are used in the repetition counter modification mode, to move the update interrupt flag (UIF) to left by (REP-REP_CNT) phases (to right when the subtracted value is negative) by shifting UIF detection point in real time. These bits should be written after the update event UG is generated (note writing to REP_CNT before the update event is generated makes the displacement invalid).</p>
7:0	RE_P	rw	0x00	<p>Repetition counter value</p> <p>These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enabled, as well as the update interrupt generation rate, if this interrupt is enabled.</p> <p>Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.</p> <p>It means in PWM mode, (REP+1) corresponds to:</p> <ul style="list-style-type: none"> <li>- the number of PWM periods in edge-aligned mode</li> <li>- the number of half PWM period in center-aligned mode.</li> </ul>

#### 15.4.14 Capture/Compare register 1 (TIMx\_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description

15:0	CCR1	rw	0x0000	Capture/Compare 1 value If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded immediately if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output. If channel CC1 is configured as input: CCR1 is the counter value transferred by the last input capture 1 event (IC1).
------	------	----	--------	--

#### 15.4.15 Capture/Compare register 2 (TIMx\_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:0	CCR2	rw	0x0000	Capture/Compare 2 value If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded immediately if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output. If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (IC2).

#### 15.4.16 Capture/compare register 3 (TIMx\_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
rw															

Bit	Field	Type	Reset	Description
15:0	CCR3	rw	0x0000	<p>Capture/Compare 3 value</p> <p>If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded immediately if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output.</p> <p>If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (IC3).</p>

#### 15.4.17 Capture/compare register 4 (TIMx\_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
rw															

Bit	Field	Type	Reset	Description
15:0	CCR4	rw	0x0000	<p>Capture/Compare 4 value</p> <p>If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded immediately if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC4 output.</p> <p>If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4).</p>

#### 15.4.18 Brake and dead-time register (TIMx\_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

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

Note: As the bits AOE, BKP, BKE, OSSR, OSSI and DTG can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

Bit	Field	Type	Reset	Description
31:17	Reserved			Reserved, always read as 0.
16	DOE	rw	0x00	<p>Direct output enable</p> <p>This bit is enabled when the break input is active and MOE is cleared.</p> <p>1: outputs in idle state directly, without any delay of dead-time</p> <p>0: when the brake input is generated, the outputs turn into idle state after a dead-time</p>
15	MOE	rw	0x00	<p>Main output enable</p> <p>This bit is cleared asynchronously by hardware as soon as the break input is active. It is cleared as '0' by software or automatically set to '1' depending on the AOE bit configuration. It is acting only on the channels which are configured in output.</p> <p>0: OC and OCN outputs are disabled or forced to idle state.</p> <p>1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).</p> <p>See OC/OCN enable description for more details (Section 15.4.9: capture/compare enable register (TIMx_CCER)).</p>
14	AOE	rw	0x00	<p>Automatic output enable</p> <p>0: MOE can be set only by software</p> <p>1: MOE can be set by software or automatically at the next update event (if the break input is not active)</p> <p>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</p>

Bit	Field	Type	Reset	Description
13	BKP	rw	0x00	<p>Break polarity</p> <p>0: Break input BRK is active low</p> <p>1: Break input BRK is active high</p> <p>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</p>
12	BKE	rw	0x00	<p>Break enable</p> <p>0: Break inputs (BRK and BRK_ACTH) disabled</p> <p>1: Break inputs (BRK and BRK_ACTH) enabled</p> <p>Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</p>
11	OSSR	rw	0x00	<p>Off-state selection for Run mode</p> <p>This bit is used when MOE = 1 on channels having a complementary output. OSSR is not implemented if no complementary output is implemented in the timer. See OC/OCN enable description for more details (Section 15.4.9: capture/compare enable register (TIMx_CCER)).</p> <p>0: When the timer is inactive, OC/OCN outputs are disabled (OC/OCN enable output signal = 0).</p> <p>1: When the timer is inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1. Then, OC/OCN enable output signal = 1.</p> <p>Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).</p>
10	OSSI	rw	0x00	<p>Off-state selection for Idle mode</p> <p>This bit is used when MOE = 0 on channels configured as outputs.</p> <p>See OC/OCN enable description for more details (Section 15.4.9: capture/compare enable register (TIMx_CCER)).</p> <p>0: When the timer is inactive, OC/OCN outputs are disabled (OC/OCN enable output signal = 0).</p> <p>1: When the timer is inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1. Then, OC/OCN enable output signal = 1</p> <p>Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).</p>

Bit	Field	Type	Reset	Description
9:8	LOCK	rw	0x00	<p>Lock configuration</p> <p>These bits offer a write protection against software errors.</p> <p>00: LOCK OFF - No bit is write protected.</p> <p>01: LOCK Level 1 = DTG, BKE, BKP and AOE bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register can no longer be written.</p> <p>10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.</p> <p>11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.</p> <p>Note: The LOCK bits can be written only once after the system reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.</p>
7:0	DTG	rw	0x00	<p>Dead-time generator setup</p> <p>This bit-field defines the duration of the dead-time inserted between the complementary outputs. Assuming that DT corresponds to this duration.</p> <p>DTG[7: 5] = 0xx:</p> $DT = (DTG[7: 0] + 1) \times t_{dtg}, t_{dtg} = t_{DTS}; DTG[7: 5] = 10x:$ $DT = (DTG[5: 0] + 1 + 64) \times t_{dtg}, t_{dtg} = 2 \times t_{DTS}; DTG[7: 5] = 110:$ $DT = (DTG[4: 0] + 1 + 32) \times t_{dtg}, t_{dtg} = 8 \times t_{DTS}; DTG[7: 5] = 111:$ $DT = (DTG[4: 0] + 1 + 32) \times t_{dtg}, t_{dtg} = 16 \times t_{DTS};$ <p>Example if <math>t_{DTS}=125\text{ns}</math> (8MHz), dead-time possible values are:</p> <p>125ns to 15875 ns by 125ns steps,      16μs to 31750ns by 250ns steps,      32μs to 63μs by 1μs steps,      64μs to 126μs by 2μs steps</p> <p>Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 have been programmed (LOCK bits in TIMx_BDTR register).</p>

### 15.4.19 DMA control register (TIMx\_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res.	DBL					Res.	DBA			
	W	W	W	W	W		W	W	W	W

Bit	Field	Type	Reset	Description
15:13	Reserved			Reserved, always read as 0
12:8	DBL	w	0x00	<p>DMA burst length This bit-field defines the length of DMA burst transfer (the timer recognizes a burst transfer when a write access to the TIMx_DMAR register is performed), ie. the number of transfers by half-words (double bytes) or bytes:</p> <ul style="list-style-type: none"> <li>00000: 1 transfer</li> <li>00001: 2 transfers</li> <li>00010: 3 transfers</li> <li>...</li> <li>10001: 18 transfers</li> </ul> <p>Example: Let us consider the following transfer: DBL = 7, DBA = TIM2_CR1</p> <ul style="list-style-type: none"> <li>- In this case, DBL = 7 and DBA = TIM2_CR1 represent the address to which the data is to be sent. Then the transfer address can be concluded from the following formula: <math>(\text{TIMx\_CR1 address}) + \text{DBA} + (\text{DMA index})</math>, where DMA index = DBL,</li> </ul> <p><math>(\text{TIMx\_CR1 address}) + \text{DBA} + 7</math> provides the address to/from which the transfer is done to/from 7 registers starting from the address <math>(\text{TIMx\_CR1 address}) + \text{DBA}</math>. The following situations may occur according to the configuration of DMA data length:</p> <ul style="list-style-type: none"> <li>- If the data is configured in half-words (16 bits), then the data is transferred to all of seven registers.</li> <li>- If the data is configured in bytes, the data is still transferred to all of seven registers: the first register contains the first MSB byte, the second contains the first LSB byte, and so on. Therefore, users must specify the data length in the DMA transfer as for timers.</li> </ul>
7:5	Reserved			Reserved and always read as 0.
4:0	DBA	w	0x00	<p>DMA base address This bit-field defines the base-address for DMA burst transfer (when a write access is done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.</p> <ul style="list-style-type: none"> <li>00000: TIMx_CR1</li> <li>00001: TIMx_CR2</li> <li>00010: TIMx_SMCR</li> </ul>

.....

### 15.4.20 DMA address for full transfer (TIMx\_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

DMAB															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15:0	DMAB	w	0x0000	<p>DMA register for burst accesses</p> <p>A write operation to the TIMx_DMAR register accesses the register located at the address:</p> <p>'TIMx_CR1 address' + DBA + DMA index, where 'TIMx_CR1 address' is the address of the control register 1 (TIMx_CR1),</p> <p>'DBA' is the base address configured in TIMx_DCR register,</p> <p>'DMA index' is the offset automatically controlled by DMA,</p> <p>and depends on the DBL configured in TIMx_DCR.</p>

### 15.4.21 Capture/compare mode register 3 (TIMx\_CCMR3)

Address offset: 0x54

Reset value: 0x0000

The channels can only be used in output (compare mode).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OC5 CE	OC5M			OC5 PE	OC5 FE	Reserved	
								rw	rw	rw	rw	rw			

#### Output Compare mode:

Bit	Field	Type	Reset	Description
15:8	Reserved			Reserved, always read as 0
7	OC5CE	rw	0x00	Output compare 5 clear enable
6:4	OC5M	rw	0x00	Output compare 5 mode
3	OC5PE	rw	0x00	Output compare 5 preload enable
2	OC5FE	rw	0x00	Output compare 5 fast enable

1:0	Reserved	Reserved and always read as 0.
-----	----------	--------------------------------

**15.4.22 Capture/Compare register 5 (TIMx\_CCR5)**

Address offset: 0x58

Reset value: 0x0000

Bit	Field	Type	Reset	Description
15:0	CCR5	rw	0x0000	<p>Capture/Compare 5 value</p> <p>CC5 channel can only be configured as output:</p> <p>CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).</p> <p>It is loaded immediately if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.</p> <p>The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC5 output.</p>

**15.4.23 PWM phase-shift/DMA repeat update request enable register (TIMx\_PDER)**

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved														CCR5_SHIFT_EN	CCR4_SHIFT_EN	CCR3_SHIFT_EN	CCR2_SHIFT_EN	CCR1_SHIFT_EN	CCD_REPEAT
									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Field	Type	Reset	Description
15:6	Reserved			Reserved, always read as 0
5	CCR5_SHIFTEN	rw	0x00	<p>Allow channel 5 to output PWM phase shift enable bit</p> <p>0: disable channel 5 to output PWM phase shift</p> <p>1: enable channel 5 to output PWM phase shift</p> <p>For details, see the phase shift description for CCRx_FALL register.</p>
4	CCR4_SHIFTEN	rw	0x00	<p>Allow channel 4 to output PWM phase shift enable bit</p> <p>0: disable channel 4 to output PWM phase shift</p> <p>1: enable channel 4 to output PWM phase shift</p> <p>For details, see the phase shift description for CCRx_FALL register.</p>

3	CCR3_SHIFTEN	rw	0x00	Allow channel 3 to output PWM phase shift enable bit 0: disable channel 3 to output PWM phase shift 1: enable channel 3 to output PWM phase shift For details, see the phase shift description for CCRx_FALL register.
Bit	Field	Type	Reset	Description
2	CCR2_SHIFTEN	rw	0x00	Allow channel 2 to output PWM phase shift enable bit SHIFTEN 0: disable channel 2 to output PWM phase shift 1: enable channel 2 to output PWM phase shift For details, see the phase shift description for CCRx_FALL register.
1	CCR1_SHIFTEN	rw	0x00	Allow channel 1 to output PWM phase shift enable bit 0: disable channel 1 to output PWM phase shift 1: enable channel 1 to output PWM phase shift For details, see the phase shift description for CCRx_FALL register.
0	CCDREPE	rw	0x00	Enable the DMA update request upon each underflow 0: Enable the DMA update request depending on the repetition counter value 1: Enable the DMA update request upon each underflow

#### 15.4.24 PWM phase-shift downcounting capture/compare register (CCRxFall)

Address offset: 0x60 ~ 0x70

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description
15:0	CCRxFall	rw	0x0000	Channel x capture/compare value when downcounting in the PWM center-aligned mode PWM phase-shift function: To realize PWM outputs with programmable phase-shift waveform (left shift or right shift as required), the user should enable the PWM phase-shift function in the PDER register and set the CCRxFALL and CCRx.

# 16

# 16-Bit General-Purpose Timer (TIM3)

## 16-Bit General-Purpose Timer (TIM3)

### 16.1 TIMx introduction

The general-purpose timers consist of a 16-bit auto-reload counter driven by a programmable prescaler. They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIMx timers are completely independent, and do not share any resources. They can be synchronized together.

### 16.2 TIMx main features

The features of the general-purpose TIMx(TIM3) timer include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler (can be changed “on the fly”) used to divide the counter clock frequency by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge or center-aligned Mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

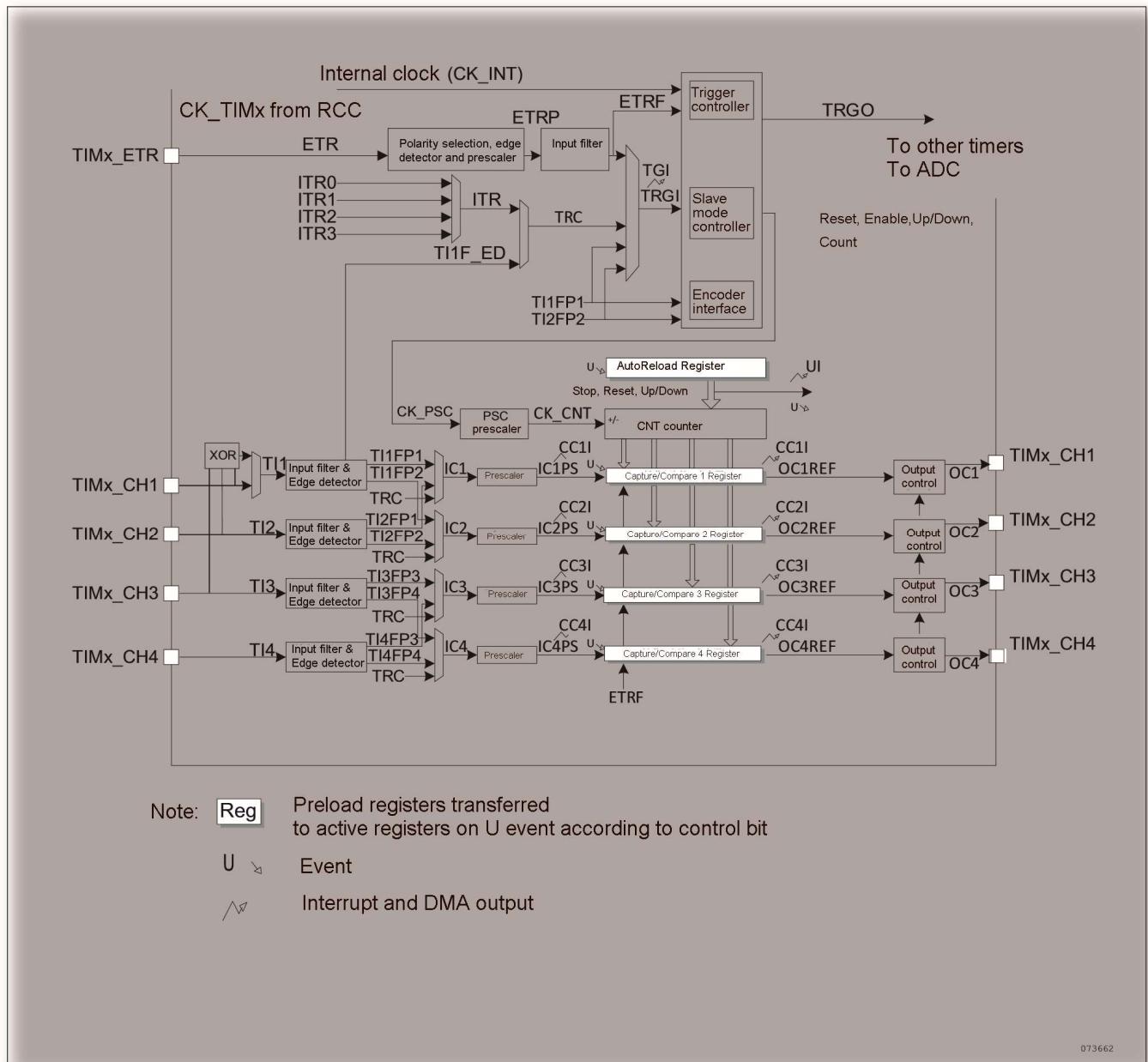


Figure 90. General-purpose timer block diagram

### 16.3. TIMx functional description

#### 16.3.1 Time-base unit

The main block of the programmable general-purpose timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running. The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register immediately or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

## Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

The figures below give some examples of the counter behavior when the prescaler ratio is changed on the fly:

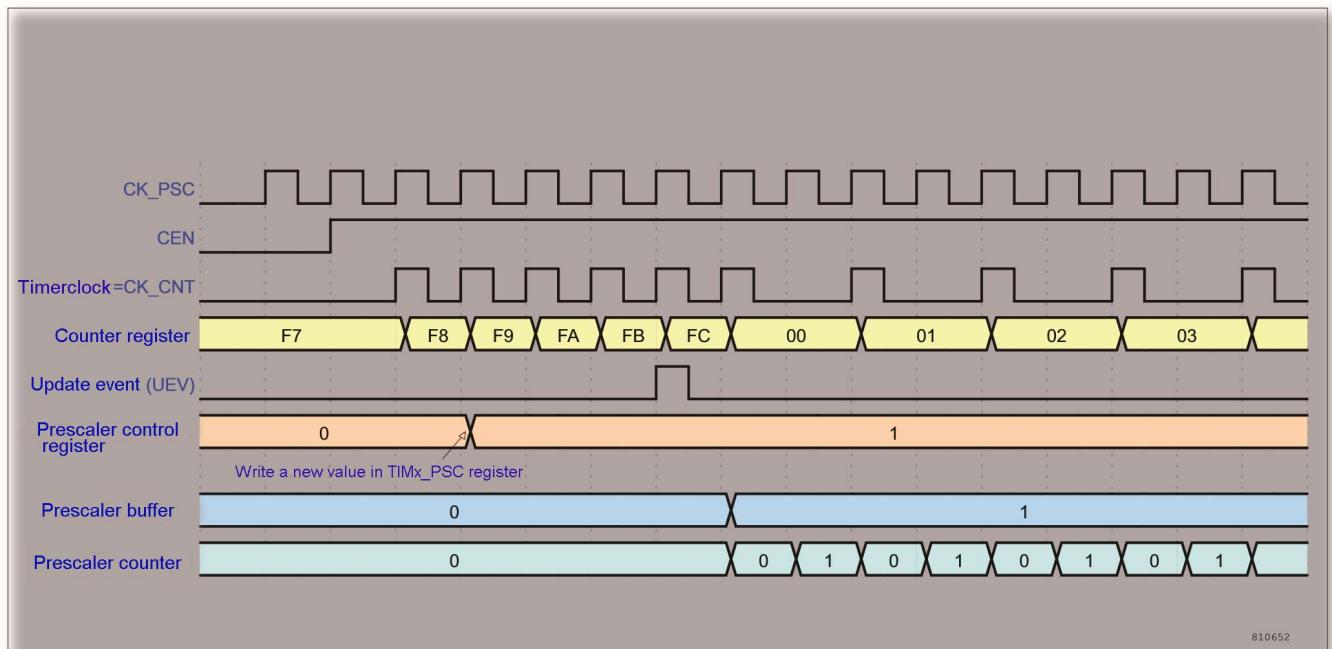


Figure 91. Counter timing diagram with prescaler division change from 1 to 2

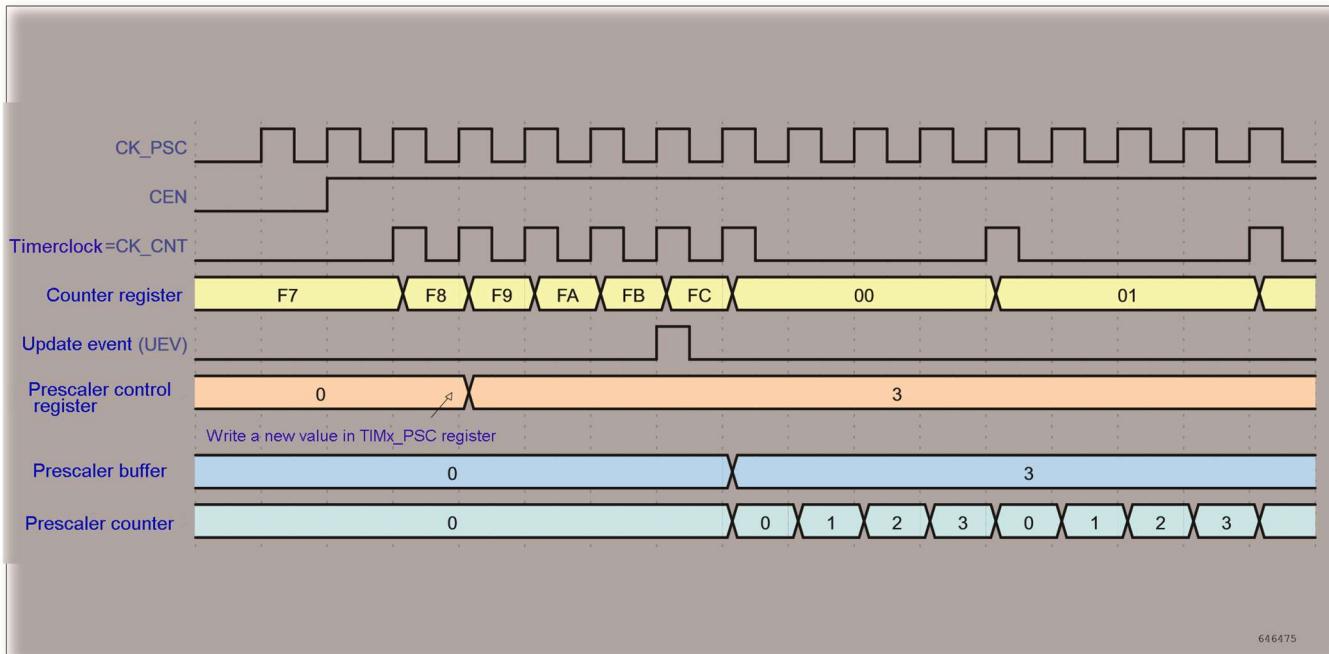


Figure 92. Counter timing diagram with prescaler division change from 1 to 4

### 16.3.2 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR counter), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change) when an update event should be generated. In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag by hardware (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture mode.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set by hardware (depending on the URS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR = 0x36:

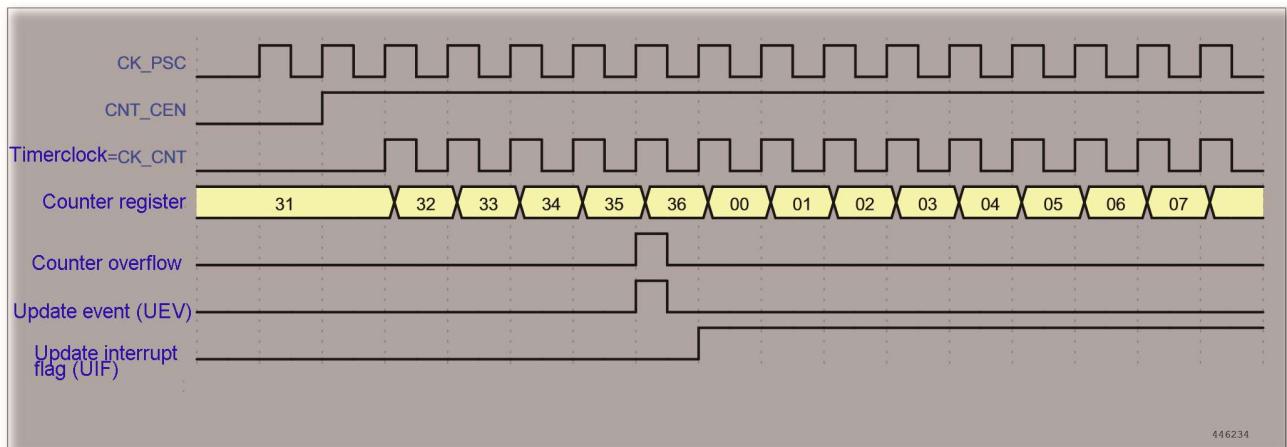


Figure 93. Counter timing diagram, internal clock divided by 1

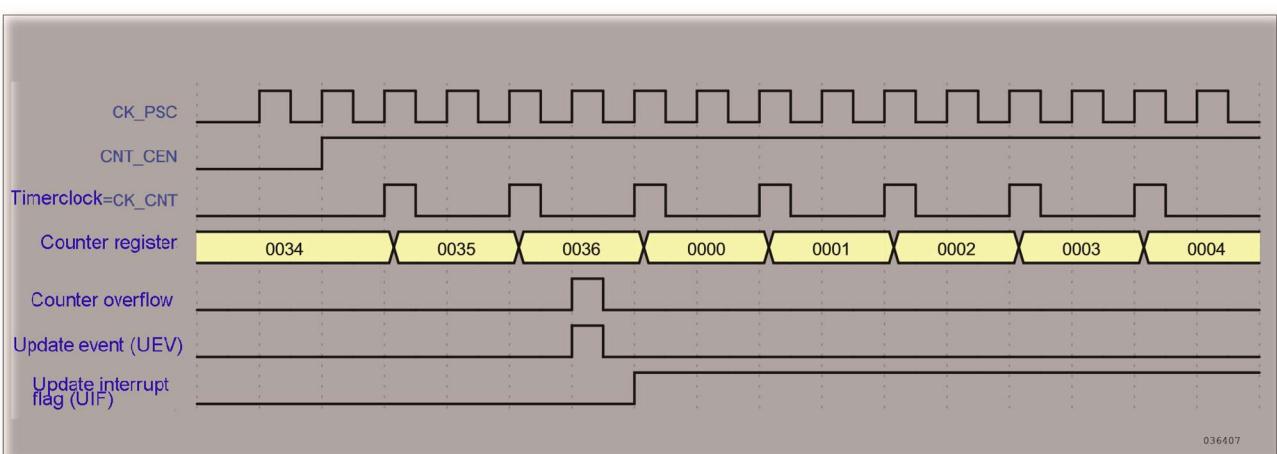


Figure 94. Counter timing diagram, internal clock divided by 2

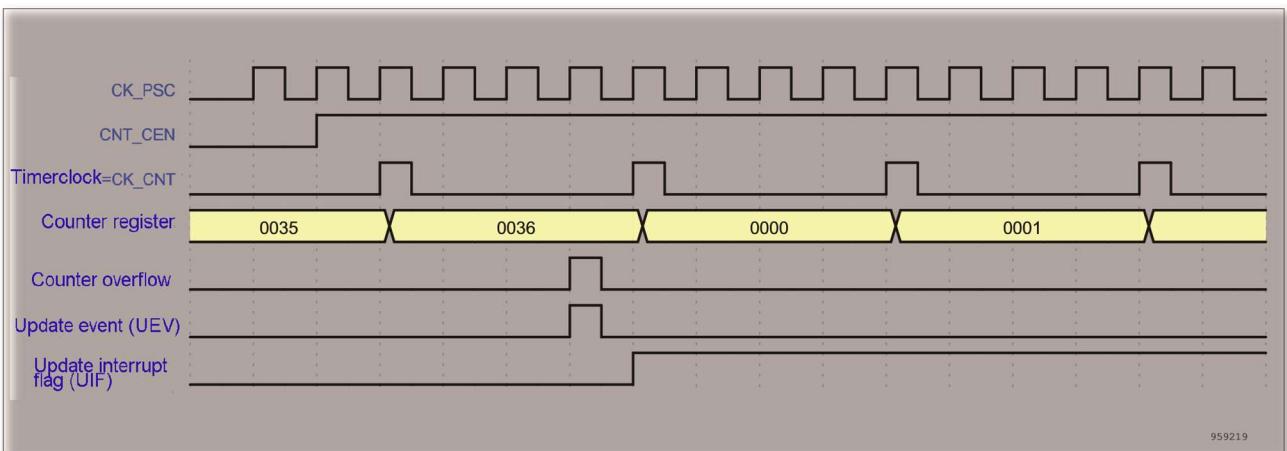


Figure 95. Counter timing diagram, internal clock divided by 4

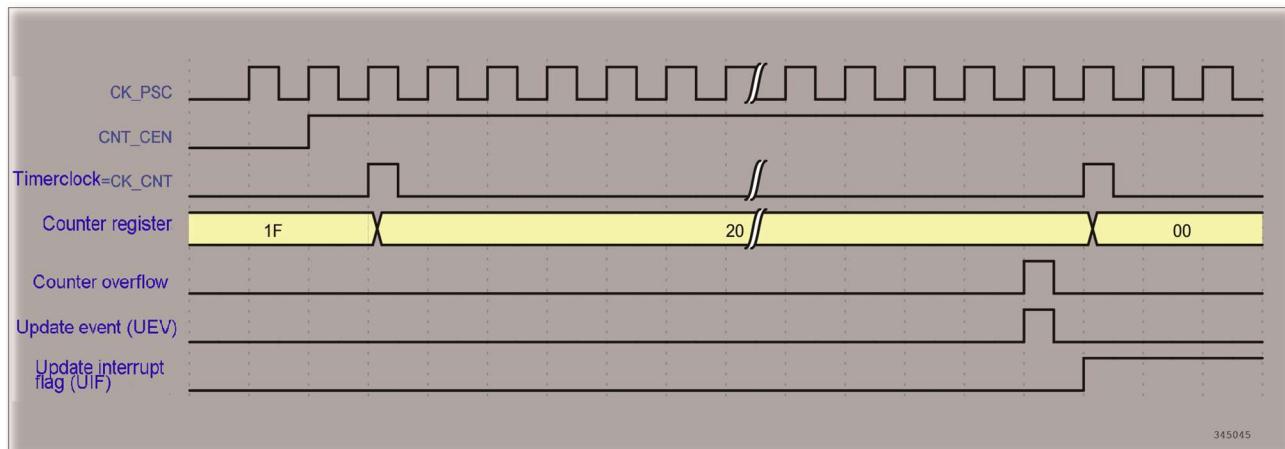


Figure 96. Counter timing diagram, internal clock divided by N

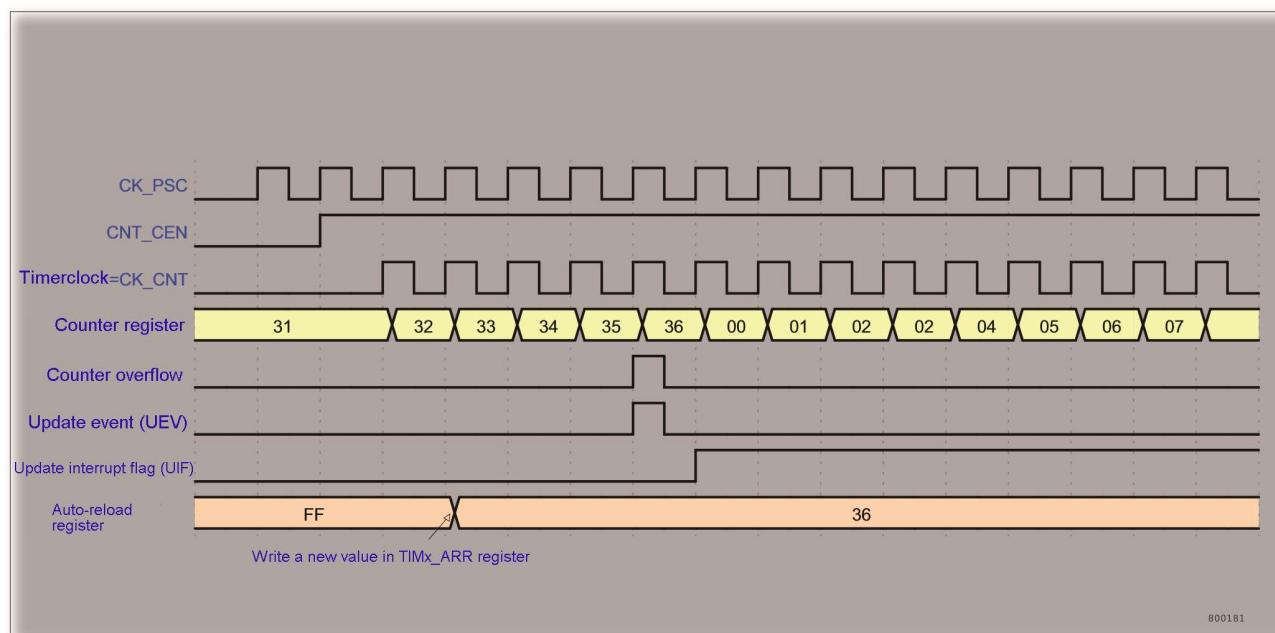


Figure 97. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)

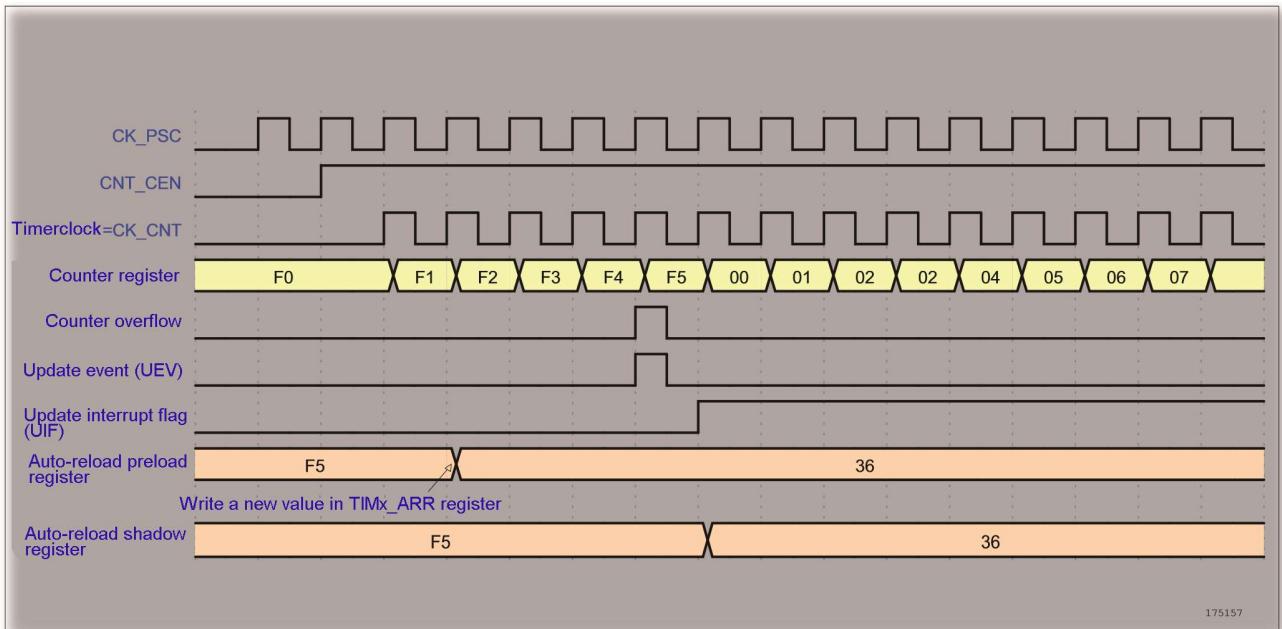


Figure 98. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)

### Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx\_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller).

The UEV update event can be disabled by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register).

Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR = 0x36.

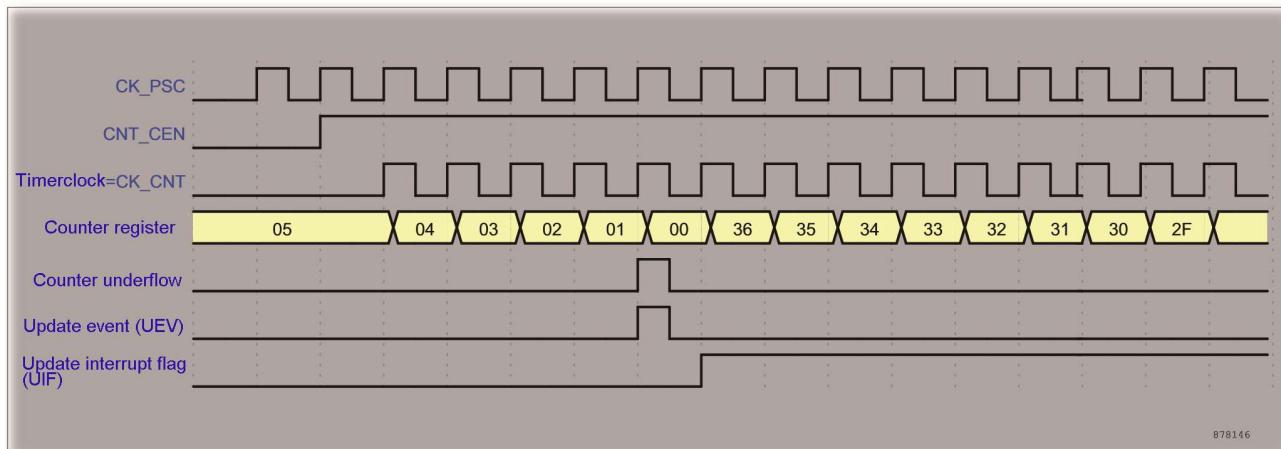


Figure 99. Counter timing diagram, internal clock divided by 1

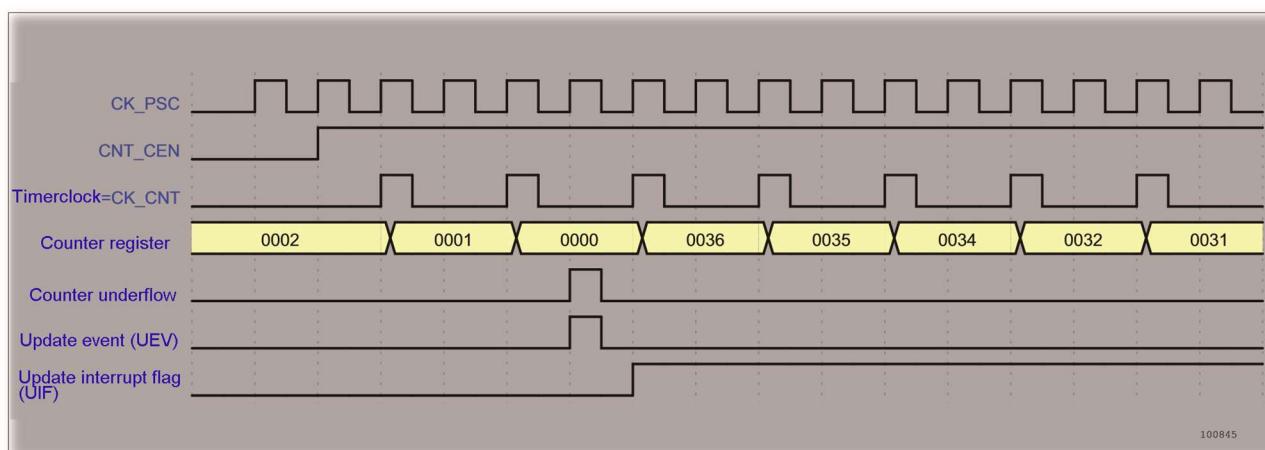


Figure 100. Counter timing diagram, internal clock divided by 2

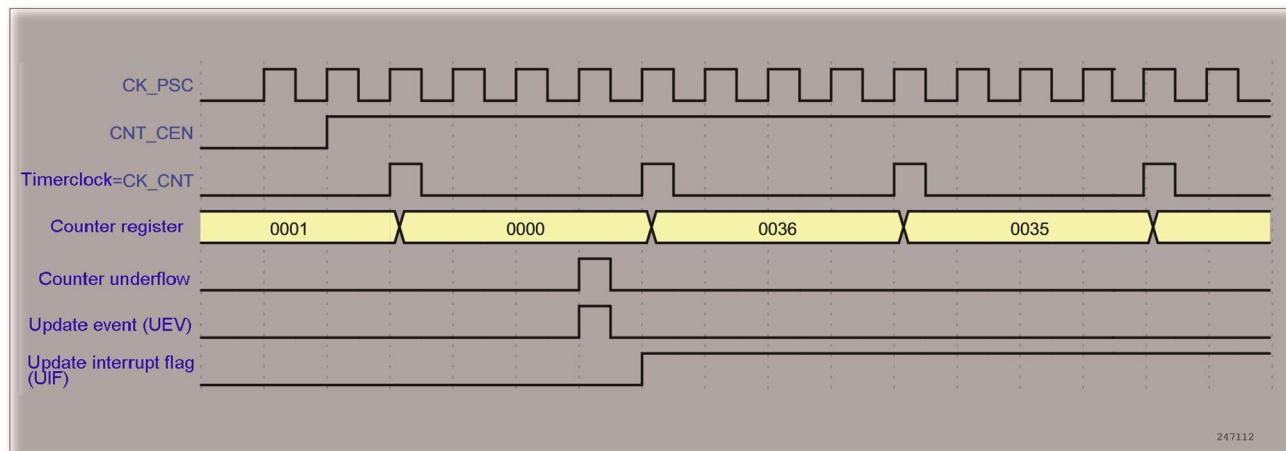


Figure 101. Counter timing diagram, internal clock divided by 4

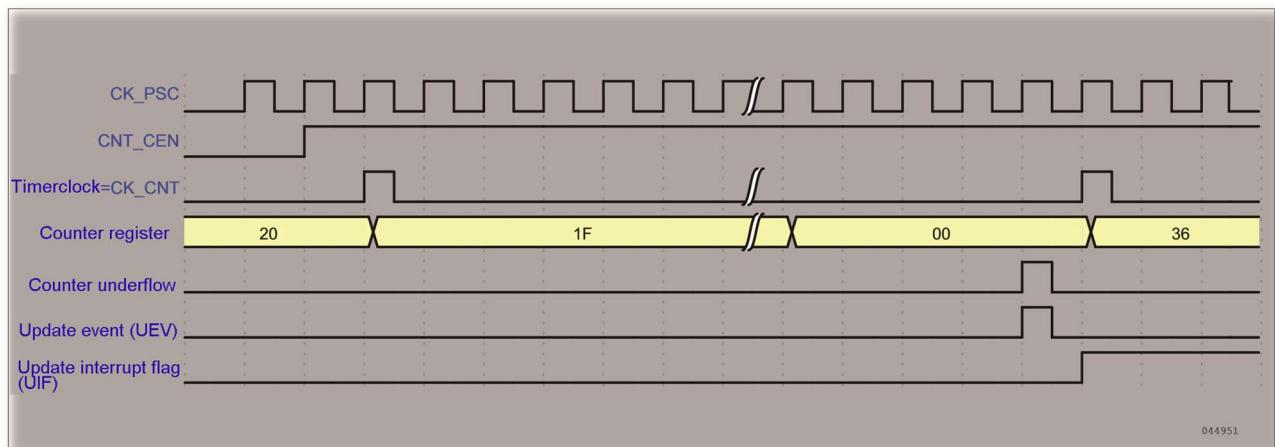


Figure 102. Counter timing diagram, internal clock divided by N

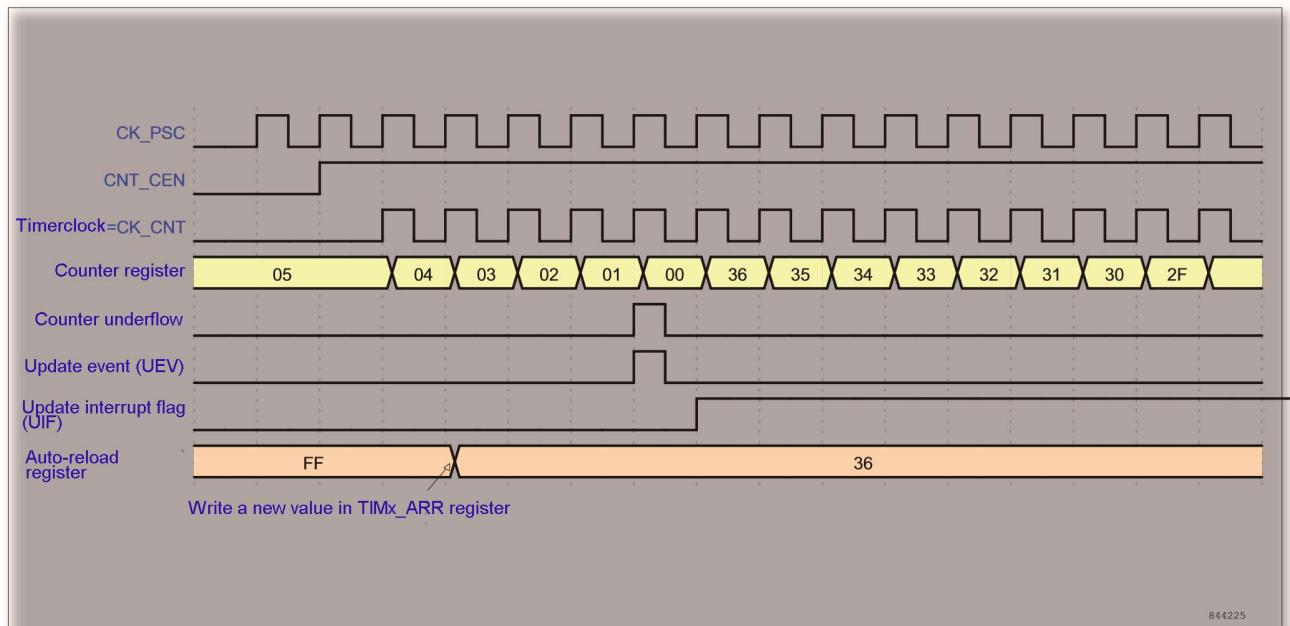


Figure 103. Counter timing diagram, update event when repetition counter is not used

### Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

In this mode, the DIR direction bit in the TIMx\_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter. The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller). In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register).

Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies:

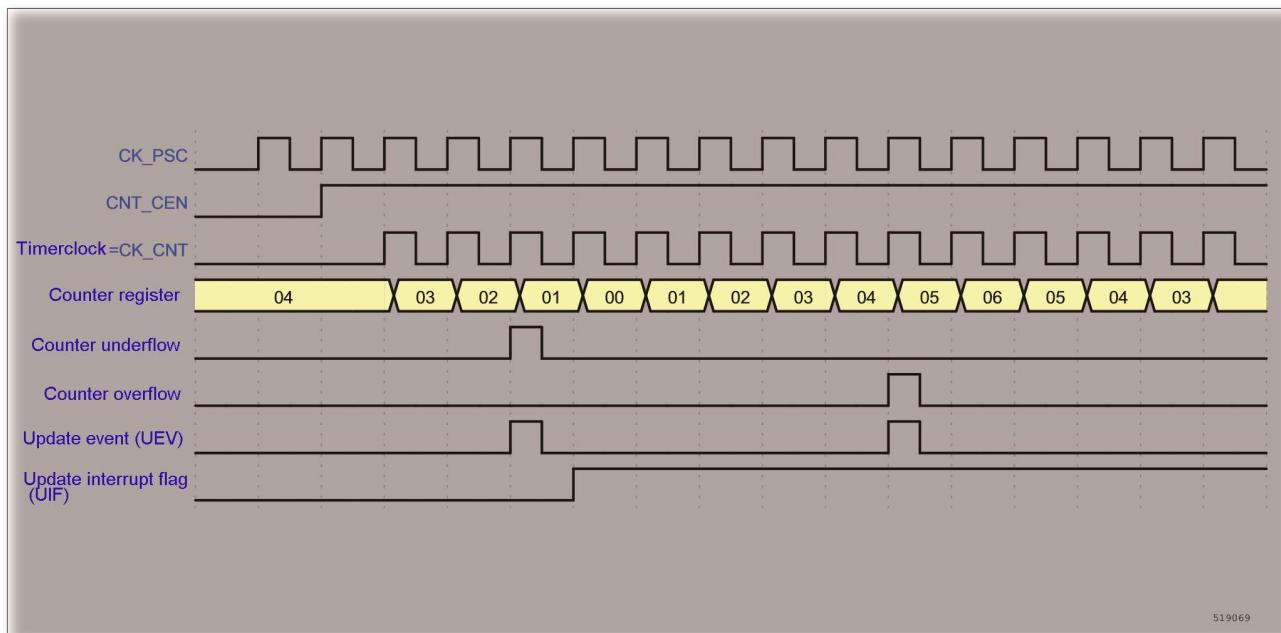


Figure 104. Counter timing diagram, internal clock divided by 1, TIMx\_ARR = 0x6

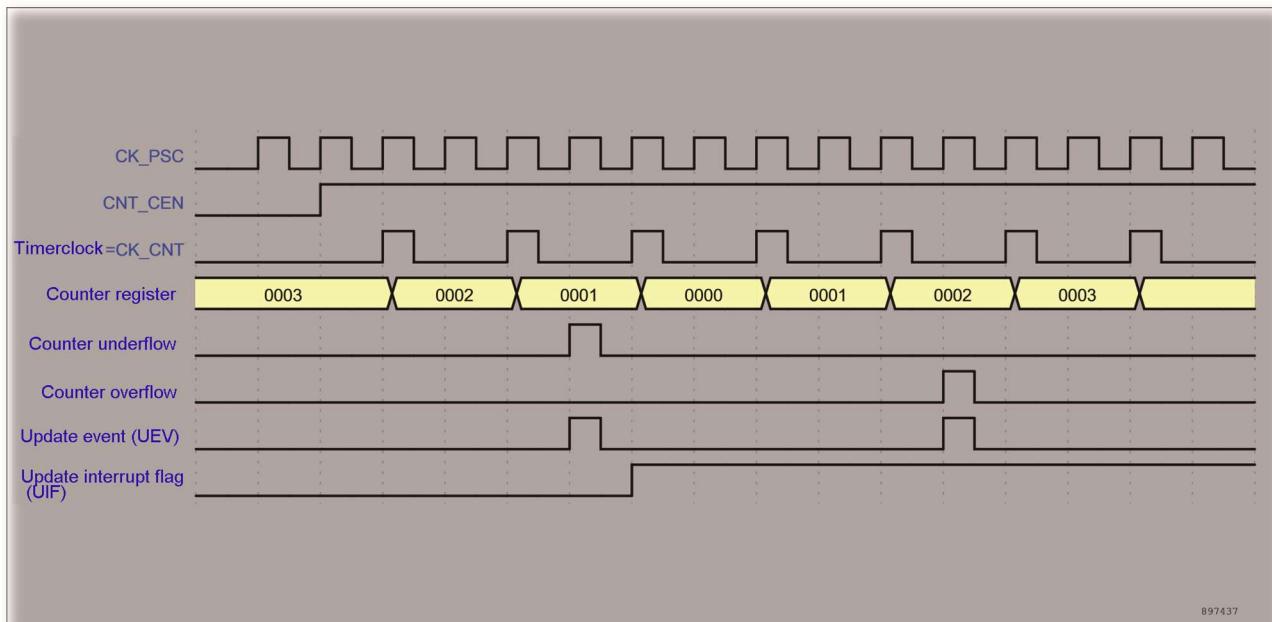


Figure 105. Counter timing diagram, internal clock divided by 2

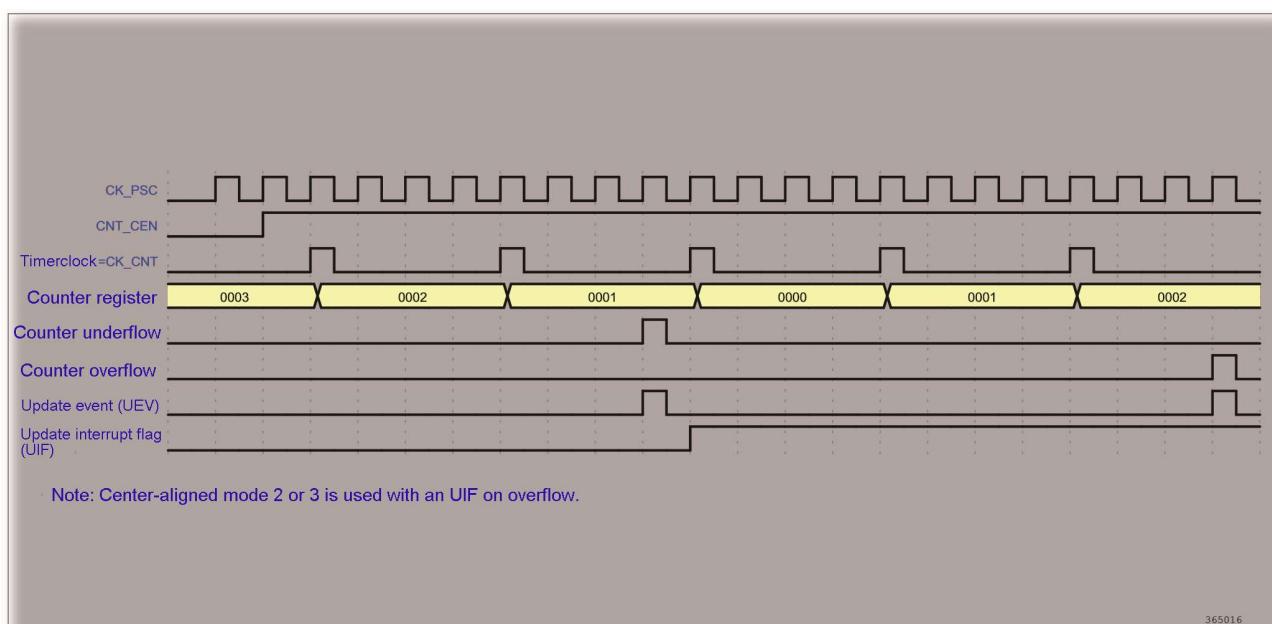


Figure 106. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x03

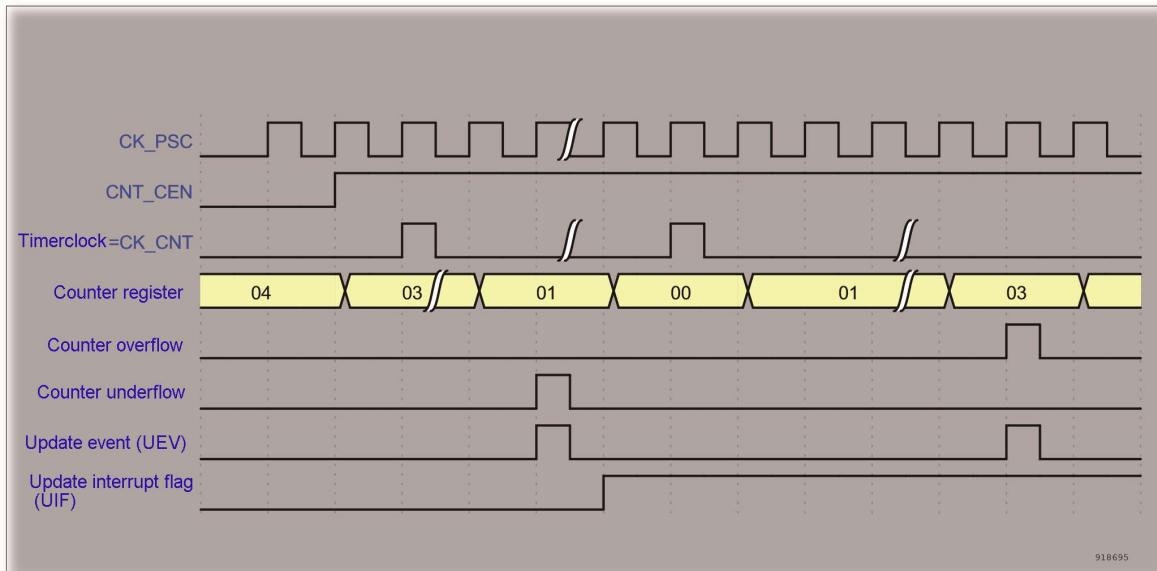


Figure 107. Counter timing diagram, internal clock divided by N

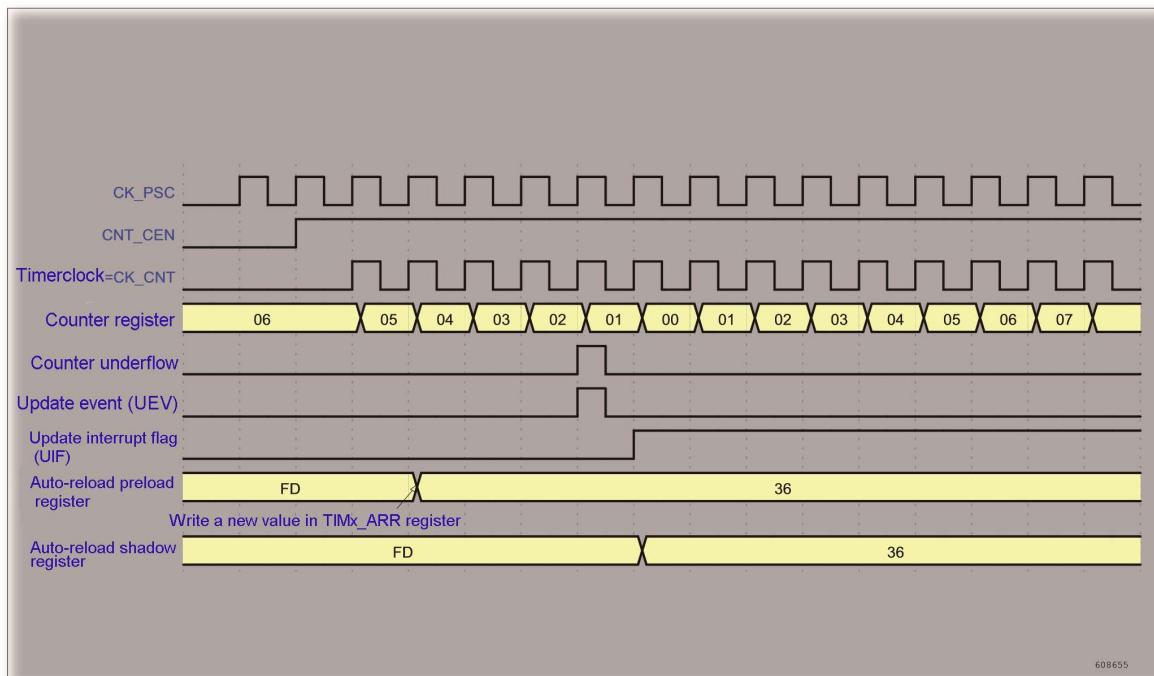


Figure 108. Counter timing diagram, update event with ARPE=1 (counter underflow)

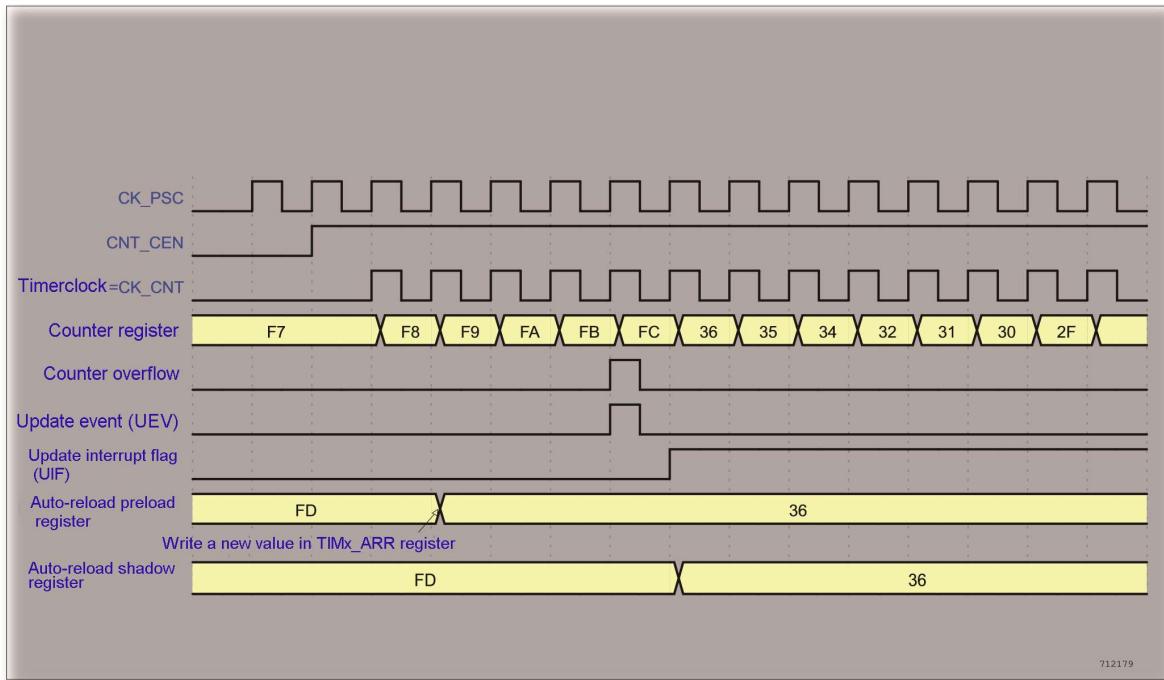


Figure 109. Counter timing diagram, Update event with ARPE=1 (counter overflow)

### 16.3.3 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK\_INT).
- External clock mode 1: external input pin (TIx).
- External clock mode 2: external trigger input (ETR).
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, the user can configure Timer 1 to act as a prescaler for Timer 2.

#### Internal clock source (CK\_INT)

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

The figure below shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

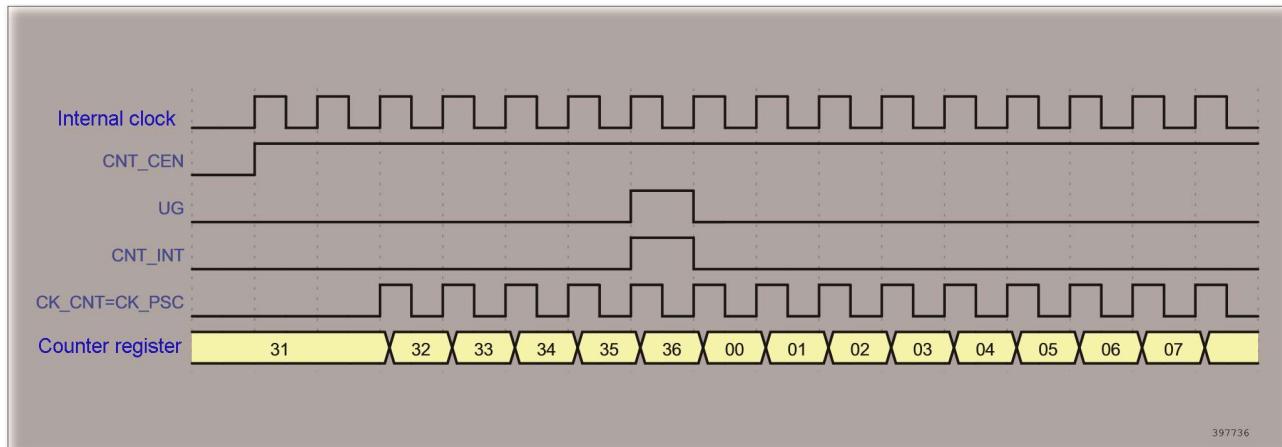


Figure 110. Control circuit in normal mode, internal clock divided by 1

### External clock source mode 1

This mode is selected when SMS=111 in the TIMx\_SMCR register. The counter can count at each rising or falling edge on a selected input.

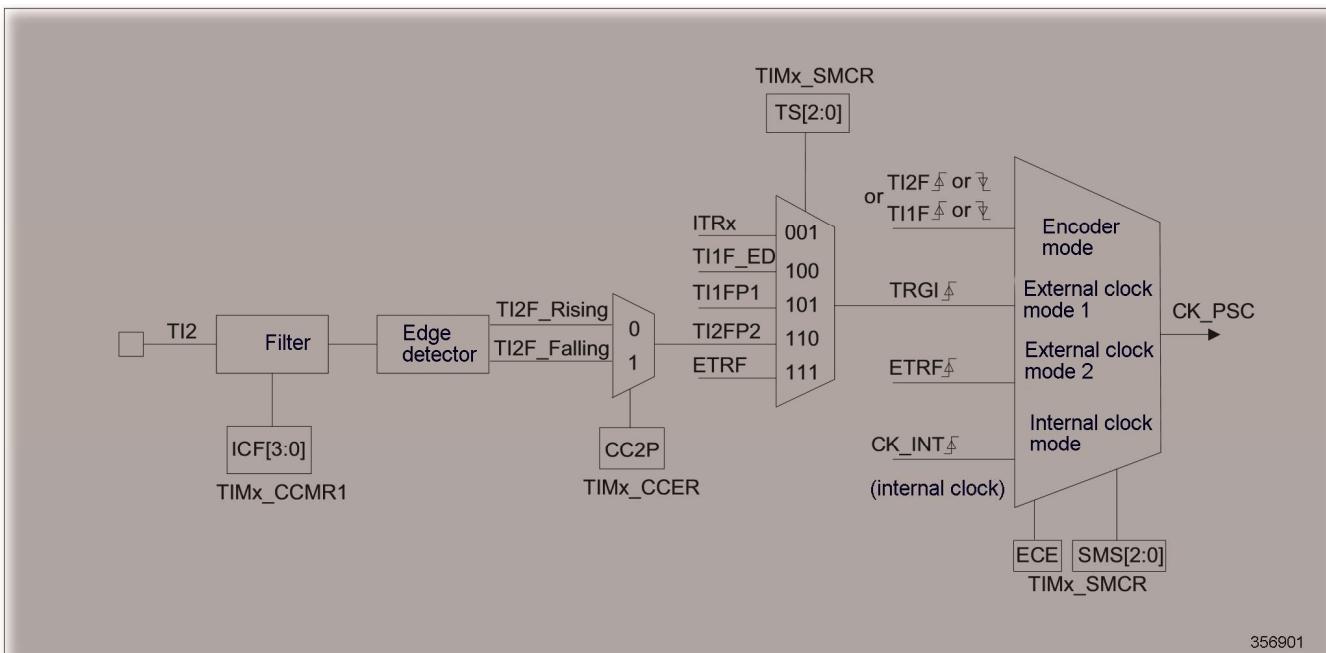


Figure 111. TI2 external clock connection example

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = 01 in the **TIMx\_CCMR1** register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the **TIMx\_CCMR1** register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 in the **TIMx\_CCER** register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the **TIMx\_SMCR** register.
5. Select TI2 as the trigger input source by writing TS = 110 in the **TIMx\_SMCR** register.
6. Enable the counter by writing CEN = 1 in the **TIMx\_CR1** register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

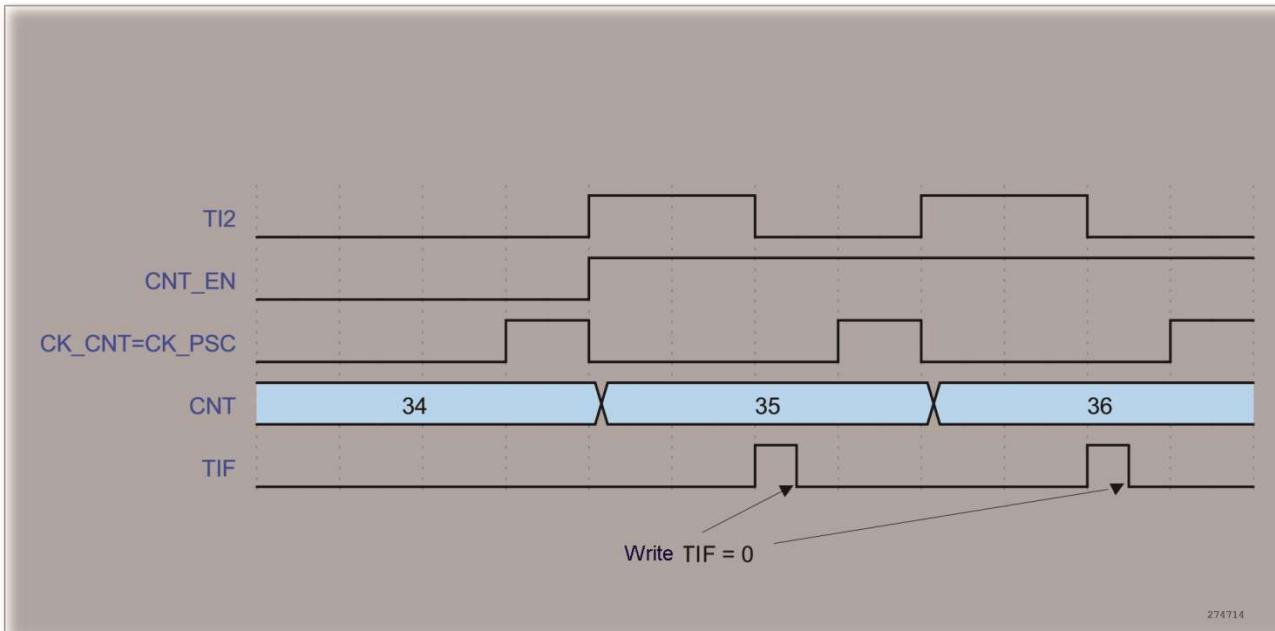


Figure 112. Control circuit in external clock mode 1

### External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx\_SMCR register. The counter can count at each rising or falling edge on the external trigger input ETR. The figure below gives an overview of the external trigger input block.

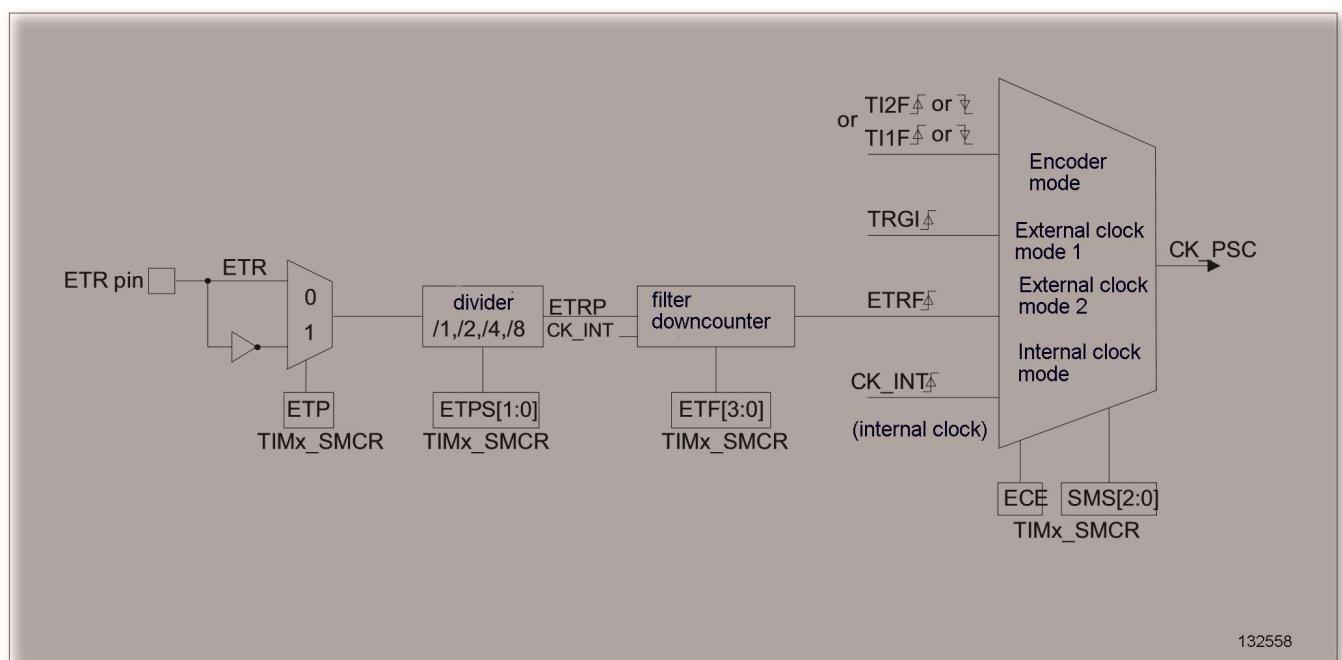


Figure 113. External trigger input block

For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write  $\text{ETF}[3:0] = 0000$  in the  $\text{TIMx}_\text{SMCR}$  register.
2. Set the prescaler by writing  $\text{ETPS}[1:0] = 01$  in the  $\text{TIMx}_\text{SMCR}$  register.
3. Select rising edge detection on the ETR pin by writing  $\text{ETP} = 0$  in the  $\text{TIMx}_\text{SMCR}$  register.
4. Enable external clock mode 2 by writing  $\text{ECE} = 1$  in the  $\text{TIMx}_\text{SMCR}$  register.
5. Enable the counter by writing  $\text{CEN} = 1$  in the  $\text{TIMx}_\text{CR1}$  register. The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

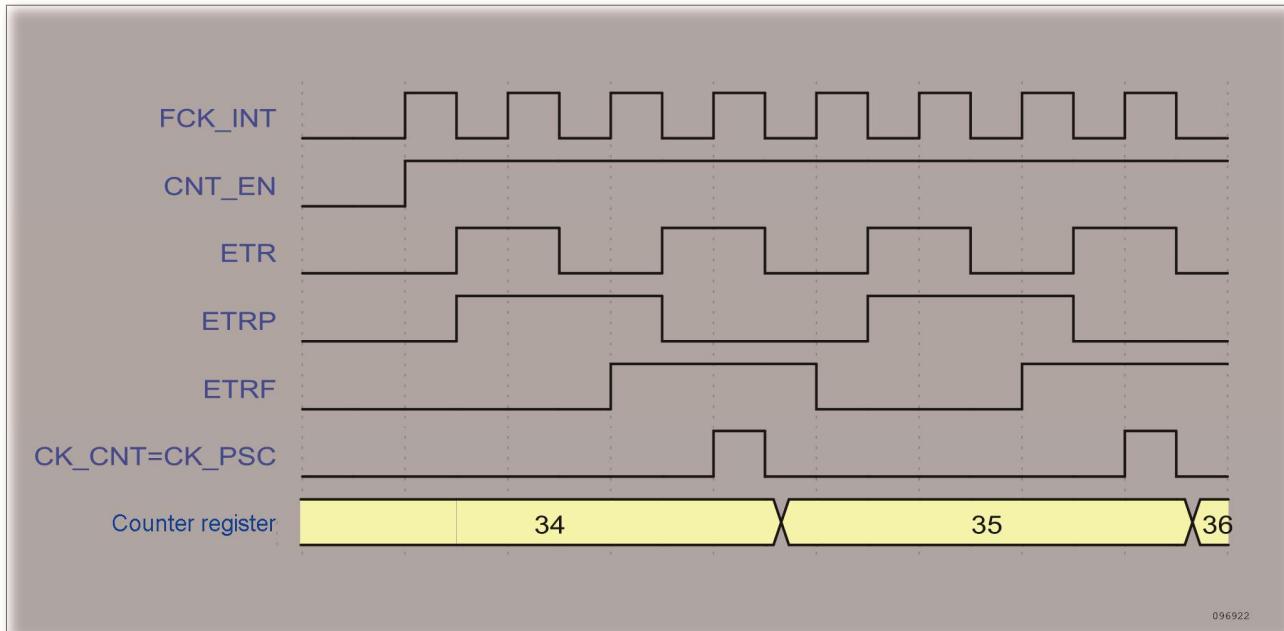
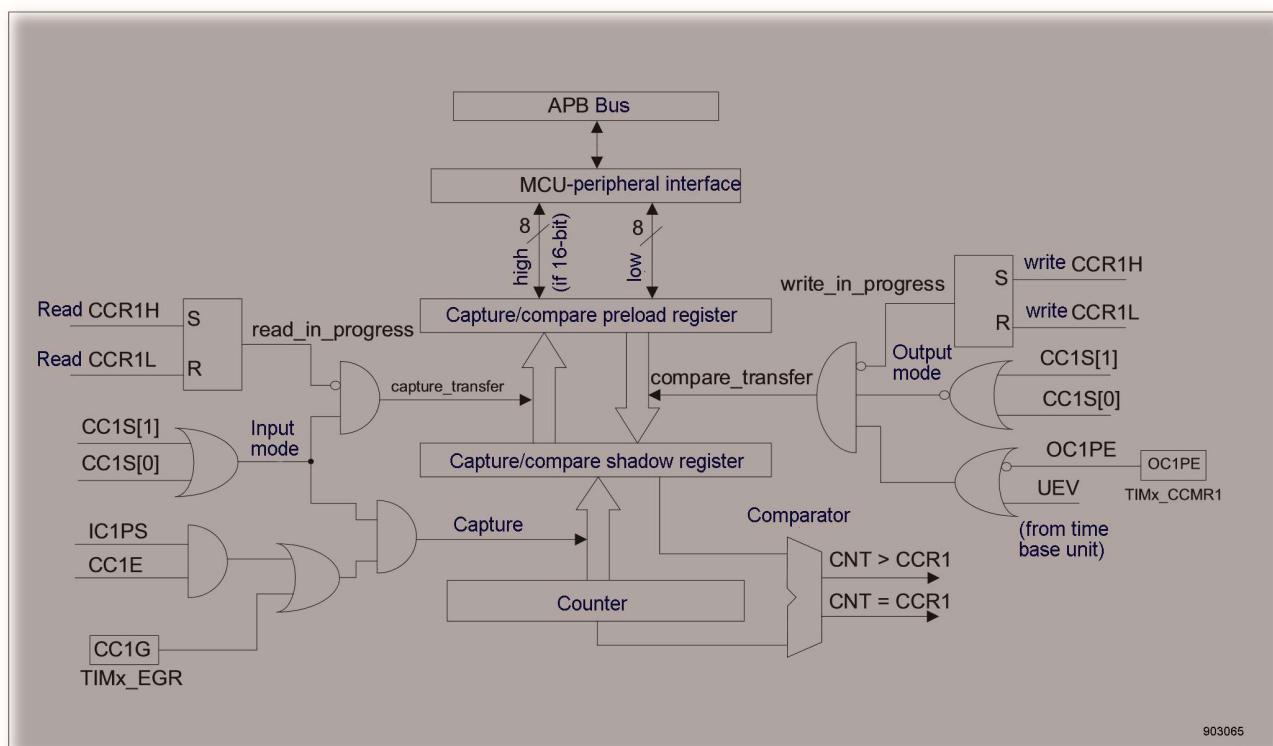
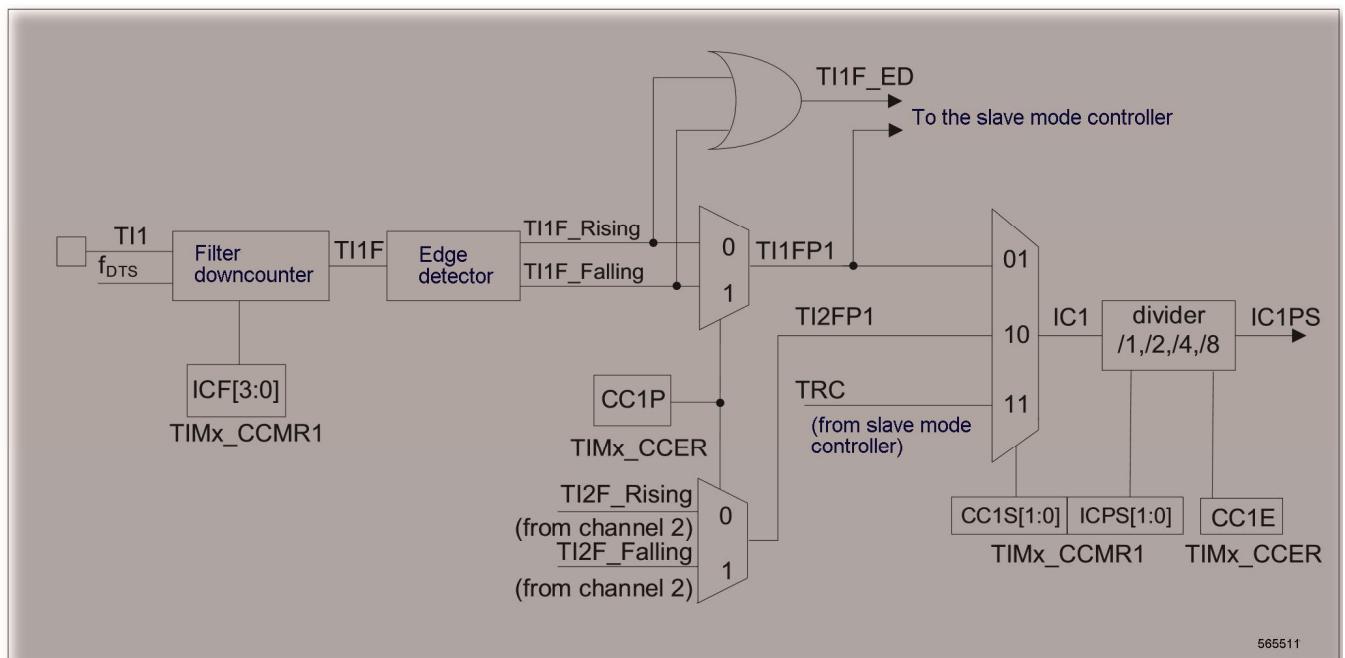


Figure 114. Control circuit in external clock mode 2

#### 16.3.4 Capture/compare channel

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figures give an overview of one Capture/Compare channel. The input stage samples the corresponding  $\text{TIx}$  input to generate a filtered signal  $\text{TIxF}$ . Then, an edge detector with polarity selection generates a signal ( $\text{TIxFPx}$ ) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register ( $\text{ICxPS}$ ).



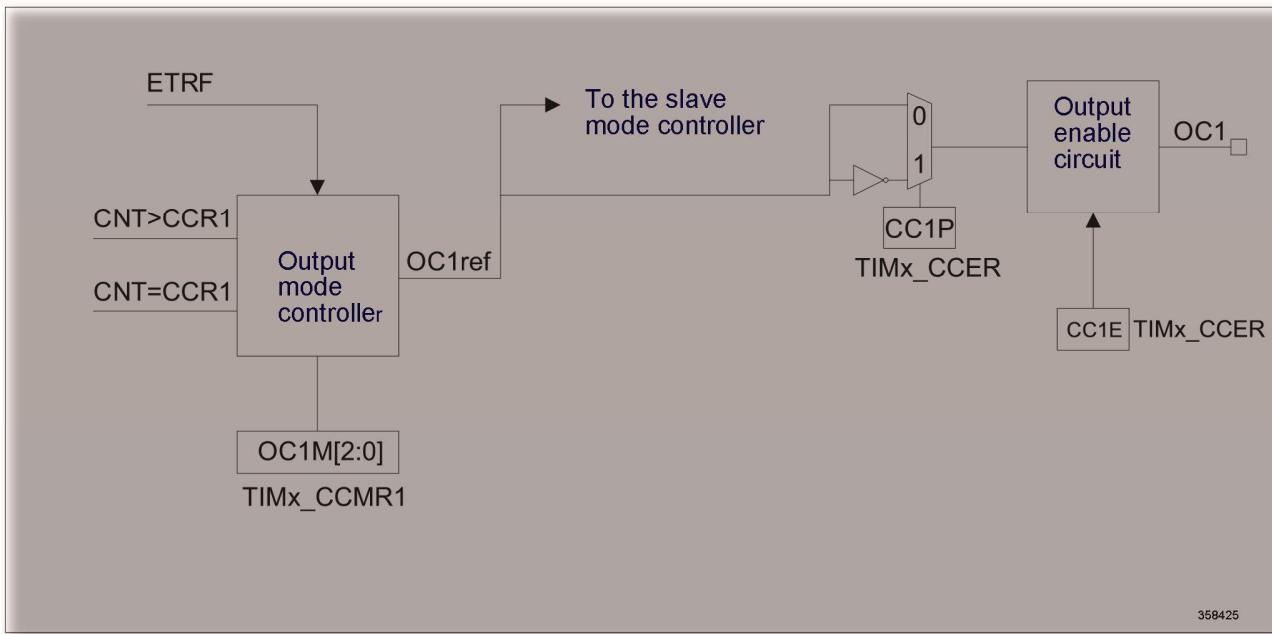


Figure 117. Output stage of capture/compare channel (channel 1)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register. In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 16.3.5 Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx\_CCRx) are used to latch the value of the counter after an edge is detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx\_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when written to '0'.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TM1\_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the input signal (by programming ICxF bits in the TIMx\_CCMRx register if the input is a TIx input). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate an edge transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at  $f_{DTS}$  frequency).

Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.

- Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx\_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).

- Enable capture from the counter into the capture register by setting the CC1E bit to ‘1’ in the TIMx\_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active level transition.
- CC1IF flag is set (interrupt flag).
- CC1OF is also set to ‘1’ if at least two consecutive captures occurred whereas the CC1IF flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.

### 16.3.6 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input
- These 2 ICx signals are active on edges with opposite polarity
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode

For example, the user can measure the period (in TIMx\_CCR1 register) and the duty cycle (in TIMx\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):

- Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P bit to ‘0’ (active on rising edge).
- Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P bit to ‘1’ (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx\_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx\_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx\_CCER register.

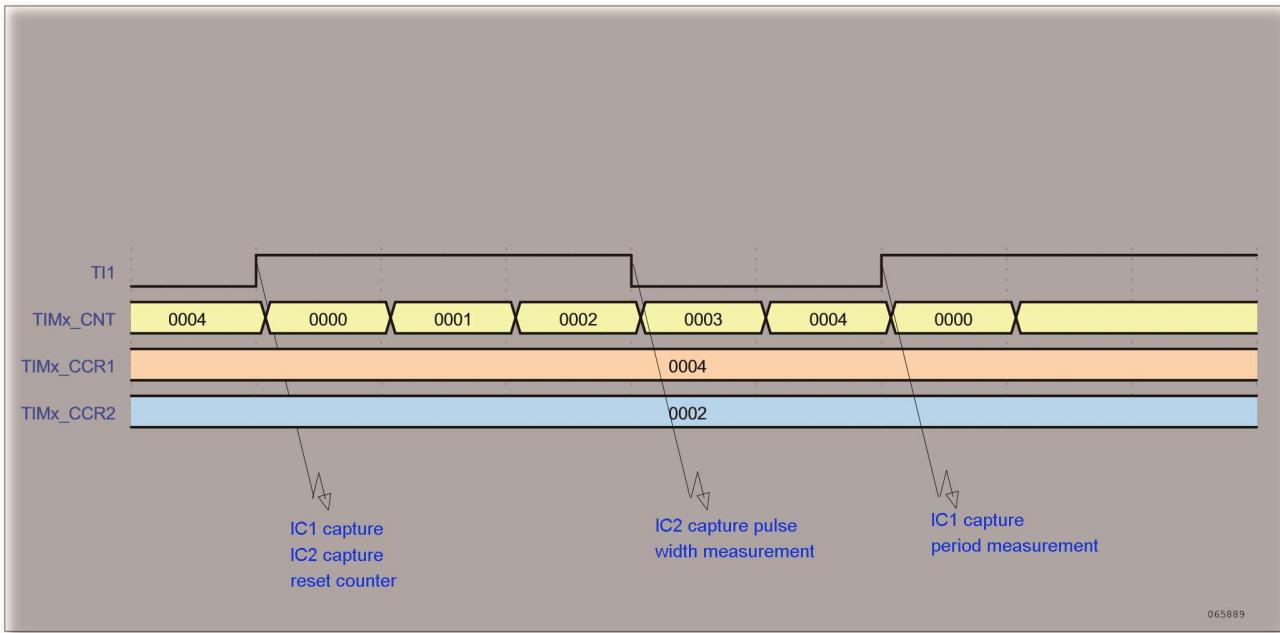


Figure 118. PWM input mode timing

The PWM input mode can be used only with the TIMx\_CH1/TIMx\_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

### 16.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCxREF is forced high (OCxREF is always active high) and OCx gets opposite value to CCxP polarity bit.

For example: CCxP = 0 (OCx active high) => OCx is forced to high level. The OCxREF signal can be forced low by writing the OCxM bit to 100 in the TIMx\_CCMRx register. Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 16.3.8 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bit in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM = 000), be set active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx\_DIER register).
- Generates a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register. In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The synchronization resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

The output compare mode is configured as follows:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxE or CCxDE bits if an interrupt or a DMA request is to be generated.
4. Select the output mode. For example, the user must write OCxM = 011, OCxPE = 0, CCxP = 0 and CCxE = 1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in the following figure.

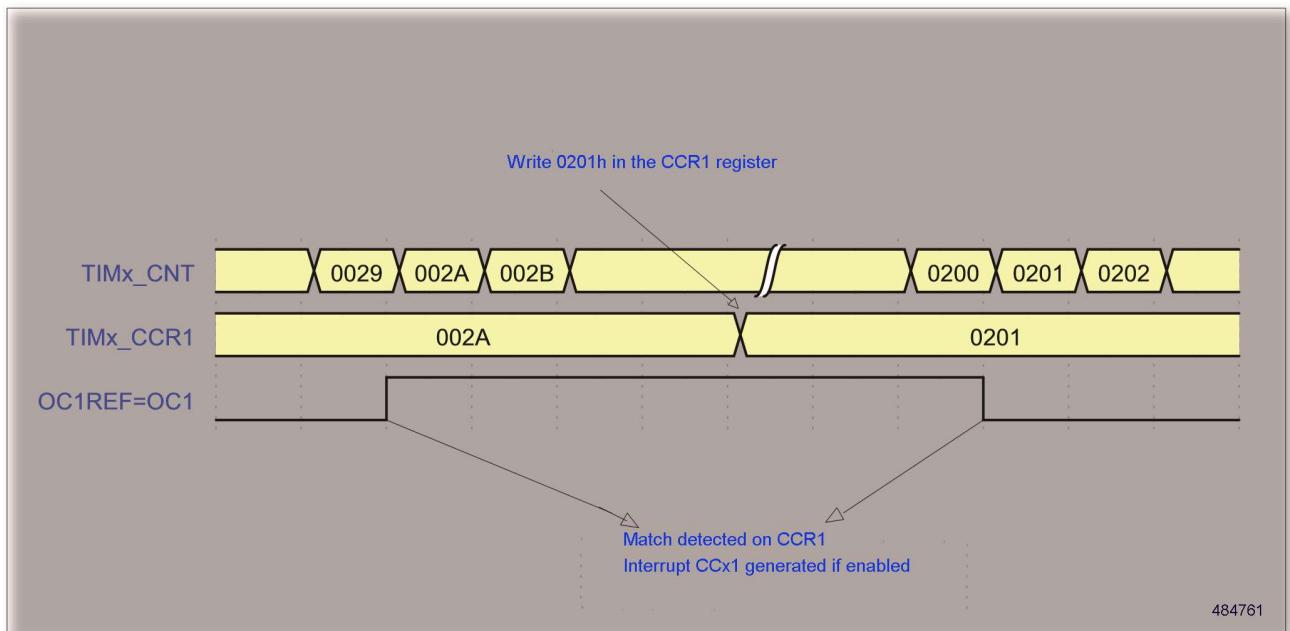


Figure 119. Output compare mode, toggle on OC1

### 16.3.9 PWM mode

Pulse Width Modulation mode allows generating a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bit in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user must initialize all the registers by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by using the CCxE bit in the TIMx\_CCER register. Refer to the TIMx\_CCERx register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIM1\_CCRx are always compared to determine whether  $\text{TIM1\_CCR}_x \leq \text{TIM1\_CNT}$  or  $\text{TIM1\_CNT} \leq \text{TIM1\_CCR}_x$  (depending on the direction of the counter).

However, to comply with the OCREF\_CLR functionality (OCxREF can be cleared by an external event through the ETR signal until the next PWM period), the OCxREF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (OCxM bits in TIMx\_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM='000) to one of the PWM modes (OCxM='110 or '111).

This forces the PWM by software while the timer is running. The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx\_CR1 register.

## PWM edge-aligned mode

### Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as  $\text{TIMx\_CNT} < \text{TIMx\_CCR}_x$ ; else it becomes low. If the compare value in TIMx\_CCRx is greater than the auto-reload value (in TIMx\_ARR), then OCxREF is held at '1'. If the compare value is 0 then OCxREF is held at '0'. The following figure shows some edge-aligned PWM waveforms in an example where  $\text{TIMx\_ARR} = 8$ .

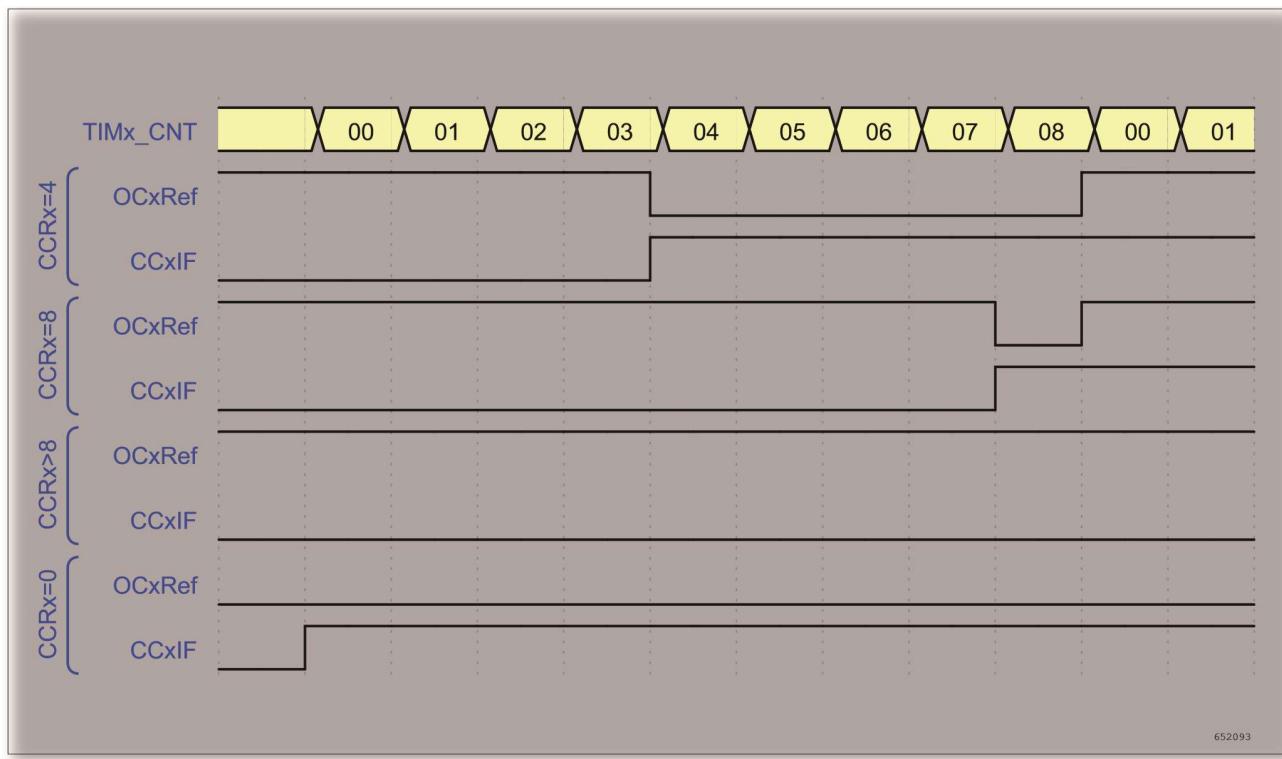


Figure 120. Edge-aligned PWM waveforms (ARR = 8)

### Downcounting configuration

Downcounting is active when DIR bit in TIMx\_CR1 register is high.

In PWM mode 1, the reference signal OCxREF is low as long as  $\text{TIMx\_CNT} > \text{TIMx\_CCR}_x$ ; else it becomes high. If the compare value in TIMx\_CCRx is greater than the auto-reload value in TIMx\_ARR, then OCxREF is held at '1'.

0% PWM is not possible in this mode.

### PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxREF/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx\_CR1 register is updated by hardware and must not be changed by software. Refer to center-aligned mode section.

The figure below shows some center-aligned PWM waveforms in an example where:

- TIMx\_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx\_CR1 register.

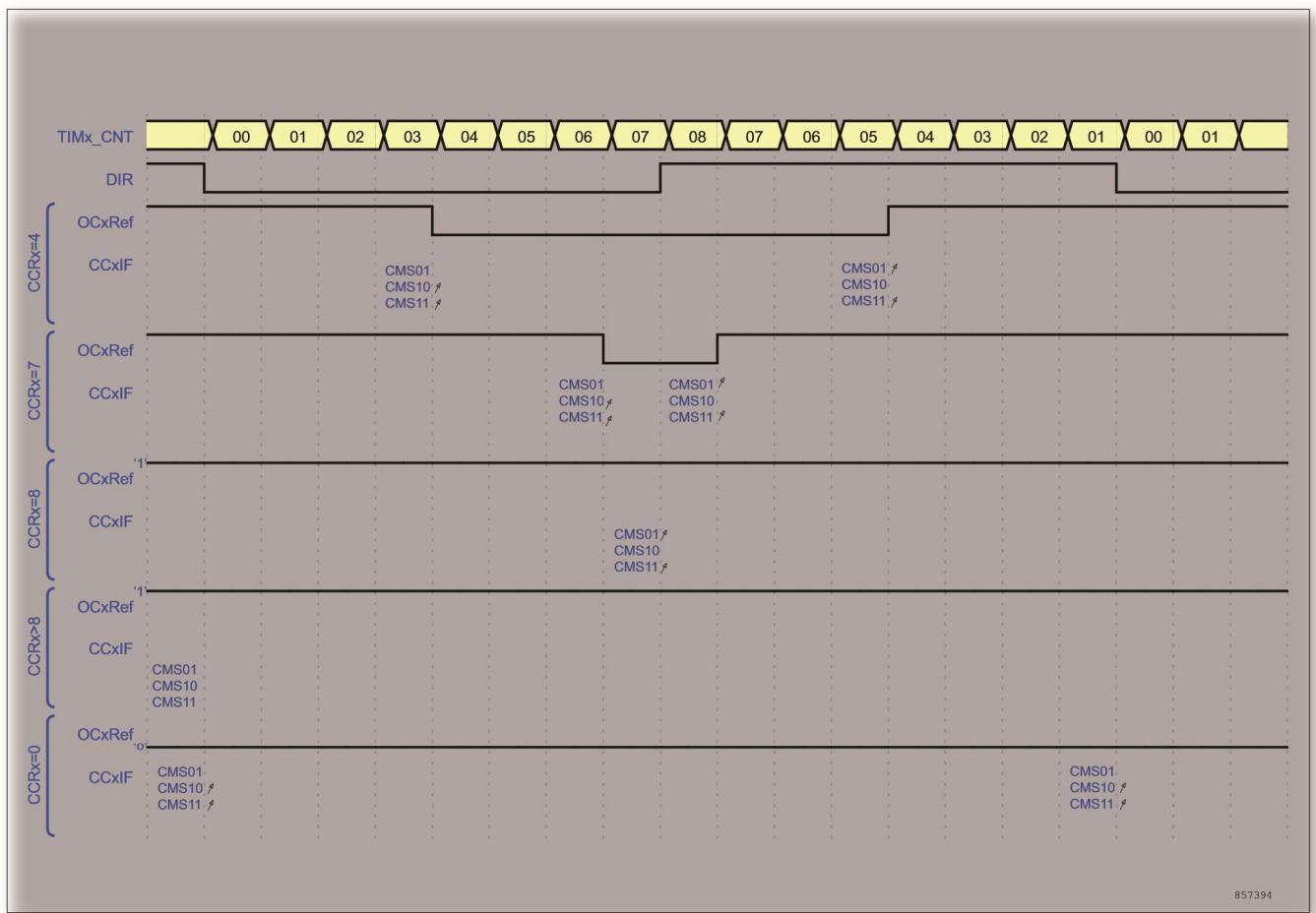


Figure 121. Center-aligned PWM waveforms (ARR = 8)

#### Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:

- The direction is not updated if the user writes a value in the counter greater than the auto-reload value ( $\text{TIMx\_CNT} > \text{TIMx\_ARR}$ ). For example, if the counter was counting up, it will continue to count up.
- The direction is updated if the user writes 0 or writes the  $\text{TIMx\_ARR}$  value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the  $\text{TIMx\_EGR}$  register) just before starting the counter and not to write the counter while it is running.

### 16.3.10 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select One-pulse mode by setting the OPM bit in the  $\text{TIMx\_CR1}$  register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting:  $\text{CNT} < \text{CCRx} \leq \text{ARR}$  (in particular,  $0 < \text{CCRx}$ )
- In downcounting:  $\text{CNT} > \text{CCRx}$

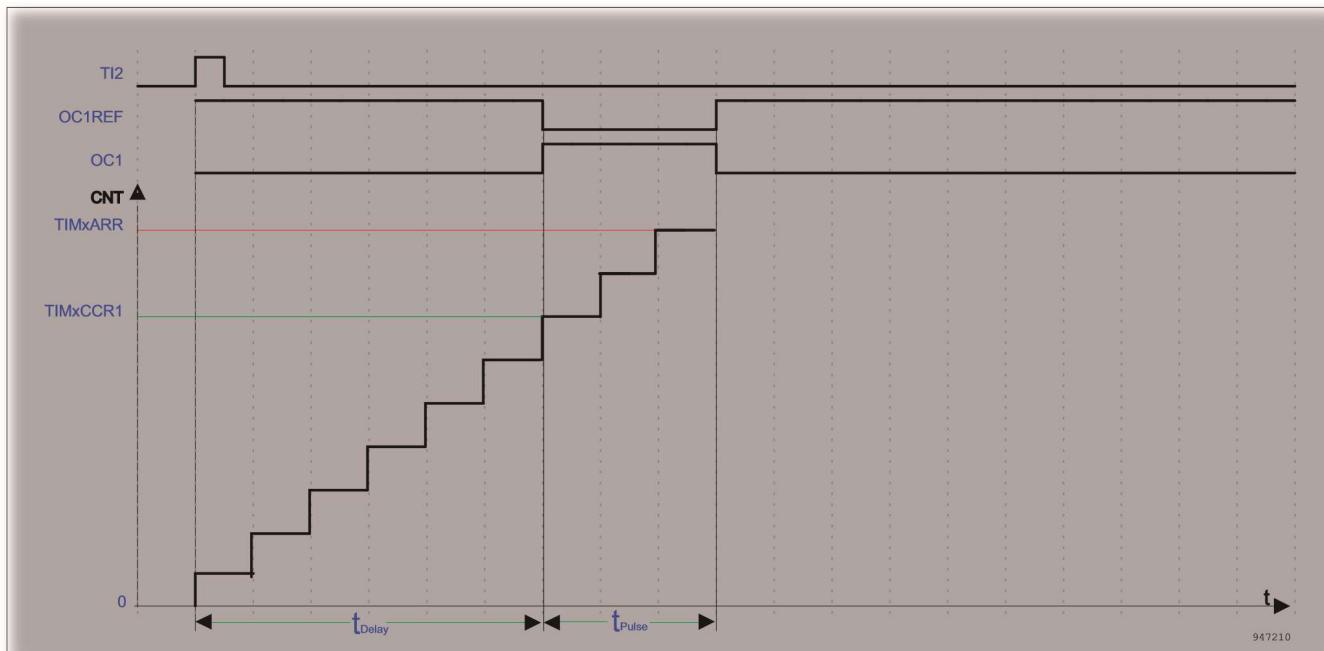


Figure 122. Example of one-pulse mode

For example the user may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S to '01' in the  $\text{TIMx\_CCMR1}$  register.

- TI2FP2 must detect a rising edge, write CC2P to '0' in the TIMx\_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS to '110' in the TIMx\_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{DELAY}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{PULSE}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let us say the user wants to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value.

To do this, enable PWM mode 2 by writing OC1M = 111 in the TIMx\_CCMR1 register. The user can optionally enable the preload registers by writing OC1PE = 1 in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case the compare value must be written in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '1' in this example.

In our example, the DIR and CMS bits in the TIMx\_CR1 register should be low.

User only wants one pulse, so write '1' in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

### **Particular case: OCx fast enable:**

In One-pulse mode, the edge detection on TIx input sets the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{DELAY}$  min we can get.

To output a waveform with the minimum delay, the user can set the OCxFE bit in the TIMx\_CCMRx register. Then OCxREF (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

#### **16.3.11 Clearing the OCxREF signal on an external event**

The OCxREF signal can be cleared by setting the OCCS bit in the TIMx\_SMCR register. The OCxREF signal of a given channel can be pulled low when OCCS = 0 and a high level is applied on the ETRF input (OCxCE enable bit in the corresponding TIMx\_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. The OCxREF signal can be pulled low when OCCS = 1 and a high level is applied on the OCREF\_CLR (output from comparator, COMPx\_CSR[13:10]=1001 for TIM2 and COMPx\_CSR[13:10]=1011 for TIM3) signal.

This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

For example, the OCxREF signal can be connected to an external output. In this case, the ETR must be configured as follow:

- The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx\_SMCR register set to '00'.

- The external clock mode 2 must be disabled: bit ECE of the TIMx\_SMCR register set to '0'.
- The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

The figure below shows the behavior of the OCxREF signal corresponding to different OCxCE values when the ETRF input becomes High. In this example, the timer TIMx is programmed in PWM mode.

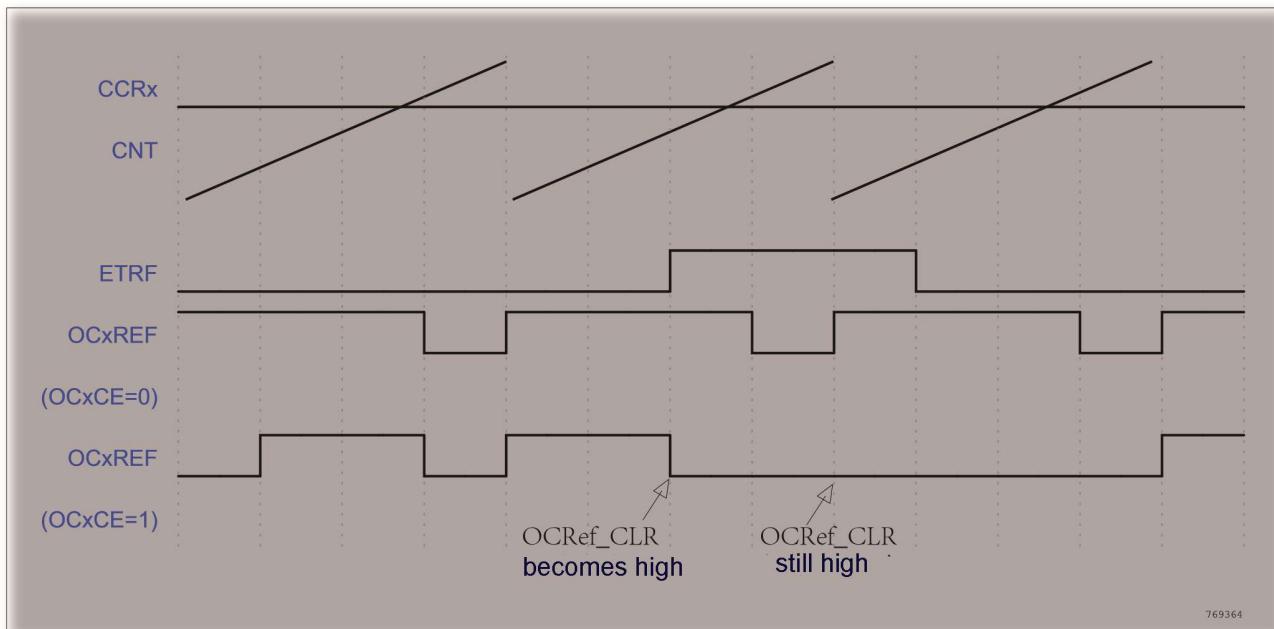


Figure 123. Clearing TIMx OCxREF

### 16.3.12 Encoder interface mode

To select Encoder Interface mode, write SMS to '001' in the TIMx\_SMCR register if the counter is counting on TI2 edges only, SMS to '010' if it is counting on TI1 edges only and SMS to '011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx\_CCER register. When needed, the user can program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to table below. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx\_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence, the counter counts up or down, the DIR bit in the TIMx\_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx\_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the user must configure TIMx\_ARR before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder's

position. The count direction corresponds to the rotation direction of the connected sensor. The following table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 48. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI1FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset. The following figure gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points.

For this example we assume that the configuration is the following:

- C1S = '01' (TIMx\_CCMR1 register, IC1FP1 mapped on TI1).
- CC2S = '01' (TIMx\_CCMR2 register, IC2FP2 mapped on TI2).
- CC1P = '0' (TIMx\_CCER register, IC1FP1 non-inverted, IC1FP1 = TI1).
- CC2P = '0' (TIMx\_CCER register, IC2FP2 non-inverted, IC2FP2 = TI2).
- SMS = '011' (TIMx\_SMCR register, both inputs are active on both rising and falling edges)
- CEN = '1' (TIMx\_CR1 register, counter enabled)

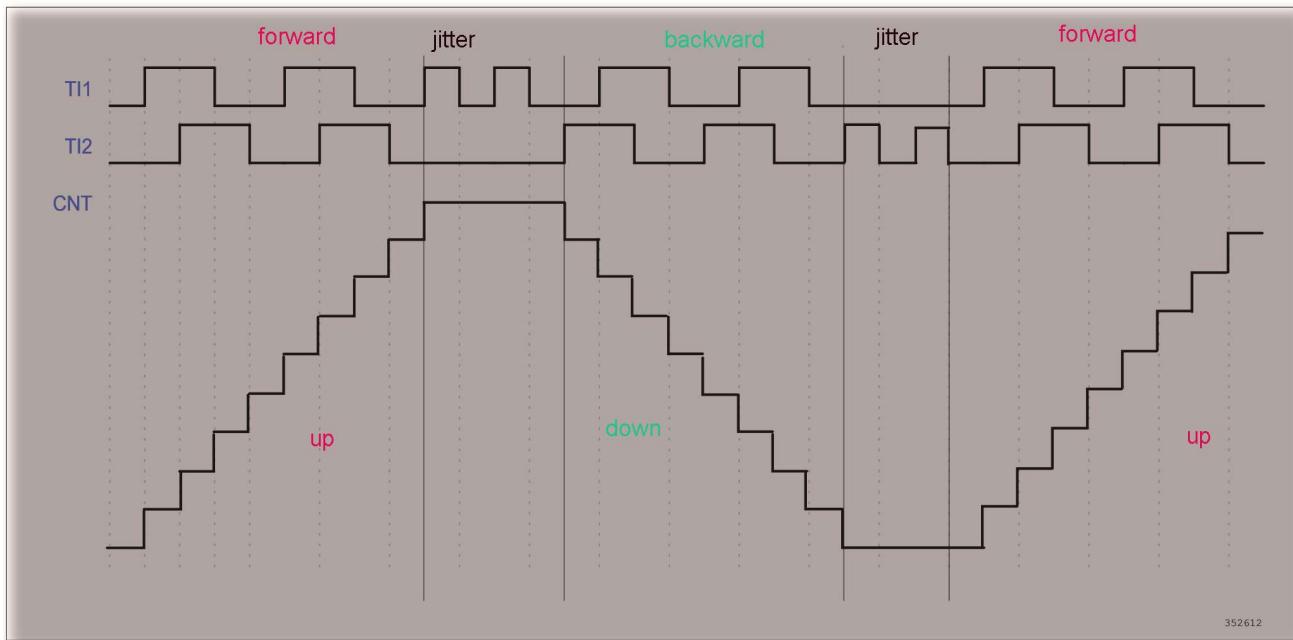


Figure 124. Example of counter operation in encoder interface mode

The following figure gives an example of counter behavior when IC1FP1 polarity is inverted (same configuration as above except CC1P = '1').

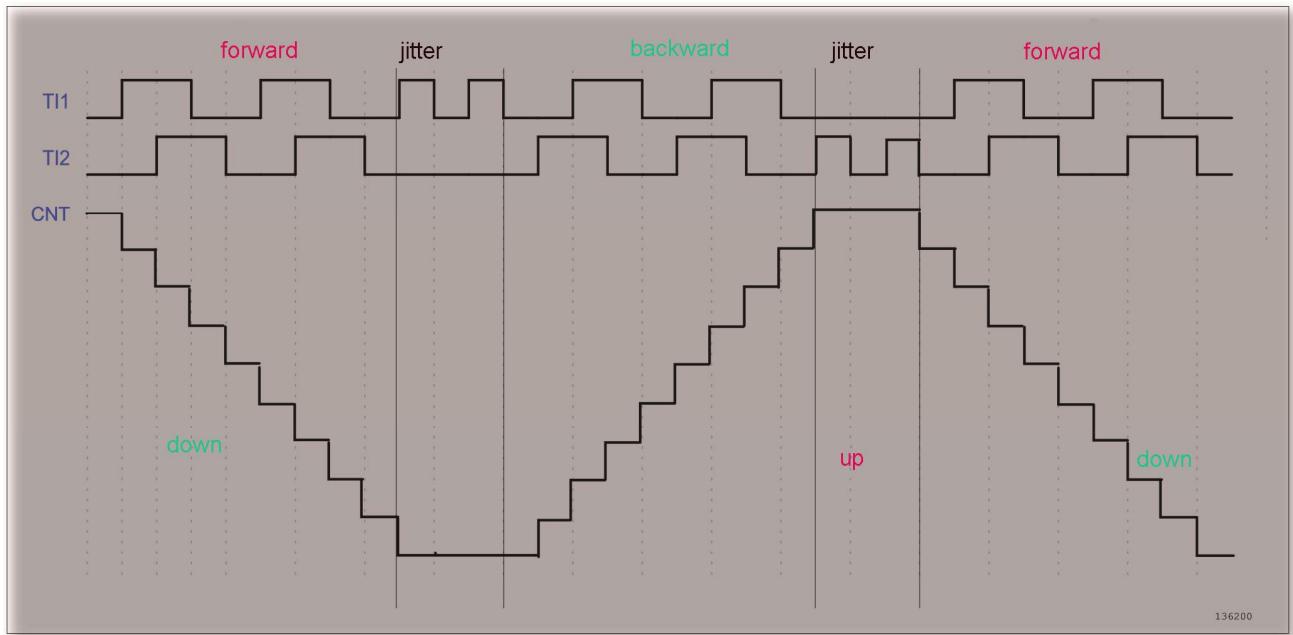


Figure 125. Example of encoder interface mode with IC1FP1 polarity inverted

The timer, when configured in Encoder Interface mode provides information on the sensor's current position. The user can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). It is also possible to read its value through a DMA request generated by a real-time clock.

### 16.3.13 Timer input XOR function

The TI1S bit in the TIMx\_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx\_CH1, TIMx\_CH2 and TIMx\_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. An example of this feature used to interface Hall sensors is given in Section Advanced-Control Timer.

### 16.3.14 TIMx and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

#### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

- In the following example, the upcounter is cleared in response to a rising edge on TI1 input
- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC1S bits select the input capture source only, i.e. CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P = 0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS = 101 in TIMx\_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR = 0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

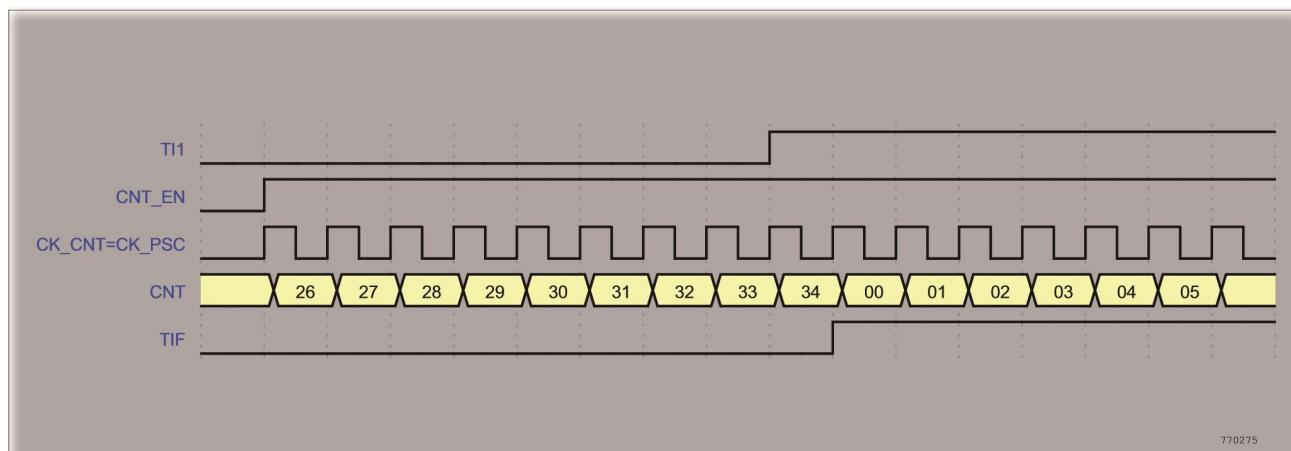


Figure 126. Control circuit in reset mode

#### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so the user does not need to configure it. The CC1S bits select the input capture source only. Write CC1S = 01 in TIMx\_CCMR1 register. Write CC1P = 1 in TIMx\_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS = 101 in TIMx\_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

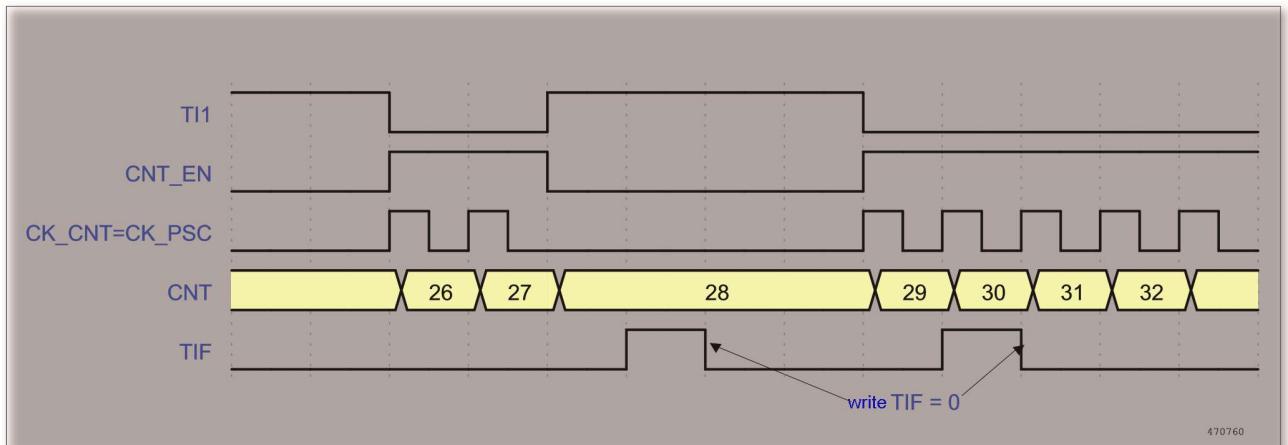


Figure 127. Control circuit in gated mode

### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC2S bits select the input capture source only. Write CC2S = 01 in TIMx\_CCMR1 register. Write CC1P = 1 in TIMx\_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIMx\_SMCR register. Select TI2 as the input source by writing TS = 110 in TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

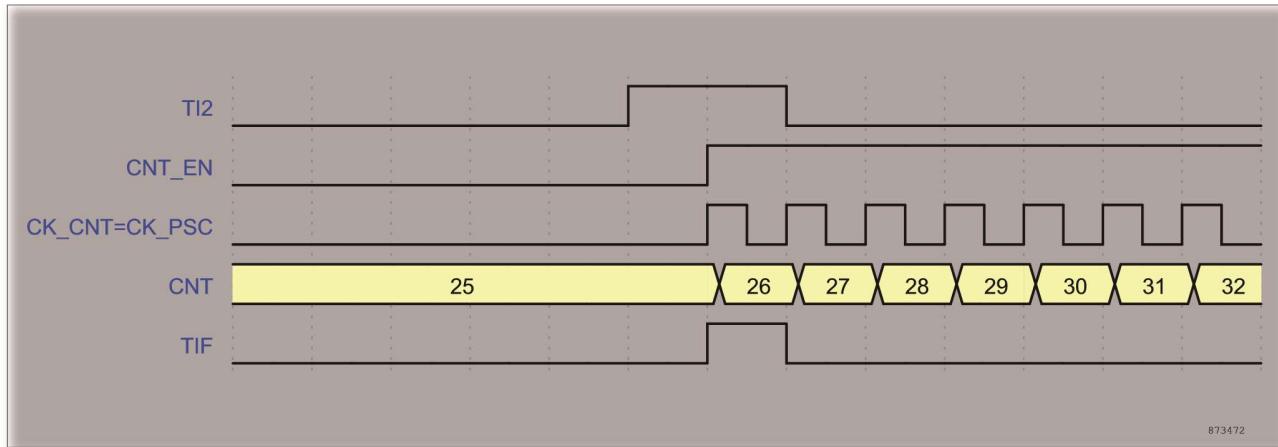


Figure 128. Control circuit in trigger mode

### Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx\_SMCR register. In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

- Configure the external trigger input circuit by programming the TIMx\_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS = 00: prescaler disabled
  - ETP = 0: detection of rising edges on ETR and ECE = 1 to enable the external clock mode 2
- Configure the channel 1 as follows, to detect rising edges on TI:
  - IC1F = 0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S = 01 in TIMx\_CCMR1 register to select only the input capture source.
  - Write CC1P = 0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in trigger mode by writing SMS = 110 in the TIMx\_SMCR register. Select TI1 as the input source by writing TS = 101 in the TIMx\_SMCR register.

A rising edge on TI1 sets the TIF flag. The counter then counts on ETR rising edges. The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

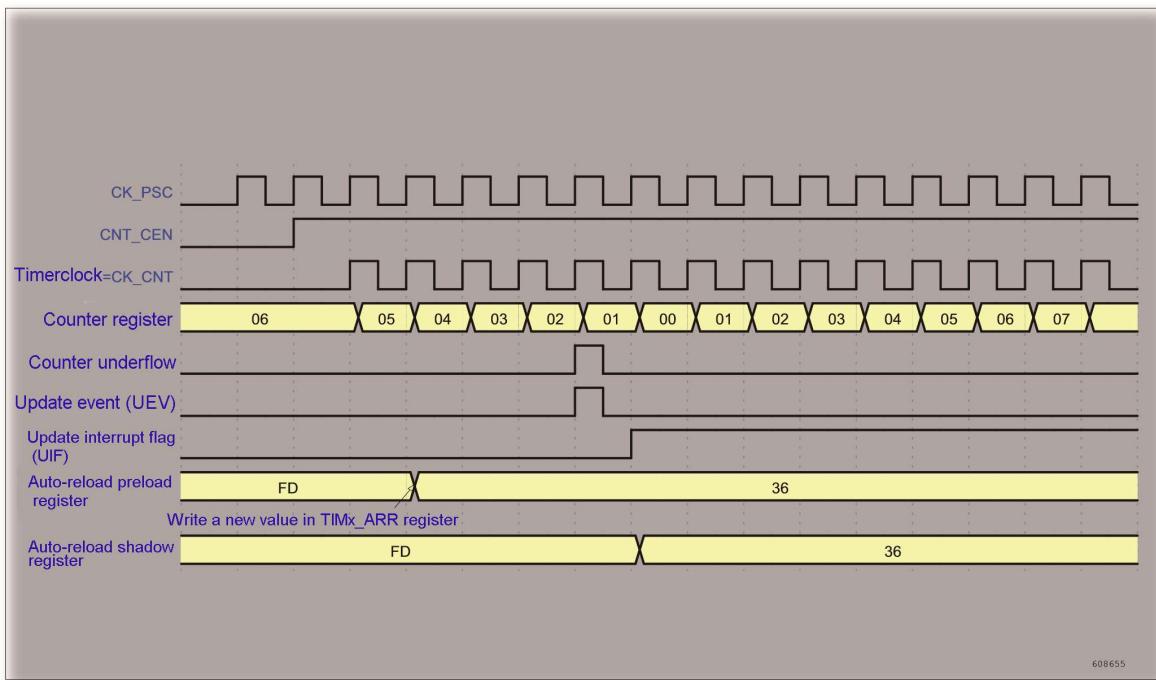


Figure 129. Control circuit in external clock mode 2 + trigger mode

### 16.3.15 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master mode, it can reset, start, stop or clock the counter of another Timer configured in Slave mode.

The figure below presents an overview of the trigger selection and the master mode selection blocks.

#### Using one timer as prescaler for another

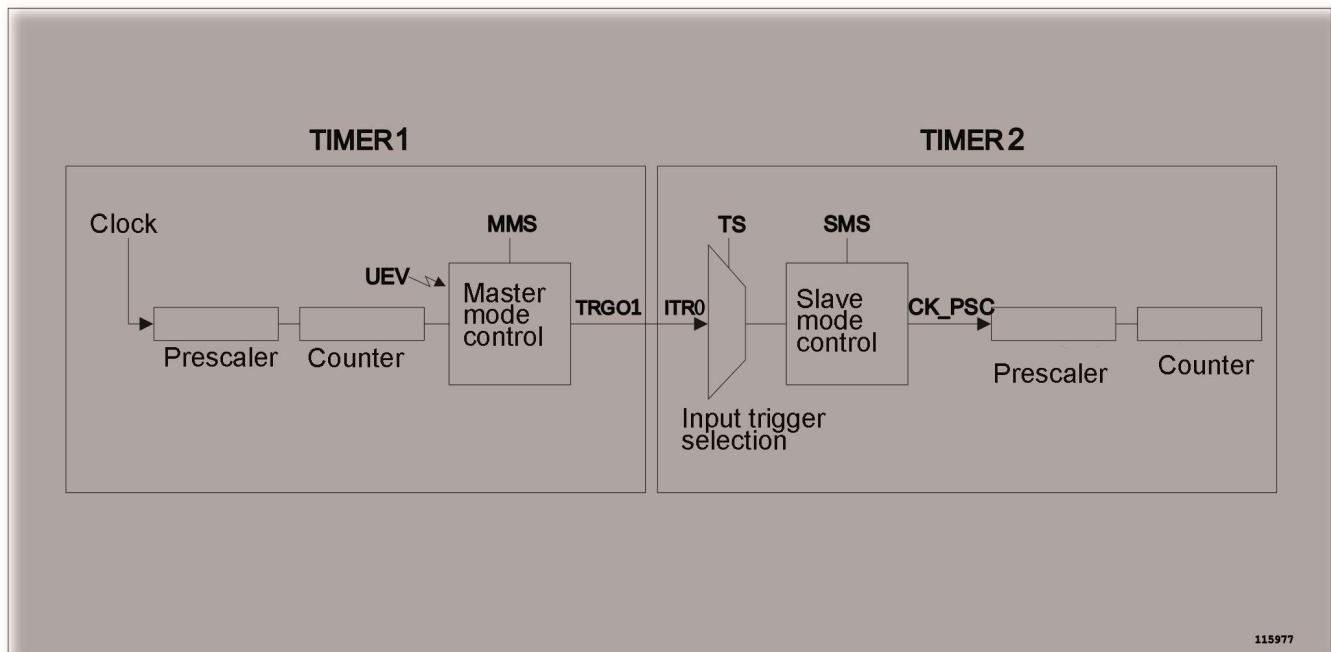


Figure 130. Master/Slave timer example

For example, the user can configure Timer 1 to act as a prescaler for Timer 2. Refer to the above diagram and perform the following operations:

- Configure Timer 1 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS = 010 in the TIM1\_CR2 register, a rising edge is output on TRGO1 each time an update event is generated.
- To connect the TRGO1 output of Timer 1 to Timer 2, Timer 2 must be configured in slave mode using ITR1 as internal trigger. You select this through the TS bits in the TIM2\_SMCR register (writing TS = 000).
- Then you put the slave mode controller in external clock mode 1 (write SMS = 111 in the TIM2\_SMCR register). This causes Timer 2 to be clocked by the rising edge of the periodic Timer 1 trigger signal (which corresponds to the Timer 1 counter overflow).
- Finally both timers must be enabled by setting their respective CEN bits (TIMx\_CR1 register).

Note: If OCx is selected on Timer 1 as trigger output (MMS = 1xx), its rising edge is used to clock the counter of Timer 2.

### Using one timer to enable another timer

In this example, we control the enable of Timer 2 with the output compare of Timer 1.

Refer to the diagram below. Timer 2 counts on the divided internal clock only when OC1REF of Timer 1 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_CNT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

- Configure Timer 1 master mode to send its Output compare 1 Reference (OC1REF) signal as trigger output (MMS = 100 in the TIM1\_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1\_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS = 001 in the TIM2\_SMCR register).
- Configure Timer 2 in gated mode (SMS = 101 in TIM2\_SMCR register).
- Enable Timer 2 by writing '1' in the CEN bit (TIM2\_CR1 register).
- Enable Timer 1 by writing '1' in the CEN bit (TIM1\_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the Timer 2 counter enable signal.

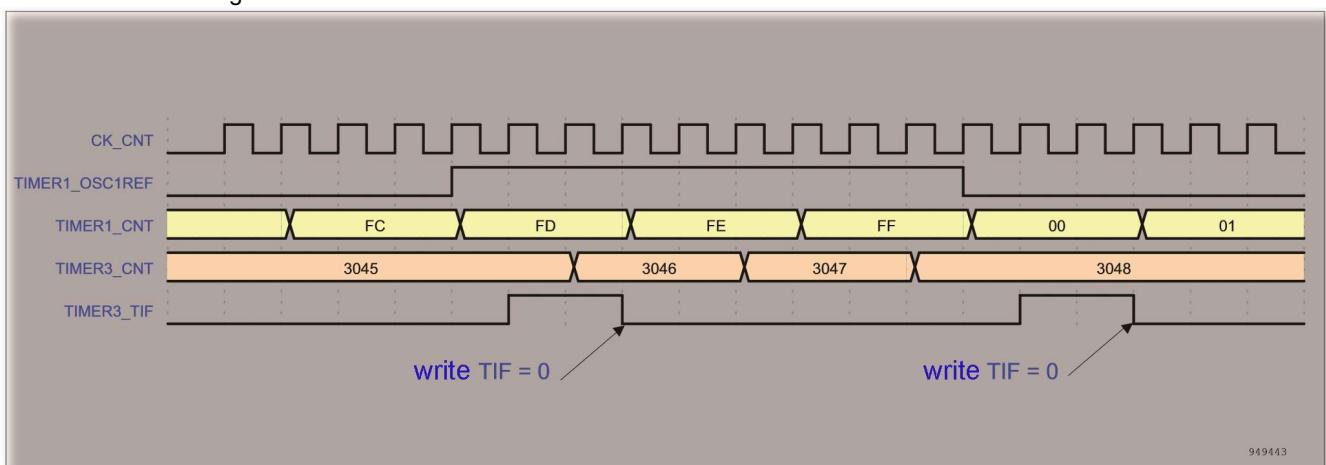


Figure 131.Gating timer 2 with OC1REF of timer 1

In the above example, the Timer 2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given

value by resetting both timers before starting Timer 1. You can then write any value you want in the timer counters. The timers can easily be reset by writing the UG bit in the TIMx\_EGR registers.

In the next example, we synchronize Timer 1 and Timer 2. Timer 1 is the master and starts from 0. Timer 2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer 2 stops when Timer 1 is disabled by writing '0' to the CEN bit in the TIM1\_CR1 register.

- Configure Timer 1 master mode to send its Output compare 1 Reference (OC1REF) signal as trigger output (MMS = 100 in the TIM1\_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1\_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS = 000 in the TIM2\_SMCR register).
- Configure Timer 2 in gated mode (SMS = 101 in TIM2\_SMCR register).
- Reset Timer 1 by writing '1' in UG bit (TIM1\_EGR register).
- Reset Timer 2 by writing '1' in UG bit (TIM2\_EGR register).
- Initialize Timer 2 to 0xE7 by writing '0xE7' in the timer 2 counter (TIM2\_CNT).
- Enable Timer 2 by writing '1' in the CEN bit (TIM2\_CR1 register).
- Start Timer 1 by writing '1' in the CEN bit (TIM1\_CR1 register).
- Stop Timer 1 by writing '0' in the CEN bit (TIM1\_CR1 register).

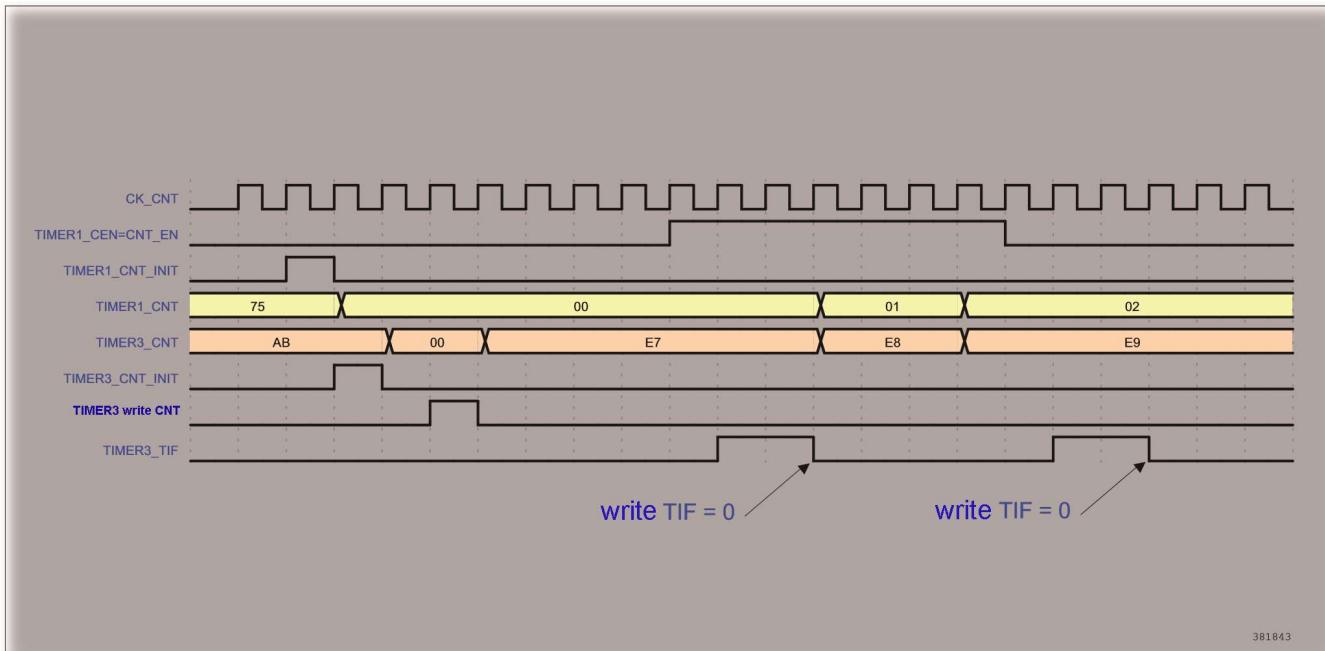


Figure 132. Gating timer 2 with enable of timer 1

### Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to the diagram below. Timer 2 starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal, its CEN bit is automatically set to '1' and the counter counts until we write '0' to the CEN bit in the TIM2\_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

- Configure Timer 1 master mode to send its Update Event (UEV) as trigger output (MMS = 010 in the TIM1\_CR2 register).

- Configure the Timer 1 period (TIM1\_ARR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS = 000 in the TIM2\_SMCR register).
- Configure Timer 2 in trigger mode (SMS = 110 in TIM2\_SMCR register).
- Start Timer 1 by writing '1' in the CEN bit (TIM1\_CR1 register).

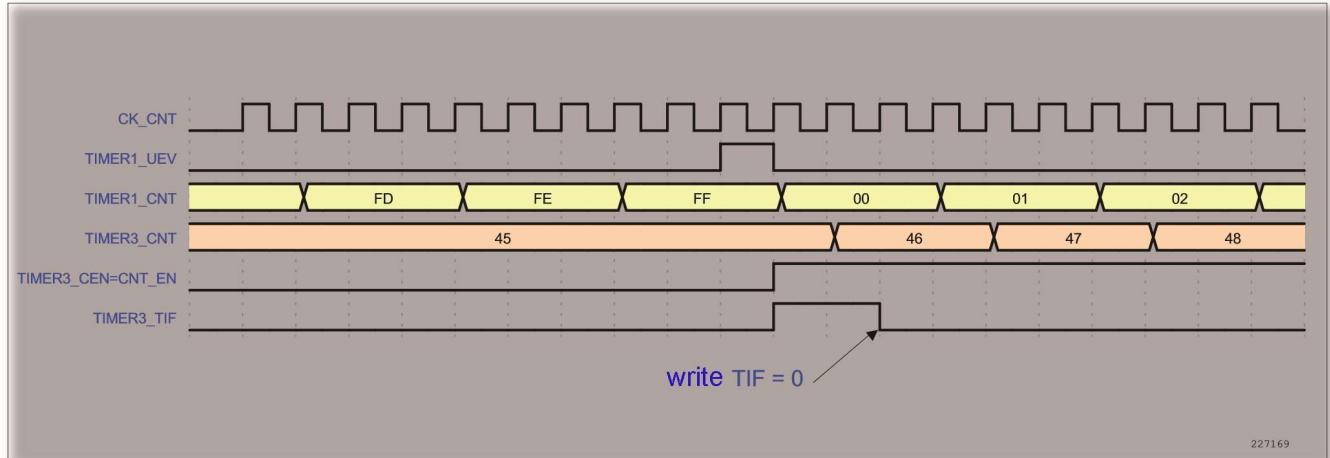


Figure 133. Triggering timer 2 with update of timer 1

As in the previous example, the user can initialize both counters before starting counting. The diagram below shows the behavior with the same configuration as 0 but in trigger mode instead of gated mode (SMS = 110 in the TIM2\_SMCR register).

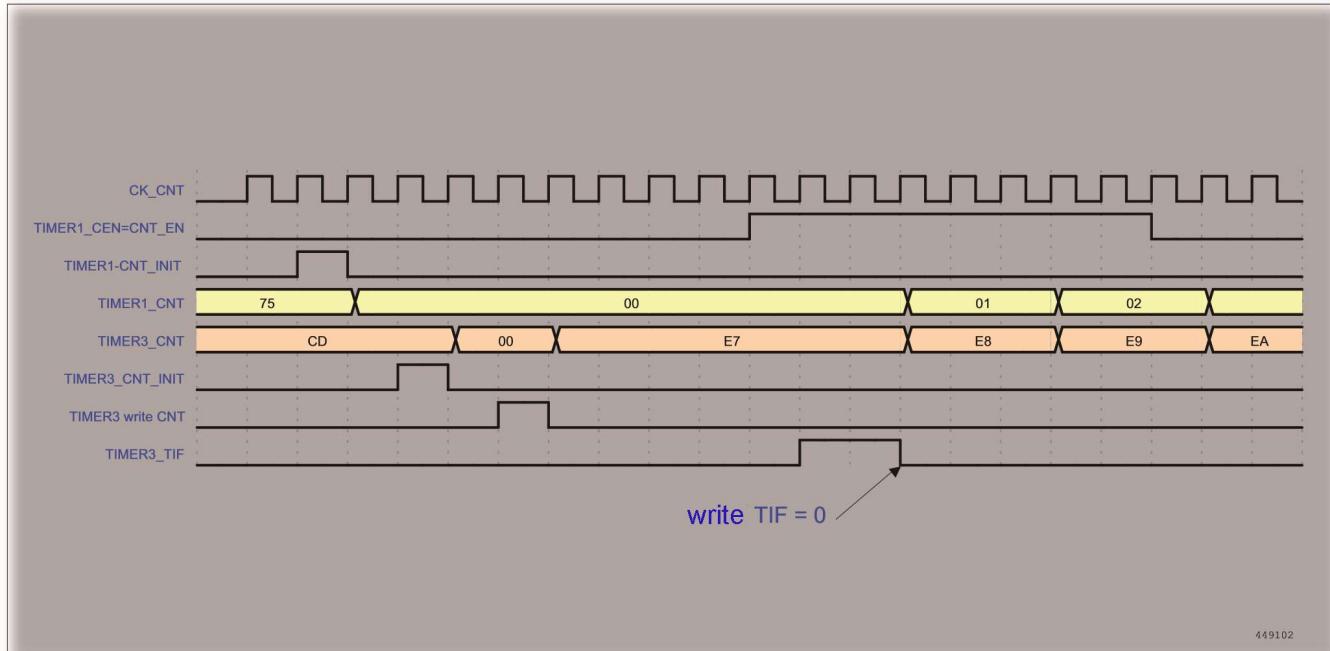


Figure 134. Triggering timer 2 with enable of timer 1

### Using one timer as prescaler for another timer

In this example, we use Timer 1 as prescaler for Timer 2. We configure as follows:

- Configure Timer 1 master mode to send its Update Event (UEV) as trigger output (MMS = 010 in the TIM1\_CR2 register). Then a periodic signal can be generated at each counter overflow.
- Configure the Timer 1 period (TIM1\_ARR register).

- Configure Timer 2 to get the input trigger from Timer 1 (TS = 000 in the TIM2\_SMCR register).
- Configure Timer 2 in external clock mode (SMS = 111 in TIM2\_SMCR register).
- Start Timer 2 by writing '1' in the CEN bit (TIM1\_CR2 register).
- Start Timer 1 by writing '1' in the CEN bit (TIM1\_CR1 register).

### Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of Timer 1 when its TI1 input rises, and the enable of Timer 2 with the enable of Timer 1. To ensure the counters are aligned, Timer 1 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to Timer 2):

- Configure Timer 1 master mode to send its Enable as trigger output (MMS = 001 in the TIM1\_CR2 register).
- Configure Timer 1 slave mode to get the input trigger from TI1 (TS = 100 in the TIM1\_SMCR register).
- Configure Timer 1 in trigger mode (SMS = 110 in the TIM1\_SMCR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS = 000 in the TIM2\_SMCR register).
- Configure Timer 2 in trigger mode (SMS = 110 in TIM2\_SMCR register).

When a rising edge occurs on TI1 (Timer 1), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but you can easily insert an offset between them by writing any of the counter registers (TIMx\_CNT). You can see from the following diagram that there is a delay between CNT\_EN and CK\_PSC on Timer 1 in the master/slave mode.

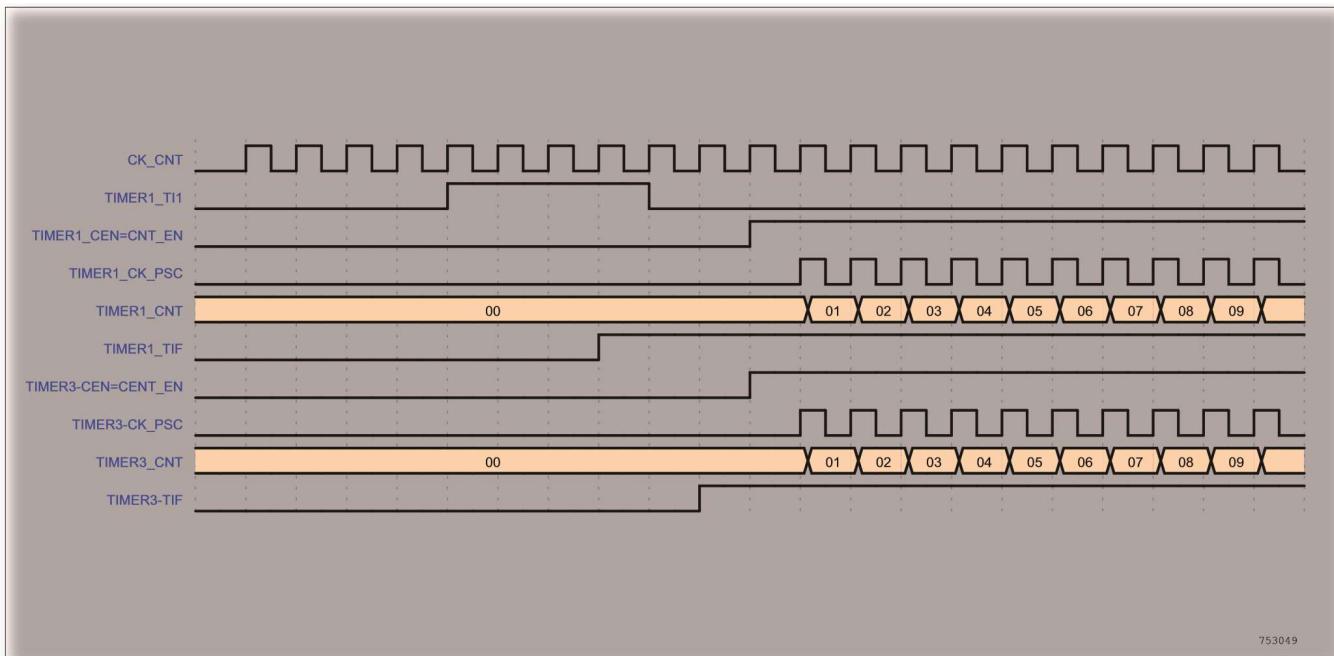


Figure 135. Triggering timer 1 and 2 with timer 1 TI1 input

#### 16.3.16 Debug mode

When the microcontroller enters debug mode (CPU core - halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIMx\_STOP configuration bit in DBG module. For more details, refer to debug module section.

## 16.4 TIMx register description

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

16-bit general-purpose timer (TIM3)  
Table 49. Overview of TIMx registers

Offset	Acronym	Register Name	Reset	Section
0x00	TIMx_CR1	Control register 1	0x00000000	Section 16.4.1
0x04	TIMx_CR2	Control register 2	0x00000000	Section 16.4.2
0x08	TIMx_SMCR	Slave mode control register	0x00000000	Section 16.4.3
0x0C	TIMx_DIER	DMA/interrupt enable register	0x00000000	Section 16.4.4
0x10	TIMx_SR	Status register	0x00000000	Section 16.4.5
0x14	TIMx_EGR	Event generate register	0x00000000	Section 16.4.6
0x18	TIMx_CCMR1	Capture/compare mode register 1	0x00000000	Section 16.4.7
0x1C	TIMx_CCMR2	Capture/compare mode register 2	0x00000000	Section 16.4.8
0x20	TIMx_CCER	Capture/compare enable register	0x00000000	Section 16.4.9
0x24	TIMx_CNT	Counter	0x00000000	Section 16.4.10
0x28	TIMx_PSC	Prescaler	0x00000000	Section 16.4.11
0x2C	TIMx_ARR	Auto-reload register	0x00000000	Section 16.4.12
0x34	TIMx_CCR1	Capture/compare register 1	0x00000000	Section 16.4.13
0x38	TIMx_CCR2	Capture/compare register 2	0x00000000	Section 16.4.14
0x3C	TIMx_CCR3	Capture/compare register 3	0x00000000	Section 16.4.15
0x40	TIMx_CCR4	Capture/compare register 4	0x00000000	Section 16.4.16
0x48	TIMx_DCR	DMA control register	0x00000000	Section 16.4.17
0x4C	TIMx_DMAR	DMA address for full transfer	0x00000000	Section 16.4.18

### 16.4.1 Control register 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CKD	ARPE	CMS	DIR	OPM	URS	UDIS	CEN			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:10	Reserved			Reserved, read as 0
9:8	CKD	rw	0x00	<p>Clock division</p> <p>These 2 bits indicate the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock used by the dead-time generators and the digital filters (ETR, TIx).</p> <p>00: <math>t_{DTS} = t_{CK\_INT}</math></p> <p>01: <math>t_{DTS} = 2 \times t_{CK\_INT}</math></p> <p>10: <math>t_{DTS} = 4 \times t_{CK\_INT}</math></p> <p>11: Reserved, do not program this value</p>

Bit	Field	Type	Reset	Description
7	ARPE	rw	0x00	<p>Auto-reload preload enable</p> <p>0: TIMx_ARR register is not buffered</p> <p>1: TIMx_ARR register is buffered</p>
6:5	CMS	rw	0x00	<p>Center-aligned mode selection</p> <p>00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).</p> <p>01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.</p> <p>10: Center-aligned mode 2. The counter counts up and down alternatively. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.</p> <p>11: Center-aligned mode 3. The counter counts up and down alternatively. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.</p> <p>Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)</p>
4	DIR	rw	0x00	<p>Direction</p> <p>0: Counter used as upcounter</p> <p>1: Counter used as downcounter</p> <p>Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.</p>
3	OPM	rw	0x00	<p>One-pulse mode</p> <p>0: Counter is not stopped at update event</p> <p>1: Counter stops counting at the next update event (clearing the bit CEN).</p>

2	URS	rw	0x00	<p>Update request source</p> <p>This bit is set by software to select the UEV event sources.</p> <p>0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:</p> <ul style="list-style-type: none"> <li>- Counter overflow/underflow</li> <li>- Setting the UG bit</li> <li>- Update generation through the slave mode controller</li> </ul> <p>1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.</p>
---	-----	----	------	--

Bit	Field	Type	Reset	Description
1	UDIS	rw	0x00	<p>Update disabled</p> <p>This bit is set by software to enable/disable UEV event generation.</p> <p>0: UEV enabled. The Update (UEV) event is generated by one of the following events:</p> <ul style="list-style-type: none"> <li>- Counter overflow/underflow</li> <li>- Setting the UG bit to '1'</li> <li>- Update generation through the slave mode controller with the subsequent update of shadow registers</li> </ul> <p>1: UEV disabled. The Update event is not generated, shadow registers keep their values (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set to '1' or if a hardware reset is received from the slave mode controller.</p>
0	CEN	rw	0x00	<p>Counter enable</p> <p>0: Counter disabled</p> <p>1: Counter enabled</p> <p>Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.</p>

#### 16.4.2 Control register 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS		CCDS	Reserved			
								rw	rw	rw	rw				

Bit	Field	Type	Reset	Description
15: 8	Reserved			Reserved, read as 0

7	TI1S	rw	0x00	TI1 selection 0: The TIMx_CH1 pin is connected to TI1 input 1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
---	------	----	------	--

Bit	Field	Type	Reset	Description
6:4	MMS	rw	0x00	<p>Master mode selection</p> <p>These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:</p> <ul style="list-style-type: none"> <li>000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.</li> <li>001: Enable - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).</li> <li>010: Update - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.</li> <li>011: Compare Pulse - The trigger output (TRGO) sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.</li> <li>100: Compare - OC1REF signal is used as trigger output (TRGO)</li> <li>101: Compare - OC2REF signal is used as trigger output (TRGO)</li> <li>110: Compare - OC3REF signal is used as trigger output (TRGO)</li> <li>111: Compare - OC4REF signal is used as trigger output (TRGO)</li> </ul>

3	CCDS	rw	0x00	Capture/compare DMA selection 0: CCx DMA request sent when CCx event occurs 1: CCx DMA requests sent when update event occurs
2: 0	Reserved			Reserved and read as 0.

### 16.4.3 Slave mode control register (TIMx\_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF		MSM		TS		OCCS		SMS			

Bit	Field	Type	Reset	Description
15	ETP	rw	0x00	External trigger polarity  This bit selects whether ETR or inverted ETR is used for trigger operations. 0: ETR is non-inverted, active at high level or rising edge 1: ETR is inverted, active at low level or falling edge
14	ECE	rw	0x00	External clock enable  This bit enables External clock mode 2. 0: External clock mode 2 disabled 1: External clock mode 2 enabled. The counter is clocked by any active rising edge on the ETRF signal. Note 1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS = 111 and TS = 111). Note 2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111). Note 3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.
13:12	ETPS	rw	0x00	External trigger prescaler  External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks. 00: Prescaler OFF 01: ETRP frequency divided by 2 10: ETRP frequency divided by 4 11: ETRP frequency divided by 8

Bit	Field	Type	Reset	Description
11:8	ETF	rw	0x00	<p>External trigger filter</p> <p>This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:</p> <ul style="list-style-type: none"> <li>0000: No filter, sampling is done at <math>f_{DTS}</math></li> <li>0001: <math>f_{SAMPLING}=f_{CK\_INT}</math>, N = 2</li> <li>0010: <math>f_{SAMPLING}=f_{CK\_INT}</math>, N = 4</li> <li>0011: <math>f_{SAMPLING}=f_{CK\_INT}</math>, N = 8</li> <li>0100: <math>f_{SAMPLING}=f_{DTS}/2</math>, N = 6</li> <li>0101: <math>f_{SAMPLING}=f_{DTS}/2</math>, N = 8</li> <li>0110: <math>f_{SAMPLING}=f_{DTS}/4</math>, N = 6</li> <li>0111: <math>f_{SAMPLING}=f_{DTS}/4</math>, N = 8</li> <li>1000: <math>f_{SAMPLING}=f_{DTS}/8</math>, N = 6</li> <li>1001: <math>f_{SAMPLING}=f_{DTS}/8</math>, N = 8</li> <li>1010: <math>f_{SAMPLING}=f_{DTS}/16</math>, N = 5</li> <li>1011: <math>f_{SAMPLING}=f_{DTS}/16</math>, N = 6</li> <li>1100: <math>f_{SAMPLING}=f_{DTS}/16</math>, N = 8</li> <li>1101: <math>f_{SAMPLING}=f_{DTS}/32</math>, N = 5</li> <li>1110: <math>f_{SAMPLING}=f_{DTS}/32</math>, N = 6</li> <li>1111: <math>f_{SAMPLING}=f_{DTS}/32</math>, N = 8</li> </ul>
6:4	TS	rw	0x00	<p>Trigger selection</p> <p>This bit-field selects the trigger input to be used to synchronize the counter.</p> <ul style="list-style-type: none"> <li>000: Internal Trigger 0 (ITR0)</li> <li>001: Internal Trigger 1 (ITR1)</li> </ul>

- 010: Internal Trigger 2 (ITR2)
- 011: Internal Trigger 3 (ITR3)
- 100: TI1 Edge Detector (TI1F\_ED)
- 101: Filtered Timer Input 1 (TI1FP1)
- 110: Filtered Timer Input 2 (TI2FP2)
- 111: External Trigger input (ETRF)

See the following table for more details on ITRx.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

3	OCCS	rw	0x00	<p>Output compare clear selection This bit is used to clear the comparator output in the PWM mode. 1: The comparator output is considered as the clear signal 0: The external trigger input is considered as the clear signal.</p>
---	------	----	------	--

Bit	Field	Type	Reset	Description
2:0	SMS	rw	0x00	<p>Slave mode selection When external signals are selected, the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control register description). 000: Slave mode disabled - if CEN = 1, then the prescaler is clocked directly by the internal clock. 001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level. 010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level. 011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input. 100: Reset mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers. 101: Gated mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled. 110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled. 111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter. Note: The gated mode must not be used if TI1F_EN is selected as the trigger input (TS = 100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.</p>

Table 50. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	None	TIM3	None
TIM3	TIM1	TIM2	None	None

**16.4.4 DMA/Interrupt enable register (TIMx\_DIER)**

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE

rw                rw

Bit	Field	Type	Reset	Description
15	Reserved			Reserved, read as 0
14	TDE	rw	0x00	Trigger DMA request enable 0: Trigger DMA request disabled 1: Trigger DMA request enabled
13	Reserved			Reserved and read as 0.
12	CC4DE	rw	0x00	Capture/Compare 4 DMA request enable 0: Capture/Compare 4 DMA request disabled 1: Capture/Compare 4 DMA request enabled
11	CC3DE	rw	0x00	Capture/Compare 3 DMA request enable 0: Capture/Compare 3 DMA request disabled 1: Capture/Compare 3 DMA request enabled
10	CC2DE	rw	0x00	Capture/Compare 2 DMA request enable 0: Capture/Compare 2 DMA request disabled 1: Capture/Compare 2 DMA request enabled
9	CC1DE	rw	0x00	Capture/Compare 1 DMA request enable 0: Capture/Compare 1 DMA request disabled 1: Capture/Compare 1 DMA request enabled
8	UDE	rw	0x00	Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled
7	Reserved			Reserved and read as 0.

6	TIE	rw	0x00	Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
5	Reserved			Reserved and read as 0.

Bit	Field	Type	Reset	Description
4	CC4IE	rw	0x00	Capture/Compare 4 interrupt enable 0: Capture/Compare 4 interrupt disabled 1: Capture/Compare 4 interrupt enabled
3	CC3IE	rw	0x00	Capture/Compare 3 interrupt enable 0: Capture/Compare 3 interrupt disabled 1: Capture/Compare 3 interrupt enabled
2	CC2IE	rw	0x00	Capture/Compare 2 interrupt enable 0: Capture/Compare 2 interrupt disabled 1: Capture/Compare 2 interrupt enabled
1	CC1IE	rw	0x00	Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt disabled 1: Capture/Compare 1 interrupt enabled
0	UIE	rw	0x00	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

#### 16.4.5 Status register (TIMx\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		CC4OF	CC3OF	CC2OF	CC1OF		Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF	
		rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	

Bit	Field	Type	Reset	Description
15:13	Reserved			Reserved, read as 0
12	CC4OF	rc_w0	0x00	Capture/Compare 4 overcapture flag Refer to CC1OF description.
11	CC3OF	rc_w0	0x00	Capture/Compare 3 overcapture flag Refer to CC1OF description.
10	CC2OF	rc_w0	0x00	Capture/Compare 2 overcapture flag Refer to CC1OF description.

Bit	Field	Type	Reset	Description
9	CC1OF	rc_w0	0x00	<p>Capture/Compare 1 overcapture flag</p> <p>This flag is set to '1' by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.</p> <p>0: No overcapture has been detected.</p> <p>1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set to 1.</p>
8: 7	Reserved			Reserved and read as 0.
6	TIF	rc_w0	0x00	<p>Trigger interrupt flag</p> <p>This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software.</p> <p>0: No trigger event occurred.</p> <p>1: Trigger interrupt pending.</p>
5	Reserved			Reserved and read as 0.
4	CC4IF	rc_w0	0x00	<p>Capture/Compare 4 interrupt flag</p> <p>Refer to CC1IF description.</p>
3	CC3IF	rc_w0	0x00	<p>Capture/Compare 3 interrupt flag</p> <p>Refer to CC1IF description.</p>
2	CC2IF	rc_w0	0x00	<p>Capture/Compare 2 interrupt flag</p> <p>Refer to CC1IF description.</p>
1	CC1IF	rc_w0	0x00	<p>Capture/Compare 1 interrupt flag</p> <p>If channel CC1 is configured as output:</p> <p>This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.</p> <p>0: No match.</p> <p>1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register.</p> <p>If channel CC1 is configured as input:</p> <p>This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.</p> <p>0: No input capture occurred</p> <p>1: The counter value has been captured in (copied to) TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)</p>

Bit	Field	Type	Reset	Description
0	UIF	rc_w0	0x00	<p>Update interrupt flag This bit is set by hardware on an update event. It is cleared by software.</p> <p>0: No update occurred. 1: Update event pending. This bit is set by hardware when the registers are updated:</p> <ul style="list-style-type: none"> <li>- At overflow or underflow regarding the repetition downcounter value (update if REP_CNT = 0) and if the UDIS = 0 in the TIMx_CR1 register.</li> <li>- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.</li> <li>- When CNT is reinitialized by a trigger event (refer to the synchronous control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.</li> </ul>

#### 16.4.6 Event generate register (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							Res.		TG	COMG	CC4G	CC3G	CC2G	CC1G	UG

Bit	Field	Type	Reset	Description
15:7	Reserved			Reserved, read as 0
6	TG	w	0x00	<p>Trigger generation This bit is set by software in order to generate a brake event, it is automatically cleared by hardware.</p> <p>0: No action 1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA can occur if enabled.</p>
5	COMG	w	0x00	<p>Capture/Compare control update generation This bit can be set by software, it is automatically cleared by hardware.</p> <p>0: No action 1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits. Note: This bit acts only on channels that have a complementary output.</p>
4	CC4G	w	0x00	Capture/Compare 4 generation Refer to CC1G description.

3	CC3G	w	0x00	Capture/Compare 3 generation Refer to CC1G description.
2	CC2G	w	0x00	Capture/Compare 2 generation Refer to CC1G description.
Bit	Field	Type	Reset	Description
1	CC1G	w	0x00	<p>Capture/Compare 1 generation This bit is set by software in order to generate a capture/compare event, it is automatically cleared by hardware.</p> <p>0: No action 1: A capture/compare event is generated on channel CC1: If channel CC1 is configured as output: CC1IF flag is set, corresponding interrupt or DMA request is sent if enabled. If channel CC1 is configured as input: The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.</p>
0	UG	w	0x00	<p>Update generation This bit can be set by software, it is automatically cleared by hardware.</p> <p>0: No action 1: Reinitialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR = 1 (downcounting).</p>

#### 16.4.7 Capture/compare mode register 1 (TIMx\_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. Take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M		OC2PE	OC2FE	CC2S	OC1CE	OC1M		OC1PE	OC1FE	CC1S				
IC2F		IC2PSC				IC1F		IC1PSC							

rw rw

### Output Compare mode:

Bit	Field	Type	Reset	Description
15	OC2CE	rw	0x00	Output compare 2 clear enable
14:12	OC2M	rw	0x00	Output compare 2 mode
11	OC2PE	rw	0x00	Output compare 2 preload enable
10	OC2FE	rw	0x00	Output compare 2 fast enable
9:8	CC2S	rw	0x00	Capture/Compare 2 selection  This bit-field defines the direction of the channel (input/output) as well as the used input. 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register). Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).
7	OC1CE	rw	0x00	Output compare 1 clear enable  0: OC1REF is not affected by the ETRF input 1: OC1REF is cleared as soon as a high level is detected on ETRF input

Bit	Field	Type	Reset	Description
6:4	OC1M	rw	0x00	<p>Output compare 1 mode</p> <p>These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.</p> <p>000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the OC1REF.</p> <p>001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).</p> <p>010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).</p> <p>011: Toggle. OC1REF level toggles when TIMx_CCR1=TIMx_CNT.</p> <p>100: Force inactive level. OC1REF is forced low.</p> <p>101: Force active level. OC1REF is forced high.</p> <p>110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT &lt; TIMx_CCR1, else inactive. In downcounting, channel 1 is inactive (OC1REF = 0) as long as TIMx_CNT &gt; TIMx_CCR1, else active (OC1REF = 1).</p> <p>111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT &lt; TIMx_CCR1, else active. In downcounting, channel 1 is active as long as TIMx_CNT &gt; TIMx_CCR1, else inactive.</p> <p>Note 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output). Note 2: In PWM mode 1 or 2, the OC1REF level changes only when the result of the comparison changes or when the output compare</p>

mode switches from “frozen” mode to “PWM” mode.

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

3	OC1PE	rw	0x00	<p>Output compare 1 preload enable</p> <p>0: Preload function of TIMx_CCR1 register disabled. TIMx_CCR1 register can be written at anytime, the new value is taken in account immediately.</p> <p>1: Preload function of TIMx_CCR1 register enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.</p> <p>Note 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).</p> <p>Note 2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.</p>
2	OC1FE	rw	0x00	<p>Output compare 1 fast enable</p> <p>This bit is used to accelerate the effect of an event on the trigger in input on the CC output.</p> <p>0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.</p> <p>1: An active edge on the trigger input acts like a compare match.</p> <p>Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles.</p> <p>OCFE acts only if the channel is configured in PWM1 or PWM2 mode.</p>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC1 channel is configured as output</li> <li>01: CC1 channel is configured as input, IC1 is mapped on TI1</li> <li>10: CC1 channel is configured as input, IC1 is mapped on TI2</li> <li>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)</li> </ul> <p>Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).</p>

### Input capture mode:

Bit	Field	Type	Reset	Description
15:12	IC2F	rw	0x00	Input capture 2 filter
11:10	IC2PSC	rw	0x00	Input capture 2 prescaler
9:8	CC2S	rw	0x00	Capture/Compare 2 selection  This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register). Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).
7:4	IC1F	rw	0x00	Input capture 1 filter  This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to generate a transition on the output: 0000: No filter, sampling is done at $f_{DTS}$ 1000: $f_{SAMPLING} = f_{DTS}/8$ , N = 6 0001: $f_{SAMPLING} = f_{CK\_INT}$ , N = 2 1001: $f_{SAMPLING} = f_{DTS}/8$ , N = 8 0010: $f_{SAMPLING} = f_{CK\_INT}$ , N = 4 1010: $f_{SAMPLING} = f_{DTS}/16$ , N = 5 0011: $f_{SAMPLING} = f_{CK\_INT}$ , N = 8 1011: $f_{SAMPLING} = f_{DTS}/16$ , N = 6 0100: $f_{SAMPLING} = f_{DTS}/2$ , N = 6 1100: $f_{SAMPLING} = f_{DTS}/16$ , N = 8 0101: $f_{SAMPLING} = f_{DTS}/2$ , N = 8 1101: $f_{SAMPLING} = f_{DTS}/32$ , N = 5 0110: $f_{SAMPLING} = f_{DTS}/4$ , N = 6 1110: $f_{SAMPLING} = f_{DTS}/32$ , N = 6 0111: $f_{SAMPLING} = f_{DTS}/4$ , N = 8 1111: $f_{SAMPLING} = f_{DTS}/32$ , N = 8
Bit	Field	Type	Reset	Description

3:2	IC1PSC	rw	0x00	<p>Input capture 1 prescaler</p> <p>This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).</p> <p>The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).</p> <p>00: no prescaler, capture is done each time an edge is detected on the capture input</p> <p>01: capture is done once every 2 events</p> <p>10: capture is done once every 4 events</p> <p>11: capture is done once every 8 events</p>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <p>00: CC1 channel is configured as output</p> <p>01: CC1 channel is configured as input, IC1 is mapped on TI1</p> <p>10: CC1 channel is configured as input, IC1 is mapped on TI2</p> <p>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)</p> <p>Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).</p>

## Output Compare mode

Bit	Field	Type	Reset	Description
15	OC4CE	rw	0x00	Output compare 4 clear enable
14:12	OC4M	rw	0x00	Output compare 4 mode
11	OC4PE	rw	0x00	Output compare 4 preload enable
10	OC4FE	rw	0x00	Output compare 4 fast enable
9:8	CC4S	rw	0x00	Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC4 channel is configured as output 01: CC4 channel is configured as input, IC4 is mapped on TI4 10: CC4 channel is configured as input, IC4 is mapped on TI3 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register). Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).
7	OC3CE	rw	0x00	Output compare 3 clear enable
6:4	OC3M	rw	0x00	Output compare 3 mode
3	OC3PE	rw	0x00	Output compare 3 preload enable
2	OC3FE	rw	0x00	Output compare 3 fast enable
1:0	CC3S	rw	0x00	Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI3 10: CC3 channel is configured as input, IC3 is mapped on TI4 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register). Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

## Input Capture mode

Bit	Field	Type	Reset	Description
15:12	IC4F	rw	0x00	Input capture 4 filter
11:10	IC4PSC	rw	0x00	Input capture 4 prescaler

### 16.4.8 Capture/compare mode register 2 (TIMx\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE		OC4M	OC4PE	OC4FE	CC4S	OC3CE		OC3M	OC3PE	OC3FE	CC3S				
	IC4F		IC4PSC			IC3F		IC3PSC							

9:8	CC4S	rw	0x00	Capture/Compare 4 selection  This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC4 channel is configured as output 01: CC4 channel is configured as input, IC4 is mapped on TI4 10: CC4 channel is configured as input, IC4 is mapped on TI3 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register).  Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).
7:4	IC3F	rw	0x00	Input capture 3 filter
3:2	IC3PSC	rw	0x00	Input capture 3 prescaler

1:0	CC3S	rw	0x00	Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI3 10: CC3 channel is configured as input, IC3 is mapped on TI4 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register). Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).
-----	------	----	------	--

#### 16.4.9 Capture/compare enable register (TIMx\_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4 NP	Res.	CC4P	CC4E	CC3 NP	Res.	CC3P	CC3E	CC2 NP	Res.	CC2P	CC2E	CC1 NP	Res.	CC1P	CC1 E

Bit	Field	Type	Reset	Description
15	CC4NP	rw	0x00	Capture/Compare 4 complementary output polarity Refer to CC1NP description.
14	Reserved			Reserved and always read as 0.
13	CC4P	rw	0x00	Capture/Compare 4 output polarity Refer to CC1P description.
12	CC4E	rw	0x00	Capture/Compare 4 output enable Refer to CC1E description.
11	CC3NP	rw	0x00	Capture/Compare 3 complementary output polarity Refer to CC1NP description.
10	Reserved			Reserved and always read as 0.
9	CC3P	rw	0x00	Capture/Compare 3 output polarity Refer to CC1P description.
8	CC3E	rw	0x00	Capture/Compare 3 output enable Refer to CC1E description.
7	CC2NP	rw	0x00	Capture/Compare 2 complementary output polarity Refer to CC1NP description.

6	Reserved			Reserved and always read as 0.
5	CC2P	rw	0x00	Capture/Compare 2 output polarity Refer to CC1P description.
4	CC2E	rw	0x00	Capture/Compare 2 output enable Refer to CC1E description.
3	CC1NP	rw	0x00	Capture/Compare 1 complementary output polarity 0: OC1N active high 1: OC1N active low Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).
2	Reserved			Reserved and always read as 0.

Bit	Field	Type	Reset	Description
1	CC1P	rw	0x00	Capture/Compare 1 output polarity CC1 channel configured as output: 0: OC1 active high 1: OC1 active low CC1 channel configured as input: This bit selects whether IC1 or the inverted signal of IC1 is used for trigger or capture operations. 0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted. 1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted. Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

0	CC1E	rw	0x00	Capture/Compare 1 output enable CC1 channel configured as output: 0: Off - OC1 is not active. OC1 level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits. 1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits. CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the TIMx_CCR1 register or not. 0: capture disabled 1: capture enabled
---	------	----	------	---

Table 51. Output control bit for standard Ocx channels

CCxE bit	OCx output state
0	Output disabled (OCx = 0, OCx_EN = 0)
1	OCx = OCxREF + Polarity, OCx_EN = 1

Note: The states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and the GPIO and AFIO registers.

#### 16.4.10 Counter (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description											
15:0	CNT	rw	0x0000	Counter value											

#### 16.4.11 Prescaler (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description											
15:0	PSC	rw	0x0000	Prescaler value											

15:0	PSC	rw	0x0000	Prescaler value The counter clock frequency (CK_CNT) is equal to $f_{CK\_Psc}/(PSC + 1)$ . PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through slave controller when configured in "reset mode").
------	-----	----	--------	---

#### 16.4.12 Auto-reload register (TIMx\_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description
15:0	ARR	rw	0x0000	Auto-reload value ARR is the value to be loaded in the actual auto-reload register. Refer to the Section 17.3.1 for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.

#### 16.4.13 Capture/compare register 1 (TIMx\_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description

15:0	CCR1	rw	0x0000	Capture/Compare 1 value If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded immediately if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output. If channel CC1 is configured as input: CCR1 is the counter value transferred by the last input capture 1 event (IC1).
------	------	----	--------	--

#### 16.4.14 Capture/compare register 2 (TIMx\_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:0	CCR2	rw	0x0000	Capture/Compare 2 value If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded immediately if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output. If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (IC2).

#### 16.4.15 Capture/compare register 3 (TIMx\_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:0	CCR3	rw	0x0000	<p>Capture/Compare 3 value</p> <p>If channel CC3 is configured as output:</p> <p>CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).</p> <p>It is loaded immediately if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output. If channel CC3 is configured as input:</p> <p>CCR3 is the counter value transferred by the last input capture 3 event (IC3).</p>

#### 16.4.16 Capture/compare register 4 (TIMx\_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:0	CCR4	rw	0x0000	<p>Capture/Compare 4 value</p> <p>If channel CC4 is configured as output:</p> <p>CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).</p> <p>It is loaded immediately if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC4 output.</p> <p>If channel CC4 is configured as input:</p> <p>CCR4 is the counter value transferred by the last input capture 4 event (IC4).</p>

#### 16.4.17 DMA control register (TIMx\_DCR)

Address offset: 0x48

Reset value: 0x0000

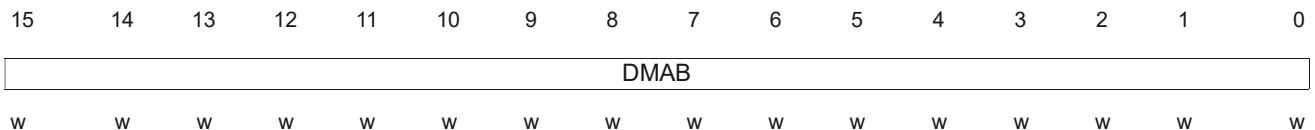
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		DBL				Res.		DBA							

Bit	Field	Type	Reset	Description
15:13	Reserved			Reserved, always read as 0
12:8	DBL	w	0x00	<p>DMA burst length</p> <p>This bit-field defines the length of DMA burst transfer (the timer recognizes a burst transfer when a write access is done to the TIMx_DMAR register), ie. the number of transfers by half-words (double bytes) or bytes:</p> <ul style="list-style-type: none"> <li>00000: 1 transfer</li> <li>00001: 2 transfers</li> <li>00010: 3 transfers...</li> <li>...</li> <li>10001: 18 transfers</li> </ul> <p>Example: Let us consider the following transfer: DBL = 7, DBA = TIM2_CR1</p> <ul style="list-style-type: none"> <li>- In this case, DBL = 7 and DBA = TIM2_CR1 represent the address to which the data is to be sent. Then the transfer address can be concluded from the following formula:  <math>(\text{TIMx\_CR1 address}) + \text{DBA} + (\text{DMA index})</math>, where DMA index = DBL,</li> <li>(TIMx_CR1 address) + DBA + 7 provides the address to/from which the transfer is done to/from 7 registers starting from the address (TIMx_CR1 address) + DBA. The following situations may occur according to the configuration of DMA data length: <ul style="list-style-type: none"> <li>- If the data is configured in half-words (16 bits), then the data is transferred to all of seven registers.</li> <li>- If the data is configured in bytes, the data is still transferred to all of seven registers: the first register contains the first MSB byte, the second contains the first LSB byte, and so on. Therefore, users must specify the data length in the DMA transfer as for timers.</li> </ul> </li> </ul>
7:5	Reserved			Reserved and always read as 0.
4:0	DBA	w	0x00	<p>DMA base address</p> <p>This bit-field defines the base-address for DMA burst transfer (when a write access is done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.</p> <ul style="list-style-type: none"> <li>00000: TIMx_CR1</li> <li>00001: TIMx_CR2</li> <li>00010: TIMx_SMCR</li> <li>.....</li> </ul>

### 16.4.18 DMA address for full transfer (TIMx\_DMAR)

Address offset: 0x4C

Reset value: 0x0000



Bit	Field	Type	Reset	Description
15:0	DMAB	w	0x0000	<p>DMA register for burst accesses</p> <p>A write operation to the TIMx_DMAR register accesses the register located at the address:</p> <p>'TIMx_CR1 address' + DBA + DMA index, where 'TIMx_CR1 address' is the address of the control register 1 (TIMx_CR1),</p> <p>'DBA' is the base address configured in TIMx_DCR register,</p> <p>'DMA index' is the offset automatically controlled by DMA, and depends on the DBL configured in TIMx_DCR.</p>

# 17 | 32-Bit General-Purpose Timer (TIM2)

## 32-Bit General-Purpose Timer (TIM2)

### 17.1 TIMx introduction

The general-purpose timers consist of a 32-bit auto-reload counter driven by a programmable prescaler. They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM). Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIMx timers are completely independent, and do not share any resources. They can be synchronized together.

### 17.2 TIMx main features

General-purpose TIMx(TIM2) timer features include:

- 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or internally/externally triggered counting)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

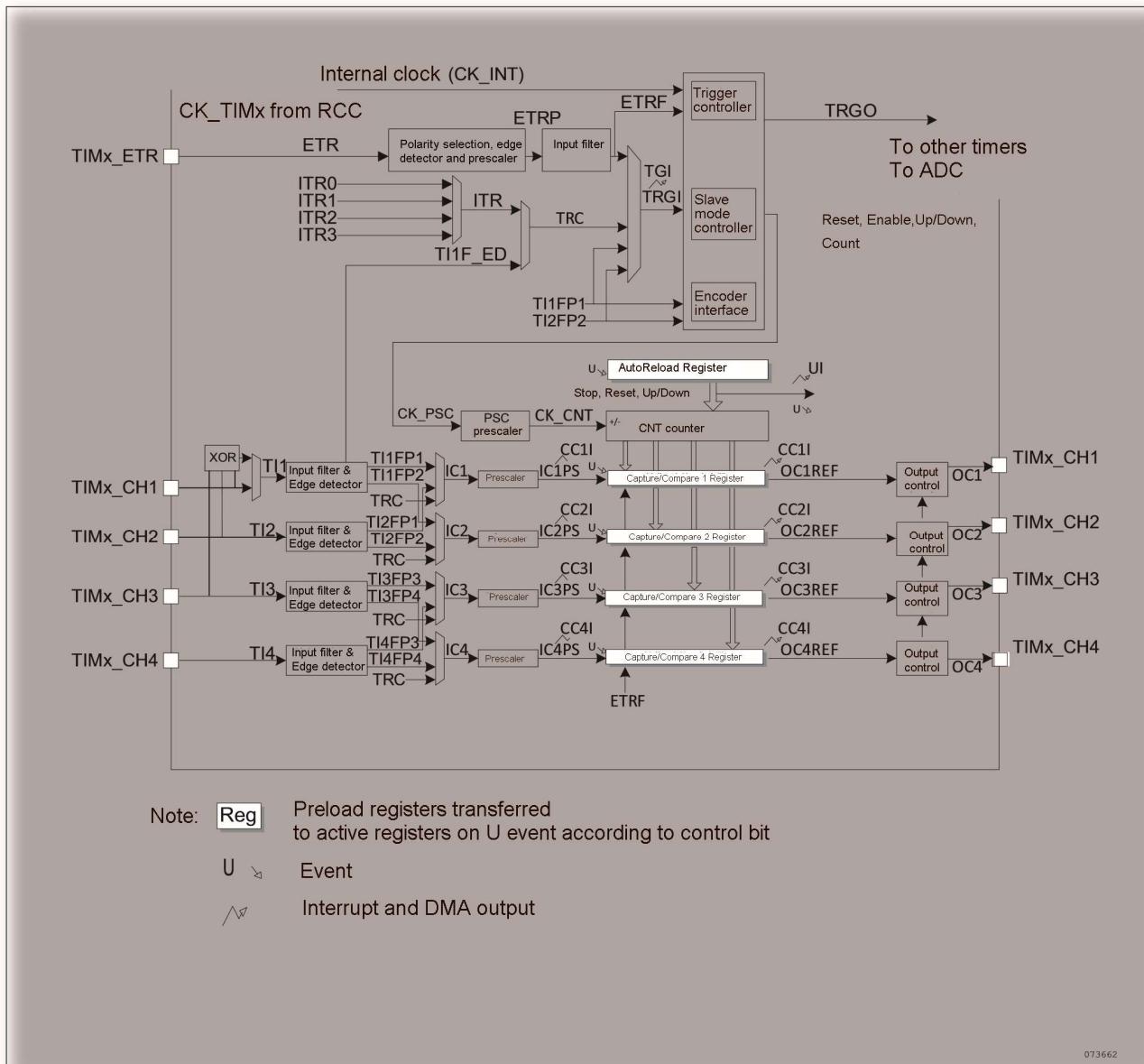


Figure 136. General-purpose timer block diagram

## 17.3. TIMx functional description

### 17.3.1 Time-base unit

The main block of the programmable general-purpose timer is a 32-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler. The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running. The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register immediately or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in the TIMx\_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

## Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 32-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

The figures below give some examples of the counter behavior when the prescaler ratio is changed on the fly:

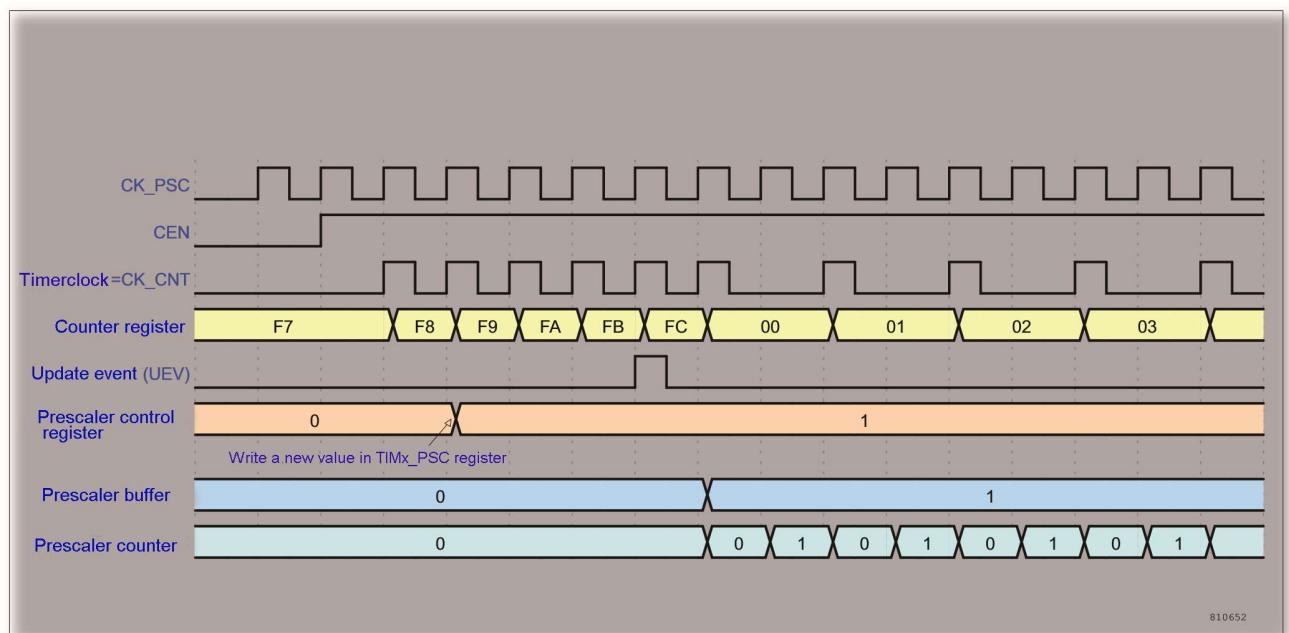


Figure 137. Counter timing diagram with prescaler division change from 1 to 2

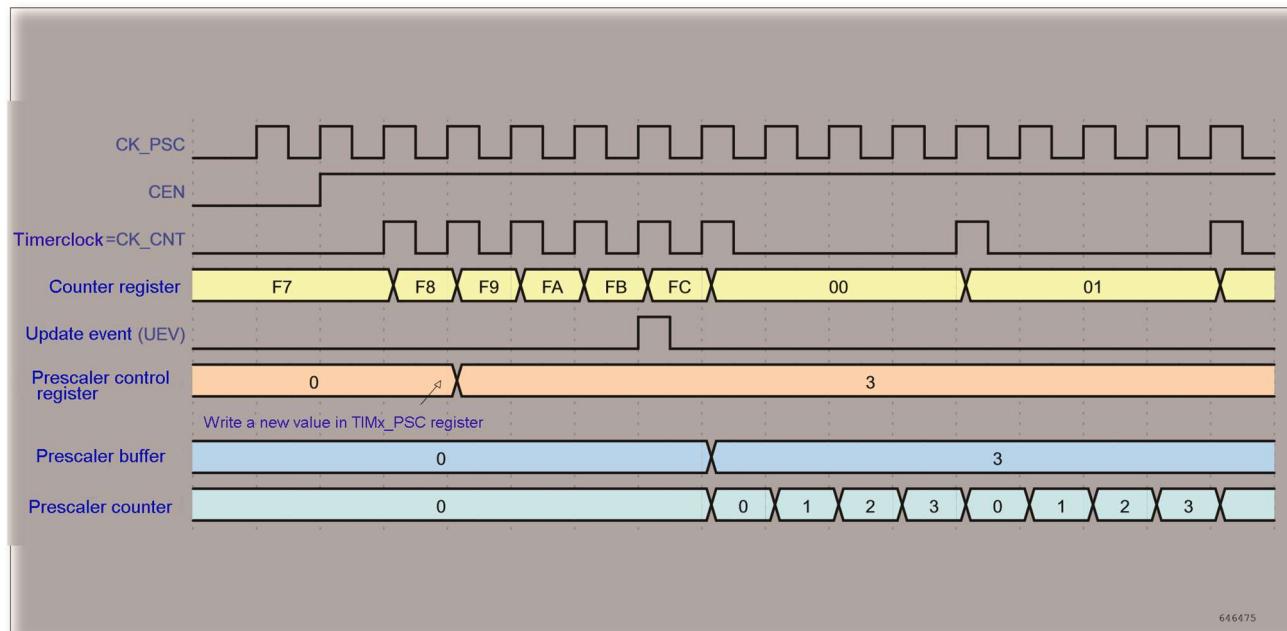


Figure 138. Counter timing diagram with prescaler division change from 1 to 4

### 17.3.2 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR counter), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change) when an update event should be generated. In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag by hardware (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture mode.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set by hardware (depending on the URS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR = 0x36:

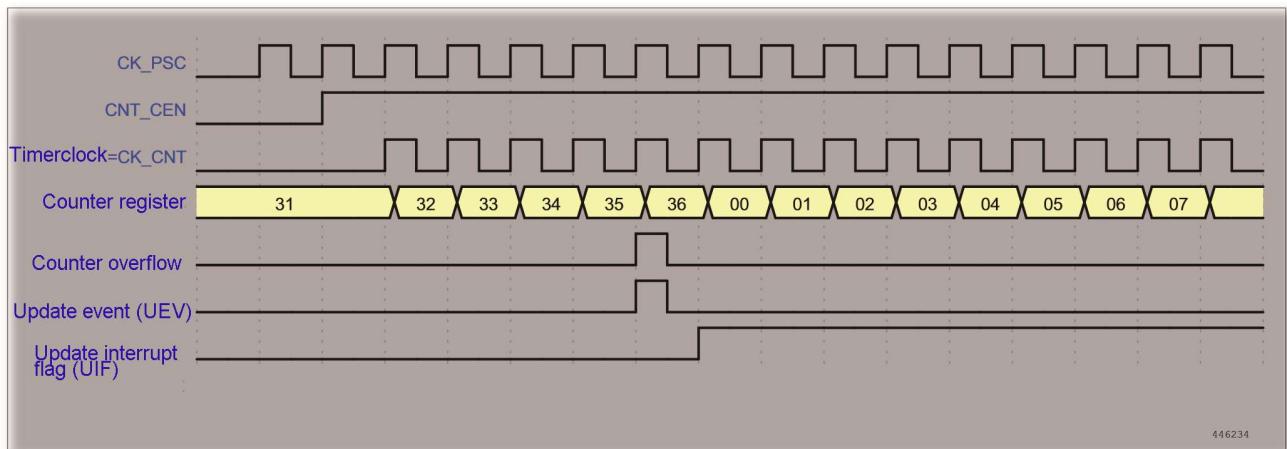


Figure 139. Counter timing diagram, internal clock divided by 1

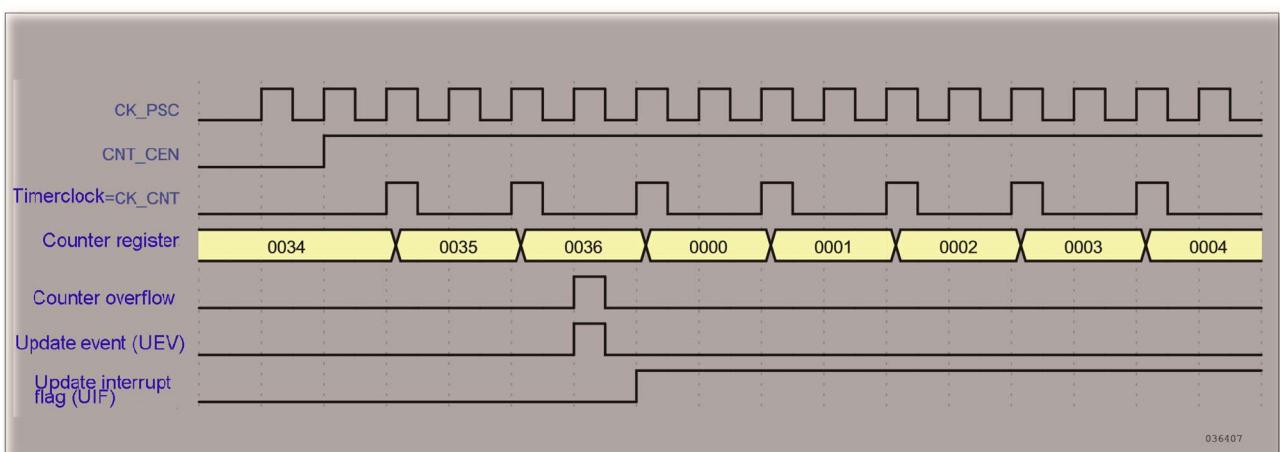


Figure 140. Counter timing diagram, internal clock divided by 2

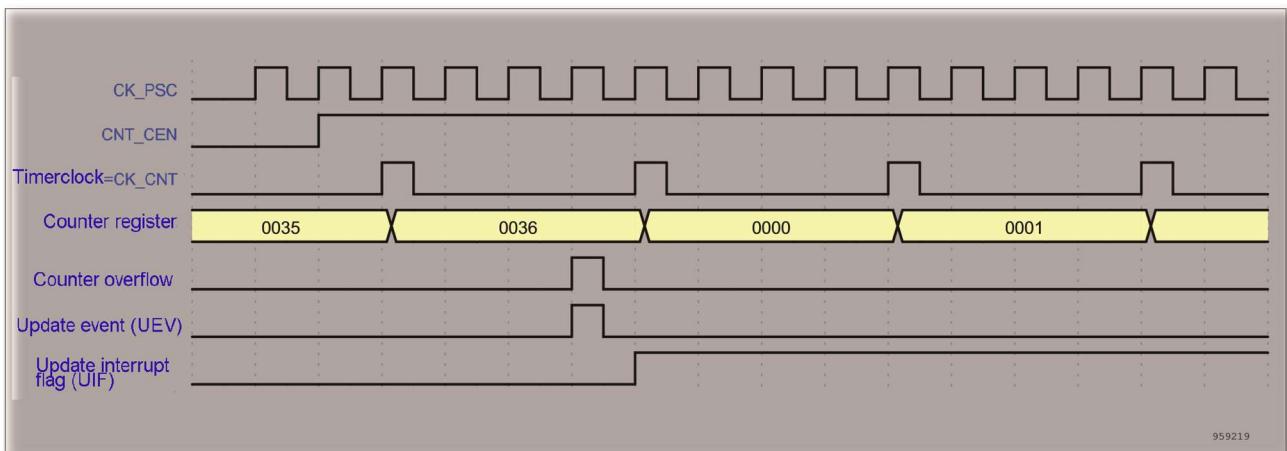


Figure 141. Counter timing diagram, internal clock divided by 4

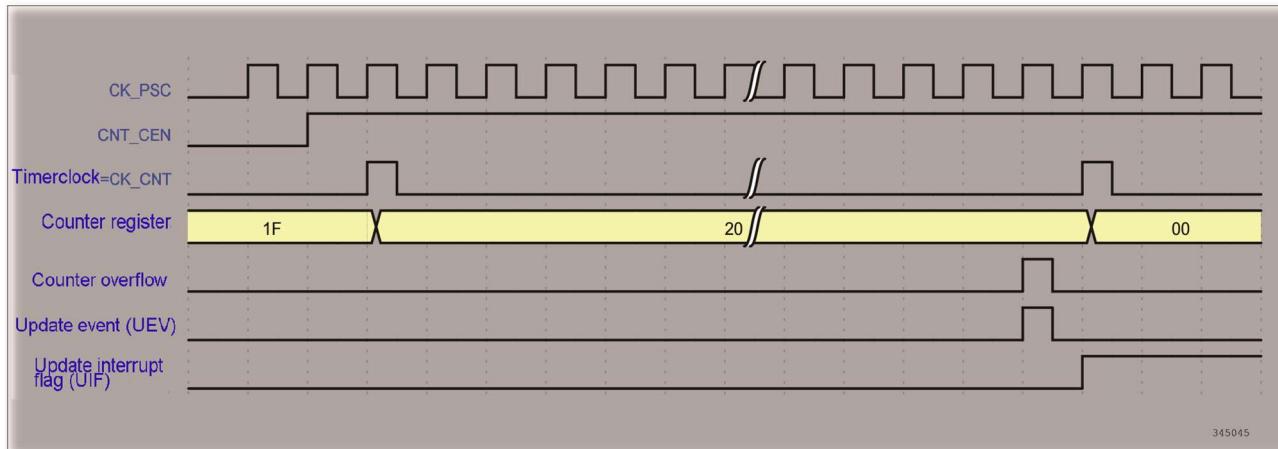


Figure 142. Counter timing diagram, internal clock divided by N

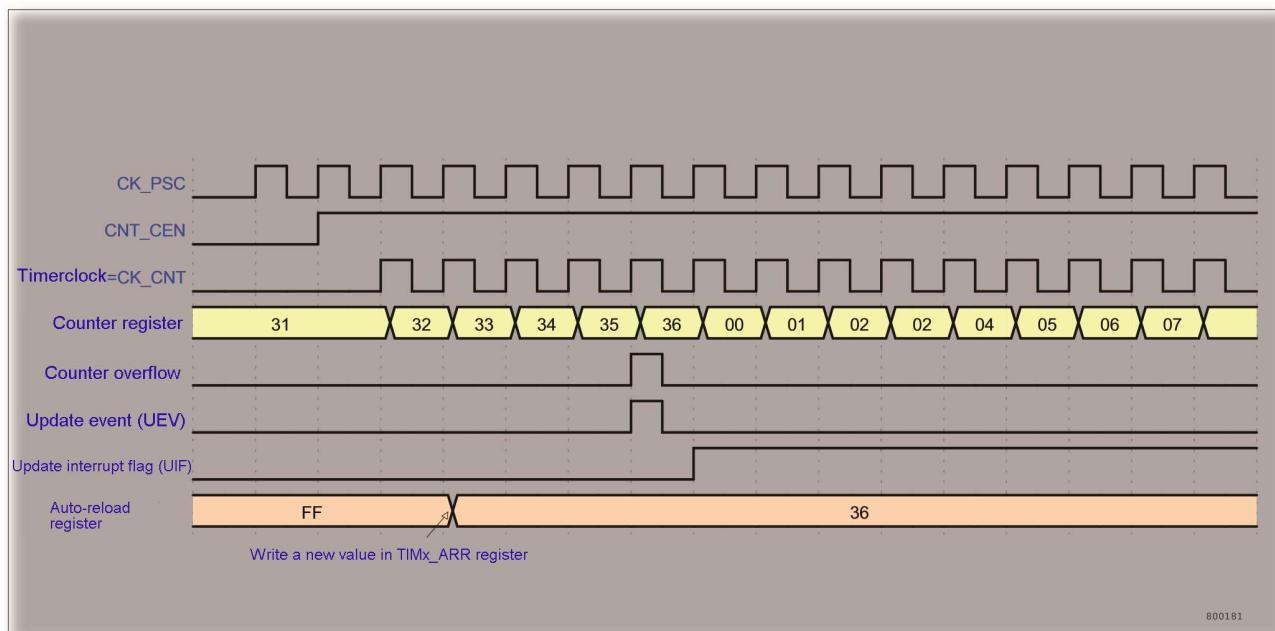


Figure 143. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)

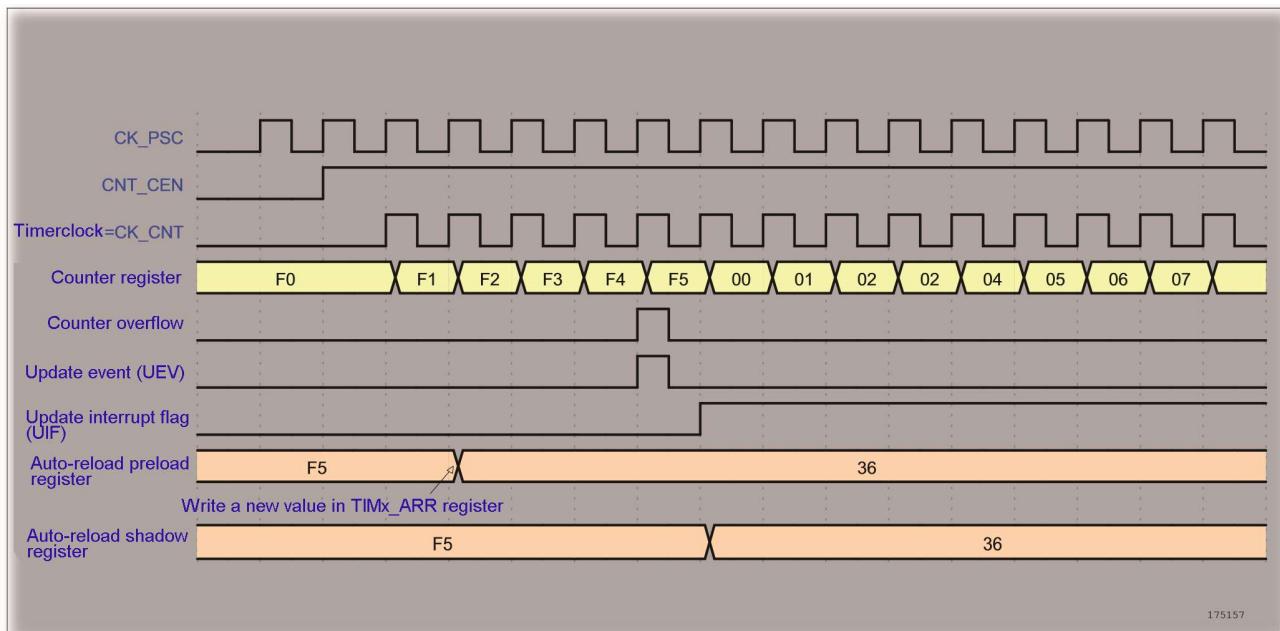


Figure 144. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)

### Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx\_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller).

The UEV update event can be disabled by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register).

Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR = 0x36.

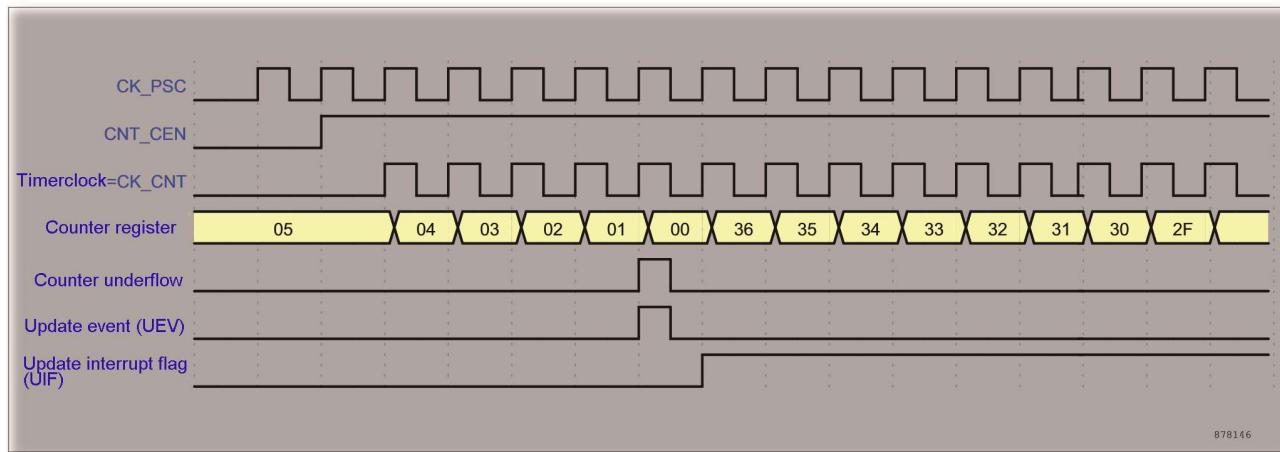


Figure 145. Counter timing diagram, internal clock divided by 1

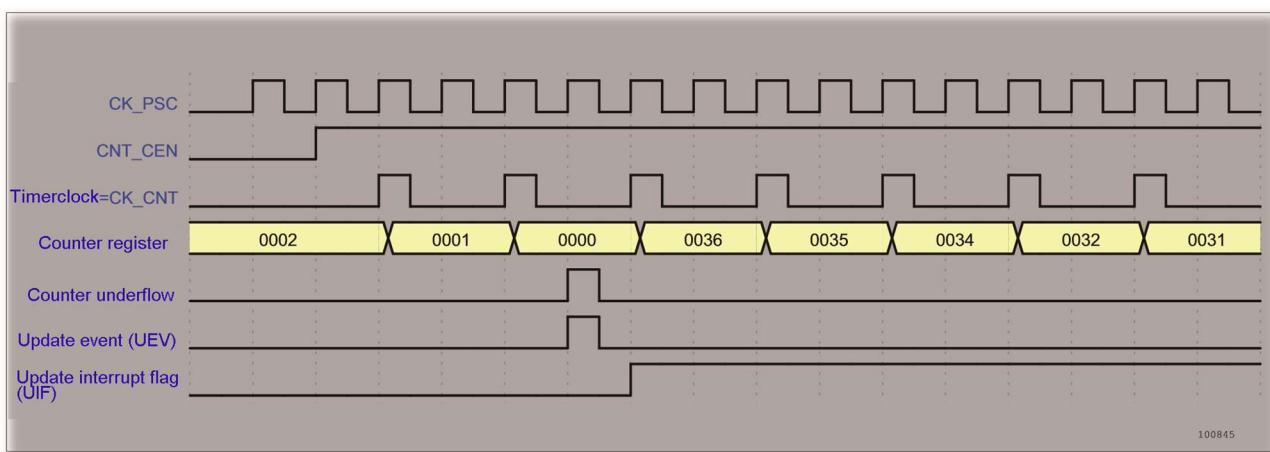


Figure 146. Counter timing diagram, internal clock divided by 2

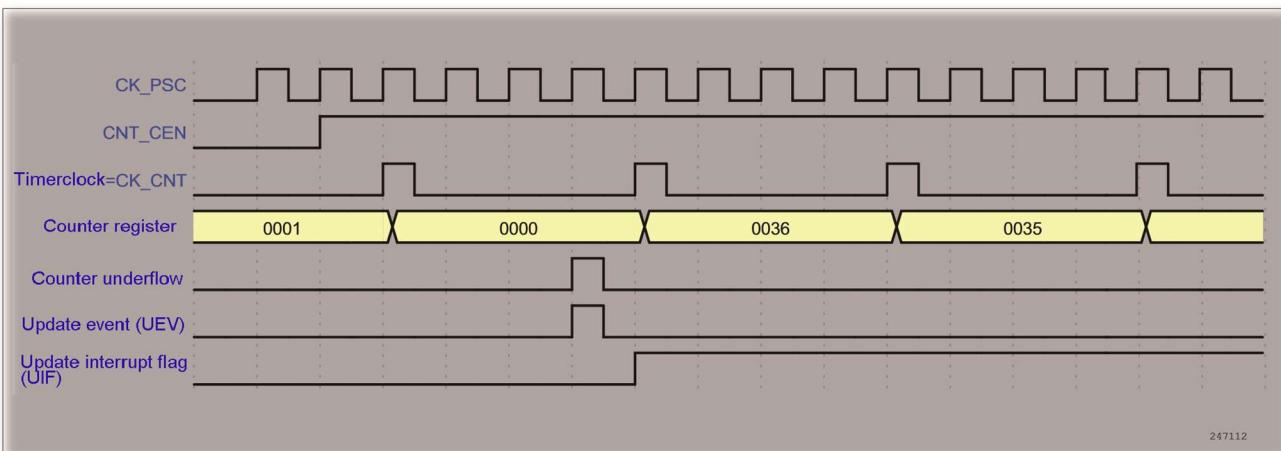


Figure 147. Counter timing diagram, internal clock divided by 4

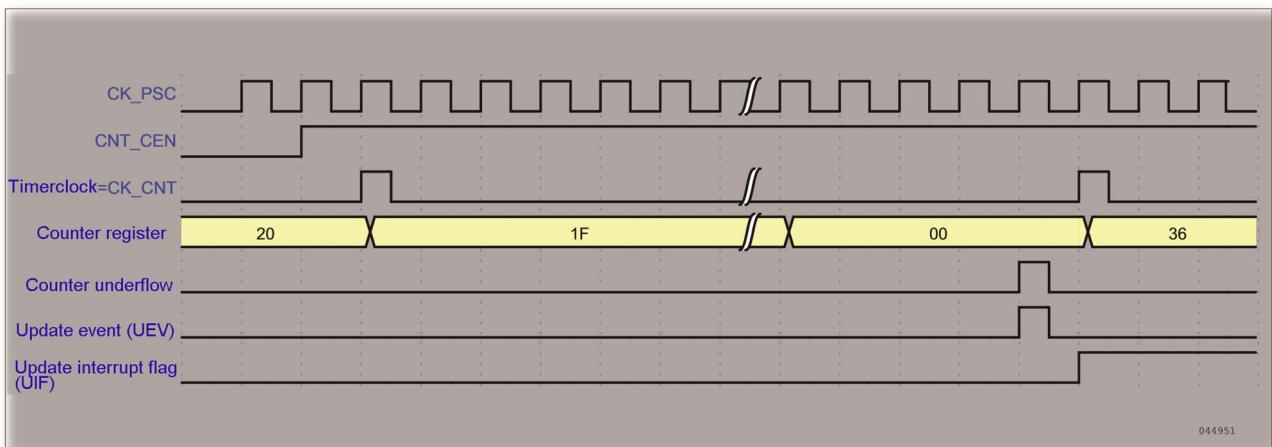


Figure 148. Counter timing diagram, internal clock divided by N

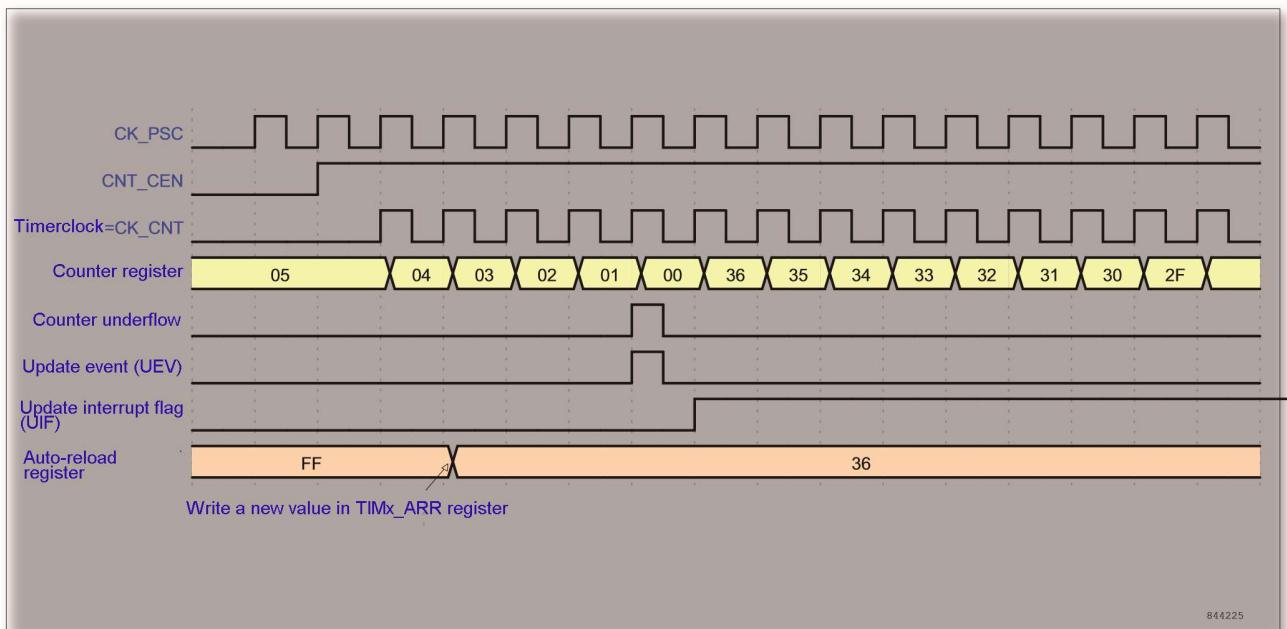


Figure 149. Counter timing diagram, update event when repetition counter is not used

### Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

In this mode, the DIR direction bit in the TIMx\_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter. The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller). In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register).

Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies:

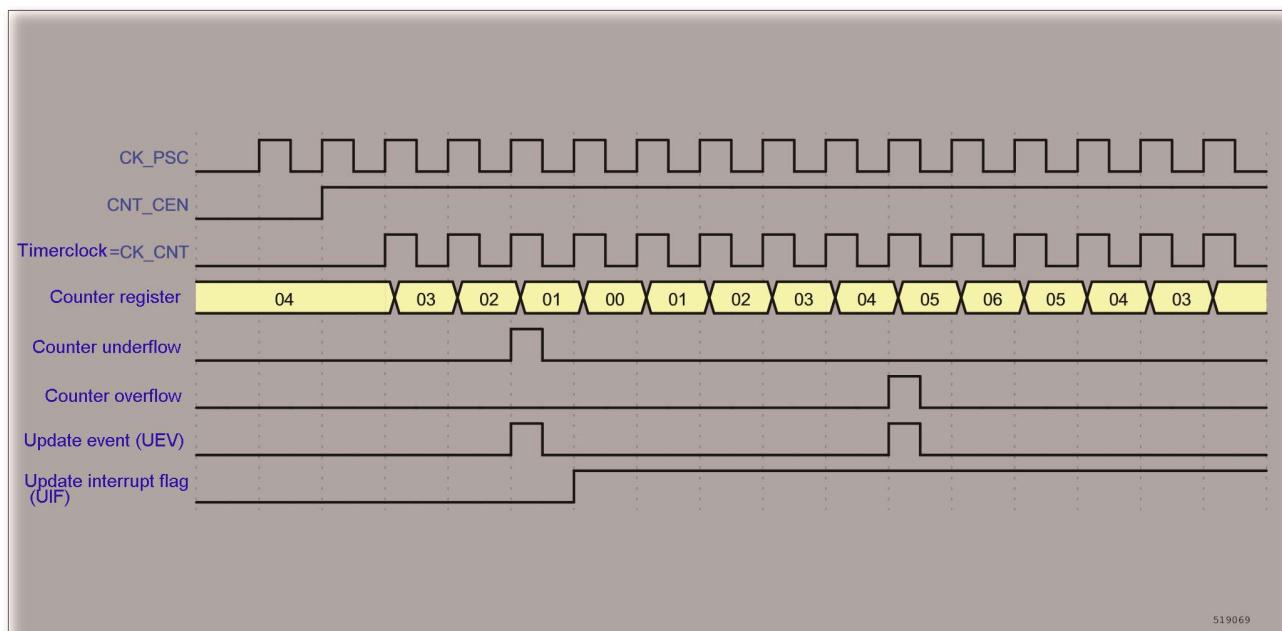


Figure 150. Counter timing diagram, internal clock divided by 1, TIMx\_ARR = 0x6

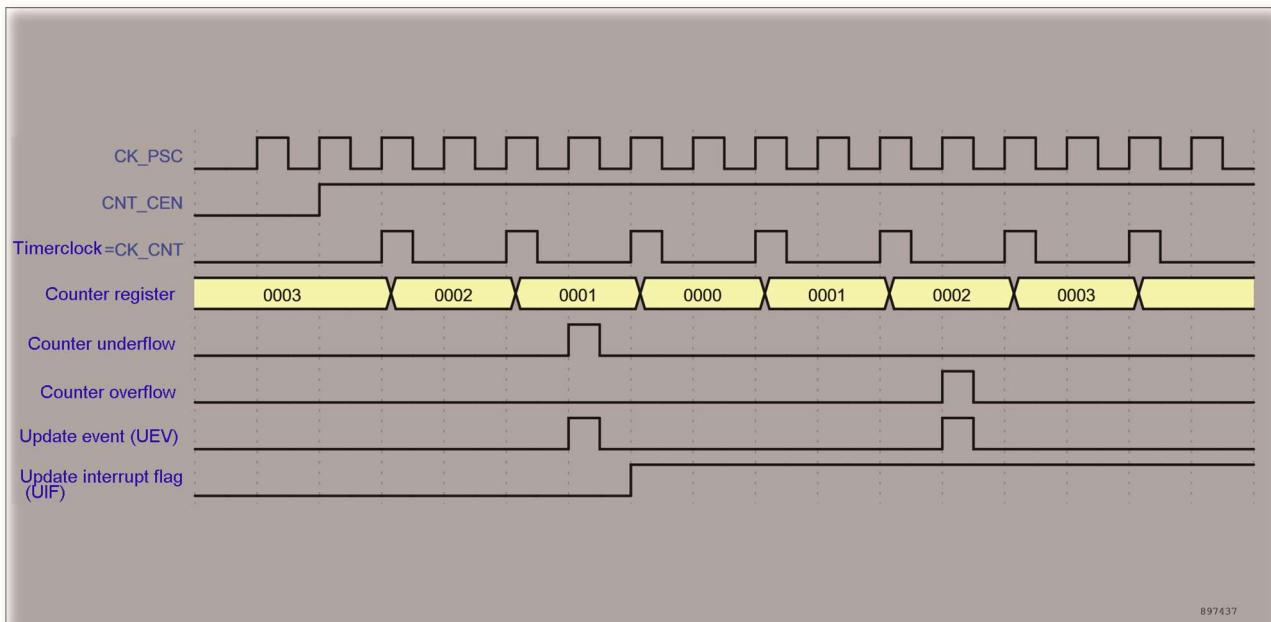


Figure 151. Counter timing diagram, internal clock divided by 2

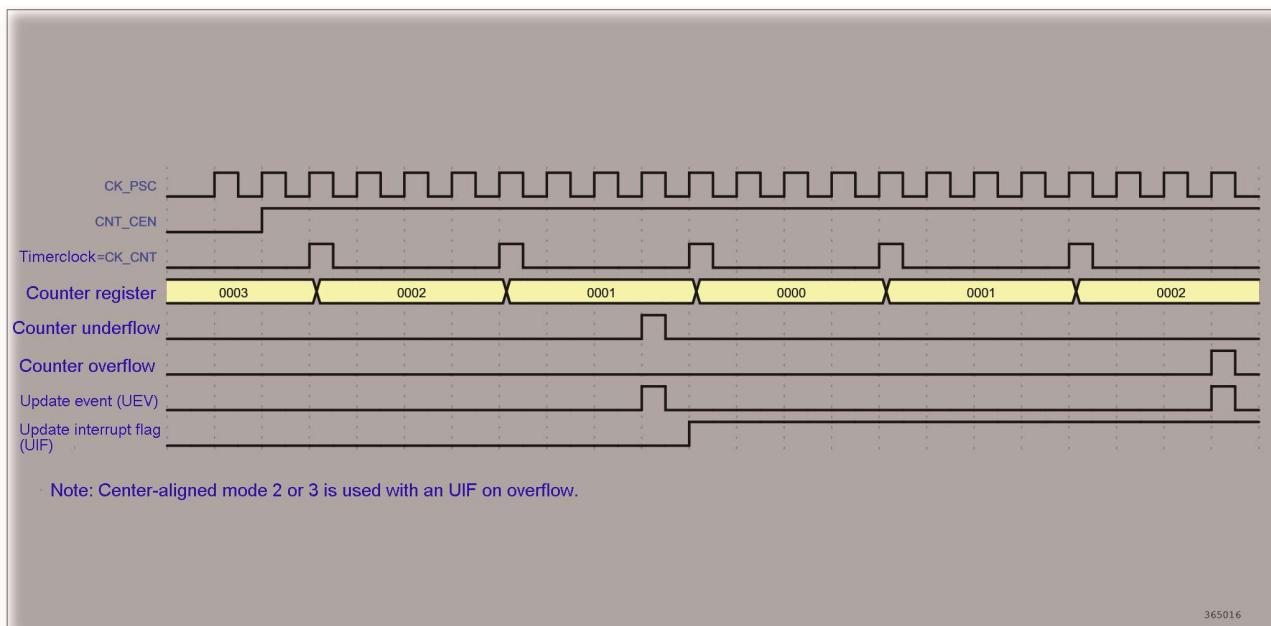


Figure 152. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x36

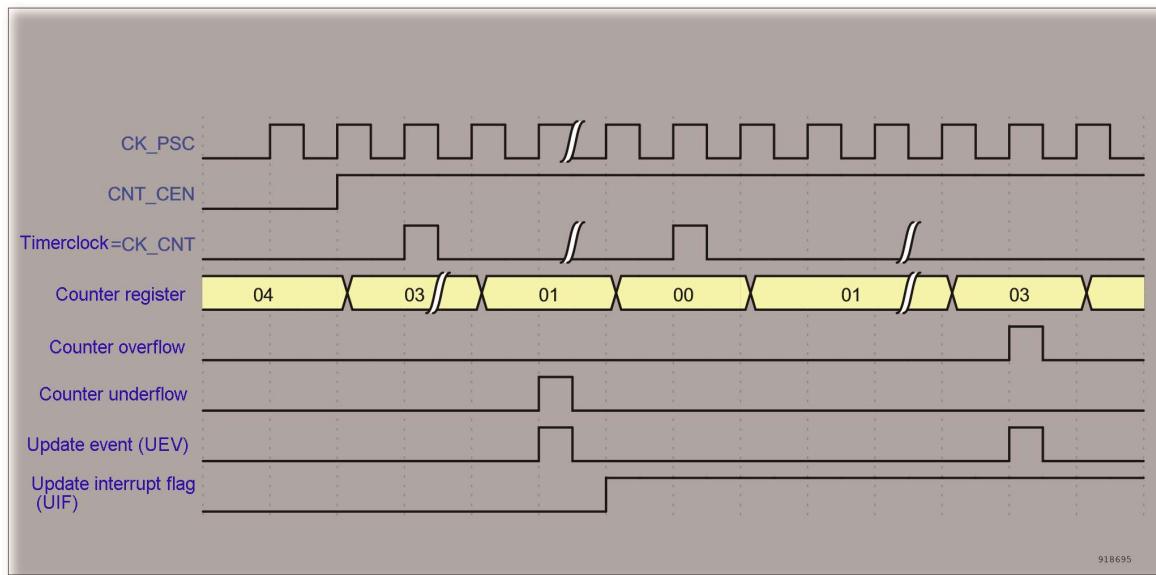


Figure 153. Counter timing diagram, internal clock divided by N

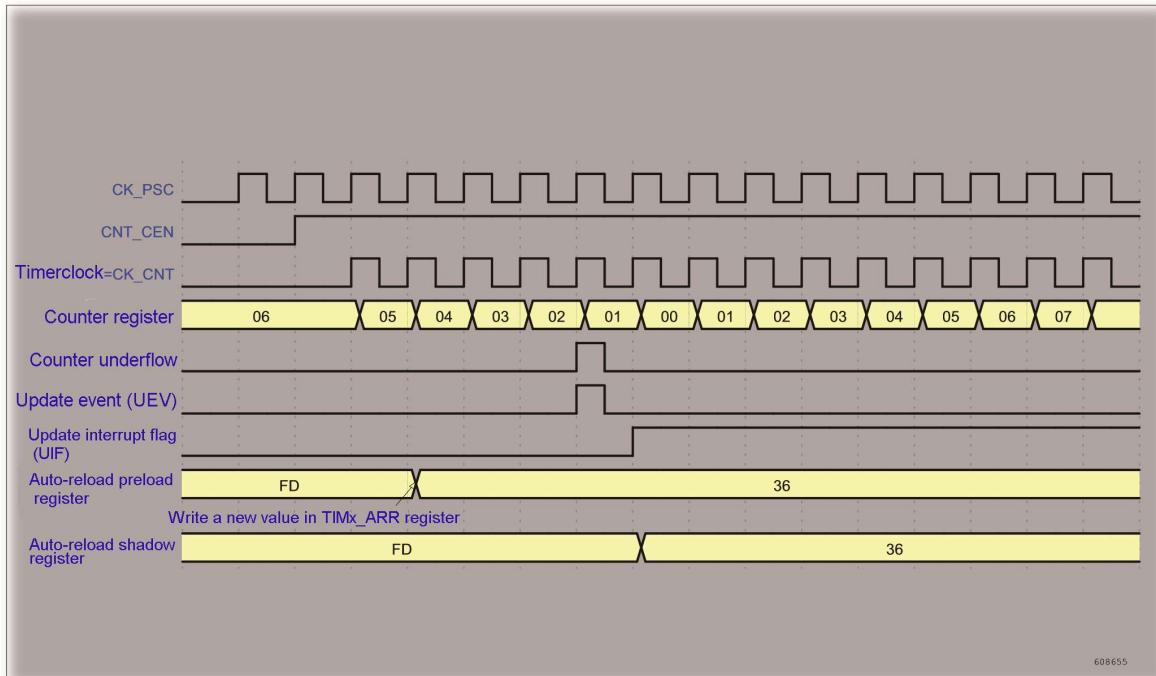


Figure 154. Counter timing diagram, update event with ARPE=1 (counter underflow)

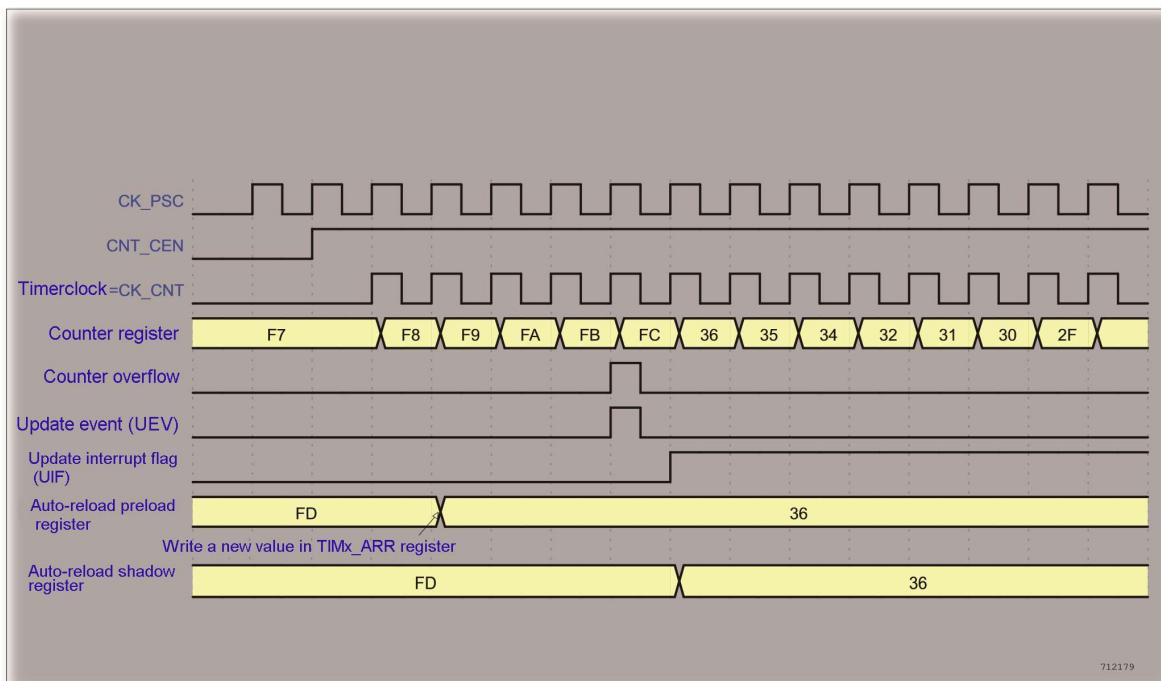


Figure 155. Counter timing diagram, Update event with ARPE=1 (counter overflow)

### 17.3.3 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK\_INT).
- External clock mode 1: external input pin (TIx).
- External clock mode 2: external trigger input (ETR).
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, the user can configure Timer 1 to act as a prescaler for Timer 2.

#### Internal clock source (CK\_INT)

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

The figure below shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

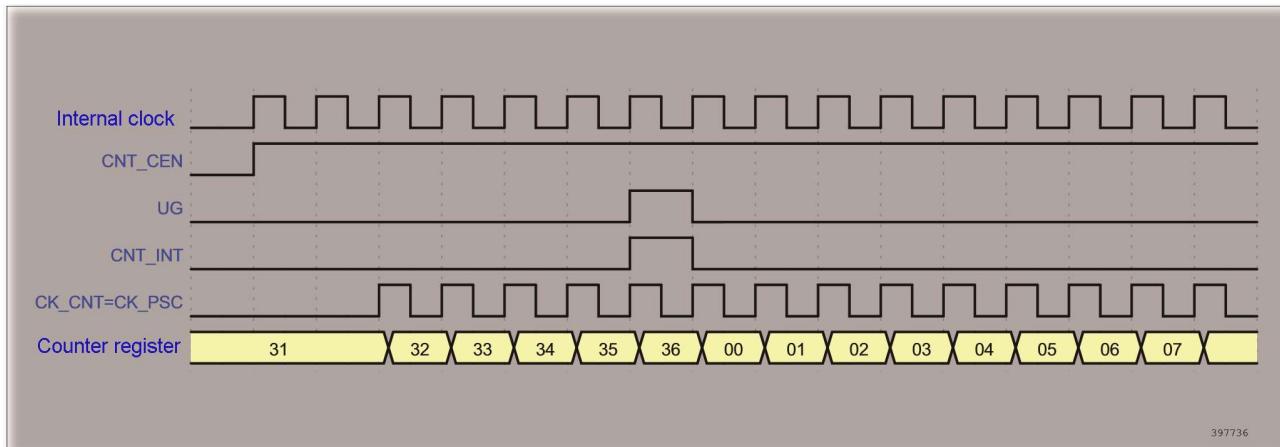


Figure 156. Control circuit in normal mode, internal clock divided by 1

### External clock source mode 1

This mode is selected when SMS = 111 in the TIMx\_SMCR register. The counter can count at each rising or falling edge on a selected input.

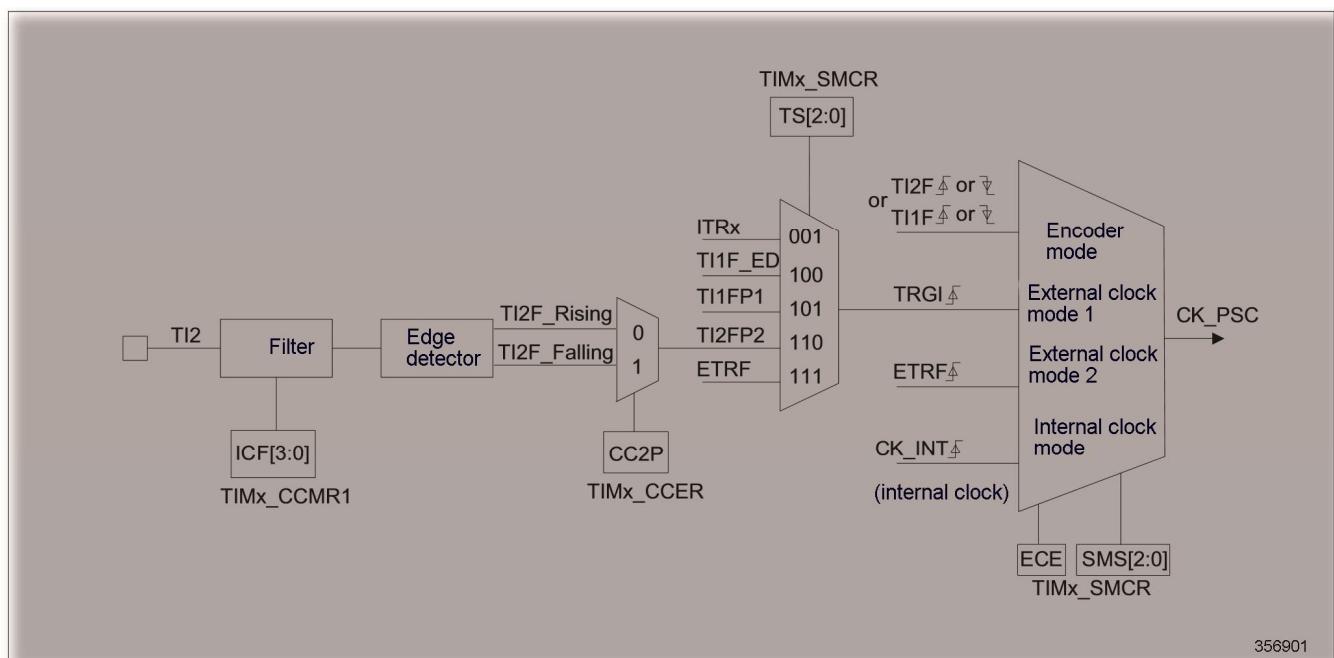


Figure 157. TI2 external clock connection example

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = 01 in the TIMx\_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F = 0000).
- Note: The capture prescaler is not used for triggering, so the user does not need to configure it.
3. Select rising edge polarity by writing CC2P = 0 in the TIMx\_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx\_SMCR register.
5. Select TI2 as the trigger input source by writing TS = 110 in the TIMx\_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx\_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

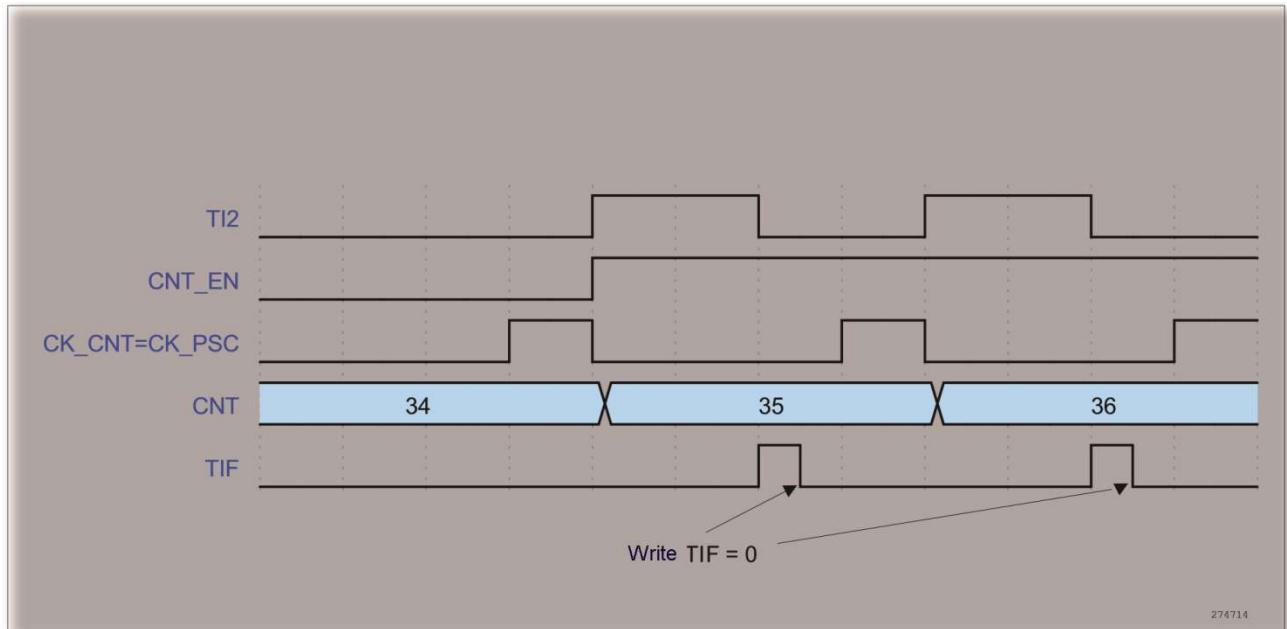


Figure 158. Control circuit in external clock mode 1

### External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx\_SMCR register. The counter can count at each rising or falling edge on the external trigger input ETR. The figure below gives an overview of the external trigger input block.

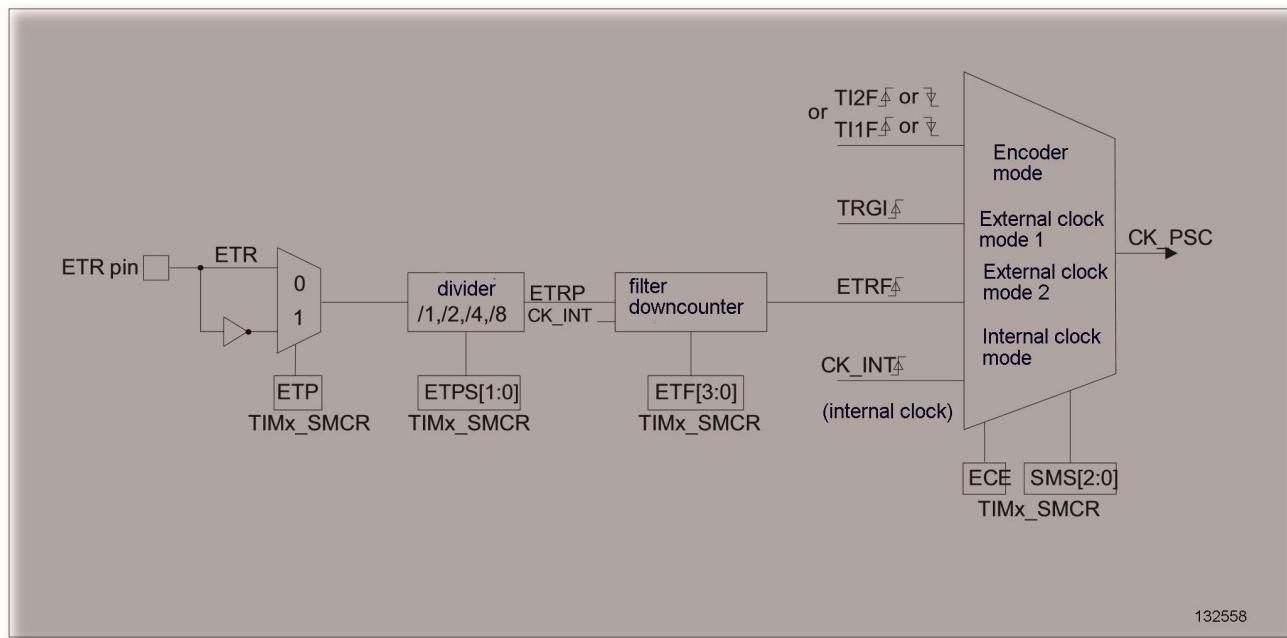


Figure 159. External trigger input block

For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx\_SMCR register.
2. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx\_SMCR register.

3. Select rising edge detection on the ETR pin by writing ETP = 0 in the TIMx\_SMCR register.
4. Enable external clock mode 2 by writing ECE = 1 in the TIMx\_SMCR register.
5. Enable the counter by writing CEN = 1 in the TIMx\_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

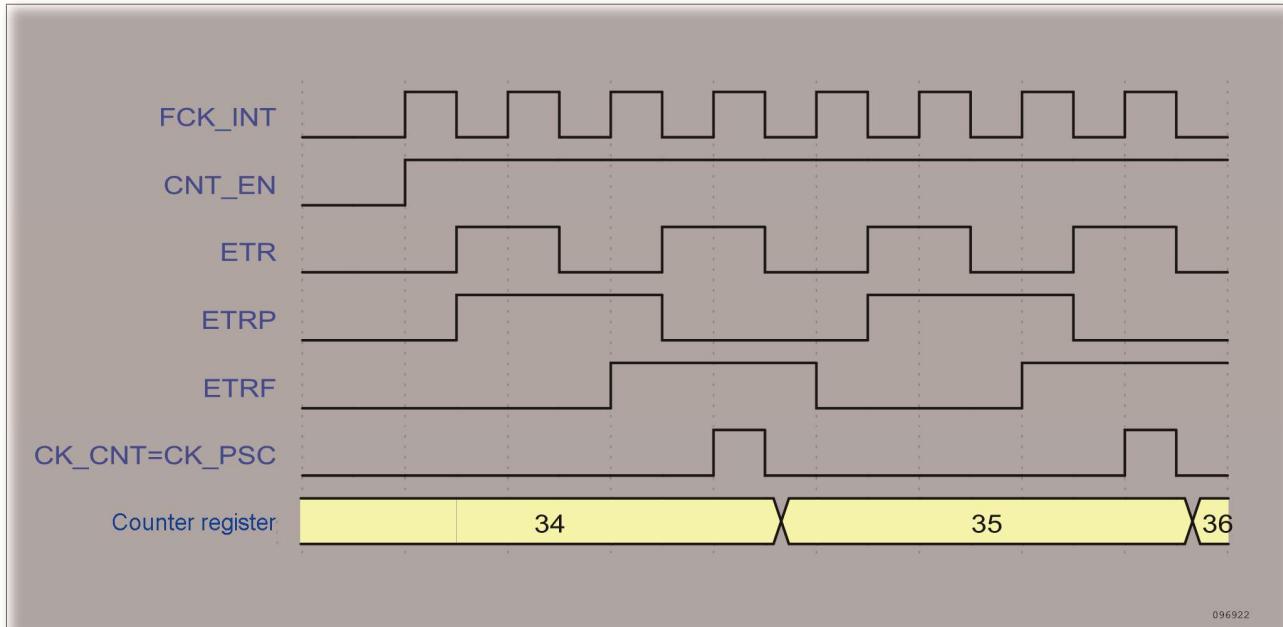


Figure 160. Control circuit in external clock mode 2

#### 17.3.4 Capture/compare channel

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figures give an overview of one Capture/Compare channel. The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

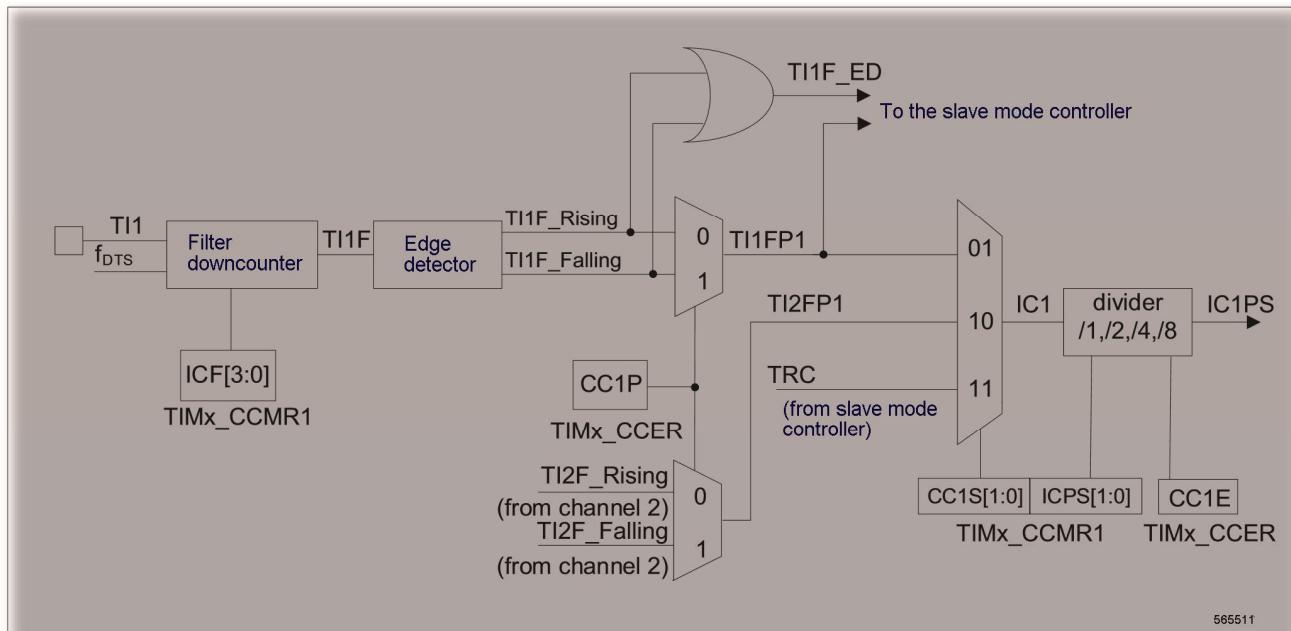


Figure 161. Capture/compare channel (example: channel 1 input stage)

The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

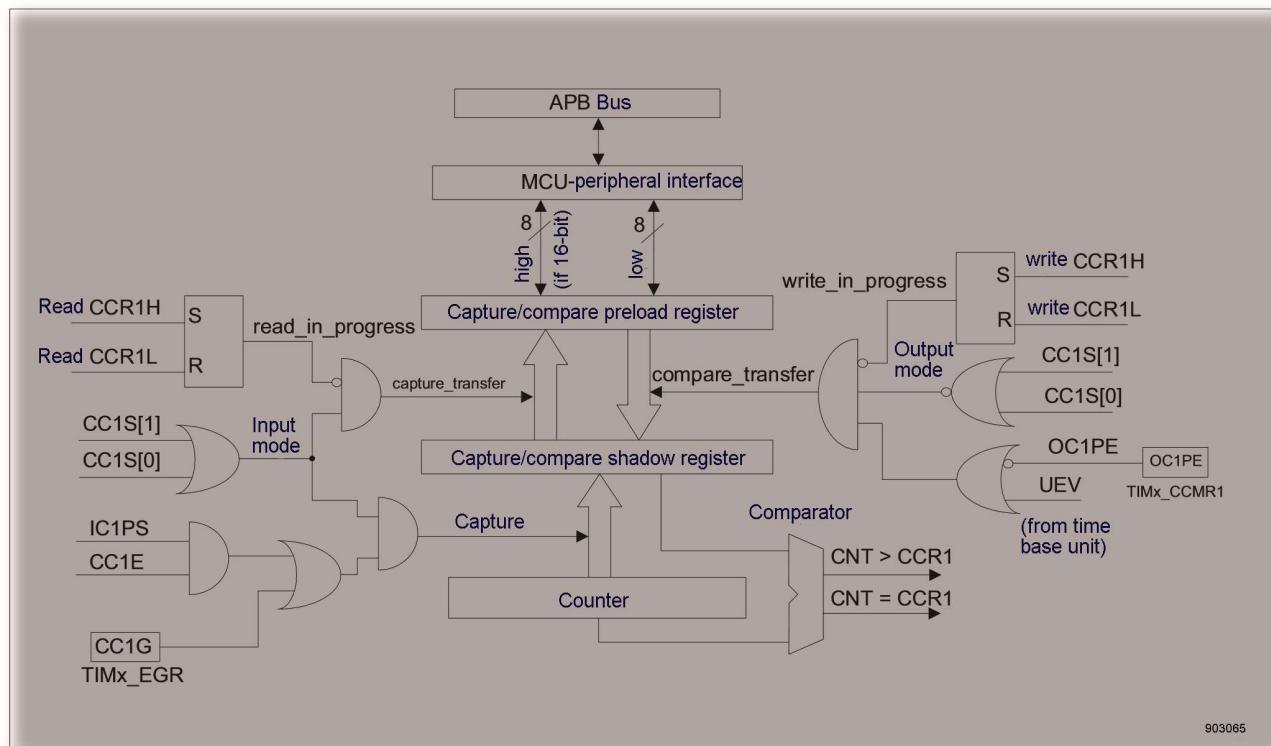


Figure 162. Capture/compare channel 1 main circuit

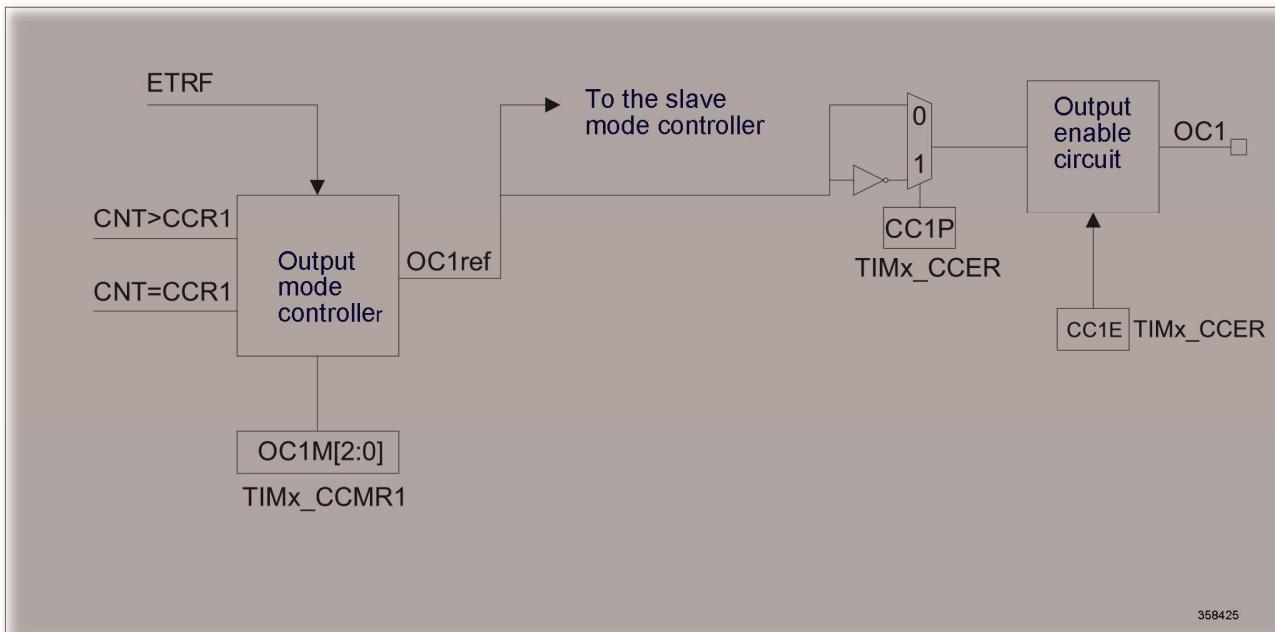


Figure 163. Output stage of capture/compare channel (channel 1)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register. In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 17.3.5 Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx\_CCRx) are used to latch the value of the counter after an edge is detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx\_SR register) is set.

CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when written to '0'.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TM1\_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the input signal (by programming ICxF bits in the TIMx\_CCMRx register if the input is a TIx input). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate an edge transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at  $f_{DTs}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx\_CCER register (rising edge in this case).

- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit to '1' in the TIMx\_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active level transition.
- CC1IF flag is set (interrupt flag).
- CC1OF is also set to '1' if at least two consecutive captures occurred whereas the CC1IF flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.

### 17.3.6 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input
- These 2 ICx signals are active on edges with opposite polarity
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode

For example, the user can measure the period (in TIMx\_CCR1 register) and the duty cycle (in TIMx\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):

- Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P bit to '0' (active on rising edge).
- Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P bit to '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx\_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx\_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

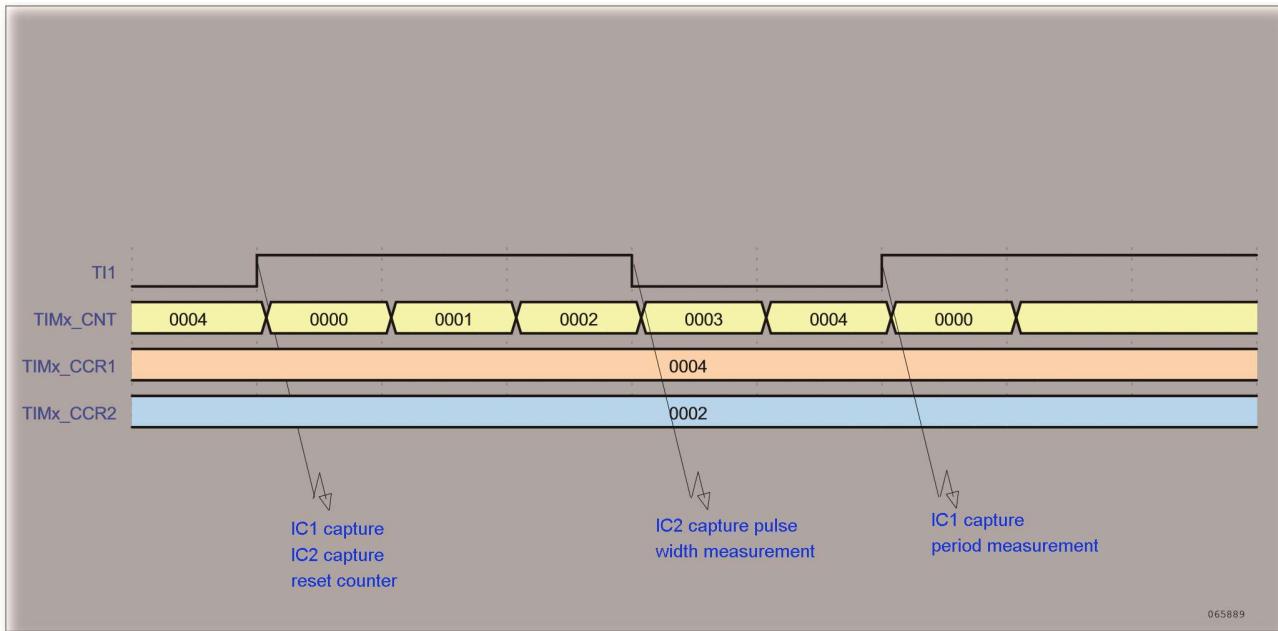


Figure 164. Output stage of capture/compare channel (channel 1)

The PWM input mode can be used only with the TIMx\_CH1/TIMx\_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

### 17.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCxREF is forced high (OCxREF is always active high) and OCx gets opposite value to CCxP polarity bit. For example: CCxP = 0 (OCx active high) => OCx is forced to high level. The OCxREF signal can be forced low by writing the OCxM bit to 100 in the TIMx\_CCMRx register. Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 17.3.8 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bit in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM = 000), be set active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx\_DIER register).
- Generates a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register. In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The synchronization resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

The output compare mode is configured as follows:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, the user must write OCxM = 011, OCxPE = 0, CCxP = 0 and CCxE = 1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in the following figure.

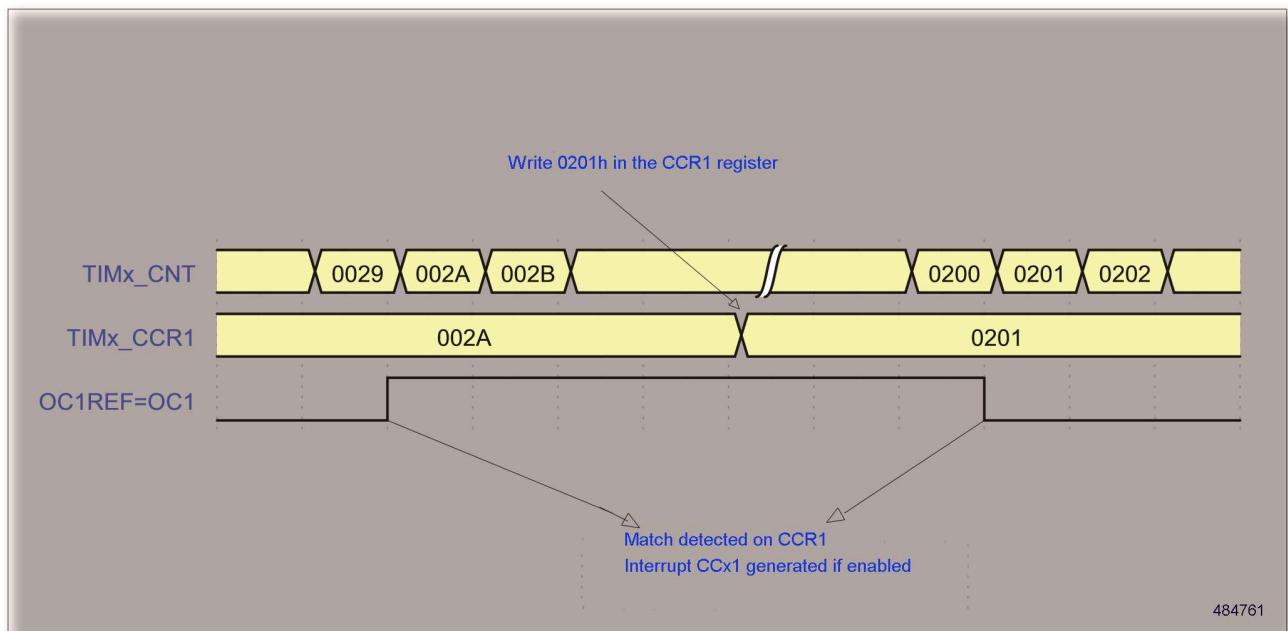


Figure 165. Output compare mode, toggle on OC1

### 17.3.9 PWM mode

Pulse Width Modulation mode allows generating a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bit in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user must initialize all the registers by setting the UG bit in the TIMx\_EGR register.

OC<sub>x</sub> polarity is software programmable using the CC<sub>xP</sub> bit in the TIM<sub>x</sub>\_CCER register. It can be programmed as active high or active low. OC<sub>x</sub> output is enabled by using the CC<sub>xE</sub> bit in the TIM<sub>x</sub>\_CCER register. For details, refer to the TIM<sub>x</sub>\_CCERx register description.

In PWM mode (1 or 2), TIM<sub>x</sub>\_CNT and TIM1\_CCR<sub>x</sub> are always compared to determine whether TIM1\_CCR<sub>x</sub> ≤ TIM1\_CNT or TIM1\_CNT ≤ TIM1\_CCR<sub>x</sub> (depending on the direction of the counter). However, to comply with the OCREF\_CLR functionality (OC<sub>x</sub>REF can be cleared by an external event through the ETR signal until the next PWM period), the OC<sub>x</sub>REF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (OC<sub>xM</sub> bits in TIM<sub>x</sub>\_CCMR<sub>x</sub> register) switches from the “frozen” configuration (no comparison, OC<sub>xM</sub>=‘000) to one of the PWM modes (OC<sub>xM</sub>=‘110 or ‘111).

This forces the PWM by software while the timer is running. The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM<sub>x</sub>\_CR1 register.

## PWM edge-aligned mode

### Upcounting configuration

Upcounting is active when the DIR bit in the TIM<sub>x</sub>\_CR1 register is low.

In the following example, we consider PWM mode 1. The reference PWM signal OC<sub>x</sub>REF is high as long as TIM<sub>x</sub>\_CNT < TIM<sub>x</sub>\_CCR<sub>x</sub>; else it becomes low. If the compare value in TIM<sub>x</sub>\_CCR<sub>x</sub> is greater than the auto-reload value (in TIM<sub>x</sub>\_ARR), then OC<sub>x</sub>REF is held at ‘1’. If the compare value is 0 then OC<sub>x</sub>REF is held at ‘0’. The following figure shows some edge-aligned PWM waveforms in an example where TIM<sub>x</sub>\_ARR = 8.

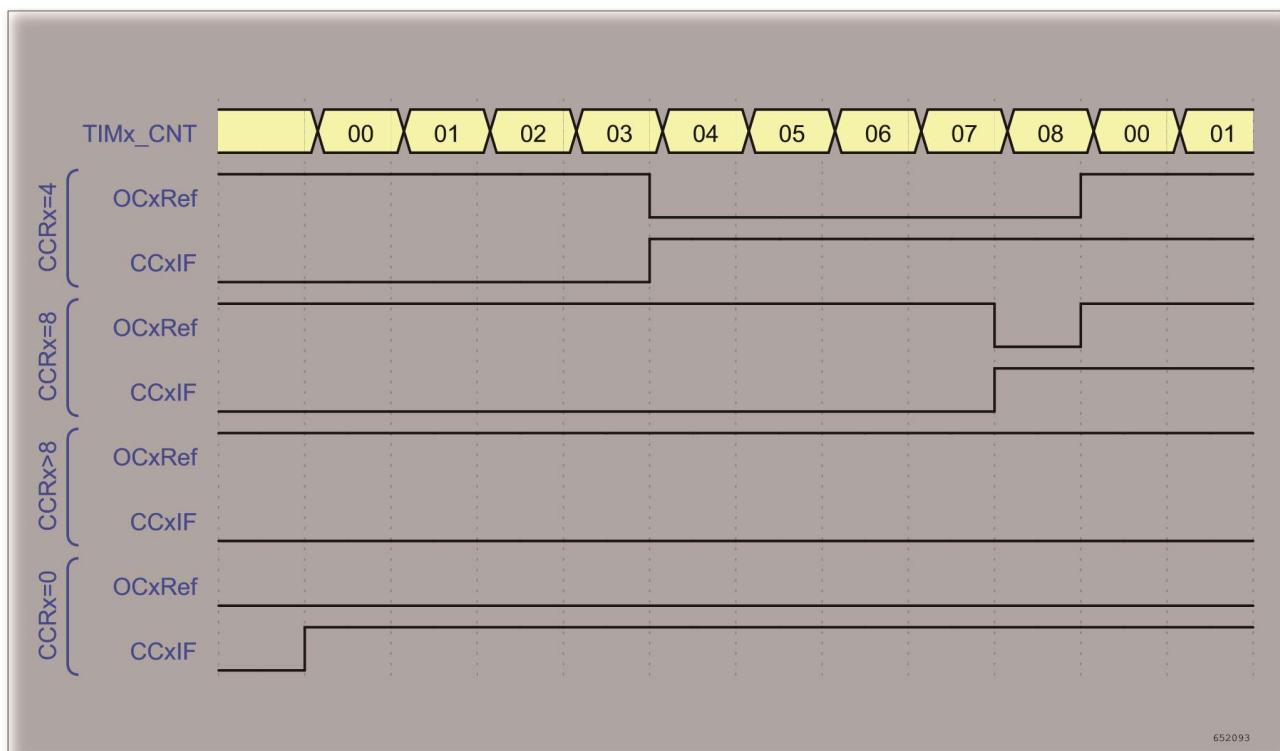


Figure 166. Edge-aligned PWM waveforms (ARR = 8)

## Downcounting configuration

Downcounting is active when DIR bit in TIMx\_CR1 register is high.

In PWM mode 1, the reference signal OCxREF is low as long as TIMx\_CNT > TIMx\_CCRx; else it becomes high. If the compare value in TIMx\_CCRx is greater than the auto-reload value in TIMx\_ARR, then OCxREF is held at '1'.

0% PWM is not possible in this mode.

## PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxREF/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx\_CR1 register is updated by hardware and must not be changed by software. Refer to center-aligned mode section.

The figure below shows some center-aligned PWM waveforms in an example where:

- TIMx\_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx\_CR1 register.

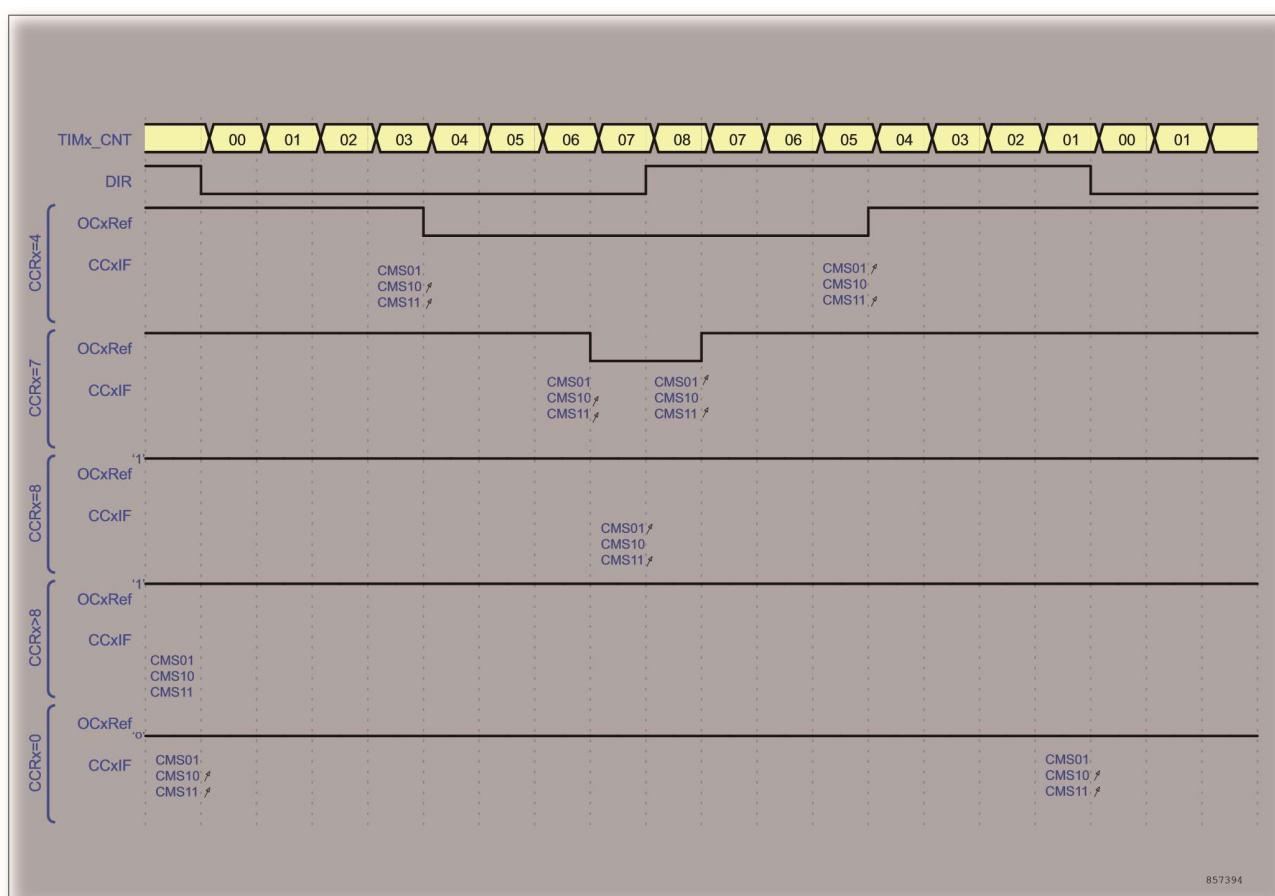


Figure 167. Center-aligned PWM waveforms (ARR = 8)

### Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if the user writes a value in the counter greater than the auto-reload value (TIMx\_CNT > TIMx\_ARR). For example, if the counter was counting up, it will continue to count up.
  - The direction is updated if the user writes 0 or writes the TIMx\_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx\_EGR register) just before starting the counter and not to write the counter while it is running.

### 17.3.10 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select One-pulse mode by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx  $\leq$  ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

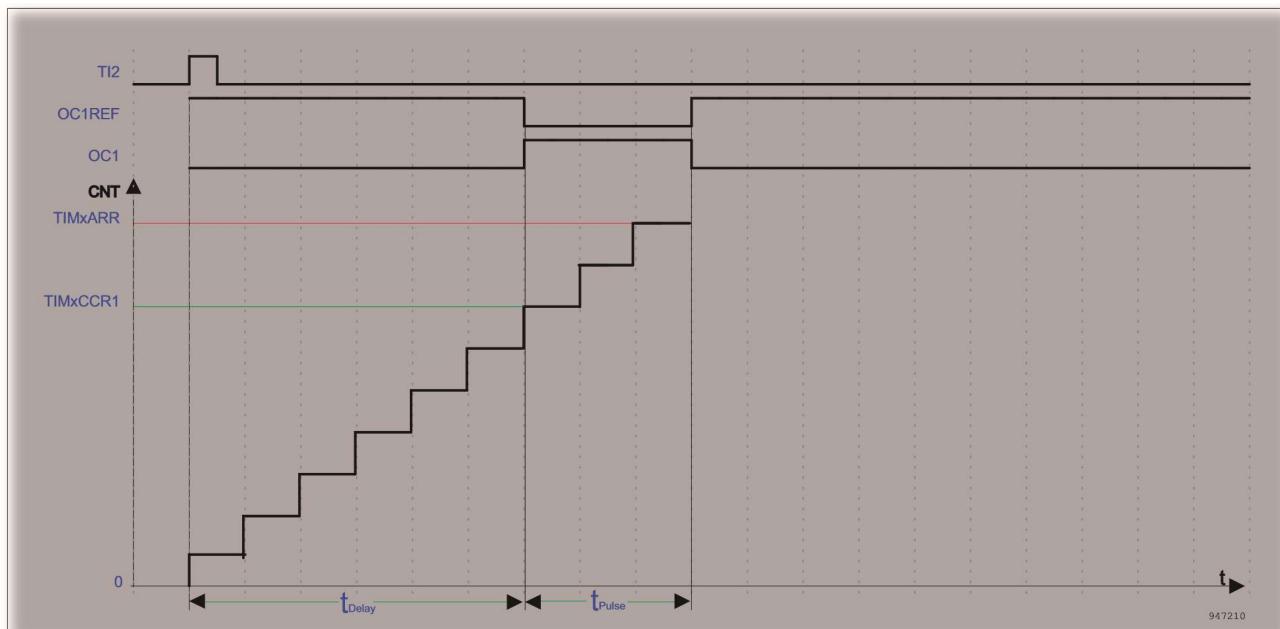


Figure 168. Example of one pulse mode

For example the user may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S to '01' in the TIMx\_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P to '0' in the TIMx\_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS to '110' in the TIMx\_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{DELAY}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{PULSE}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let us say the user wants to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value.

To do this, enable PWM mode 2 by writing OC1M = 111 in the TIMx\_CCMR1 register. The user can optionally enable the preload registers by writing OC1PE = 1 in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case the compare value must be written in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '1' in this example.

In our example, the DIR and CMS bits in the TIMx\_CR1 register should be low.

The user only wants one pulse, so '1' must be written in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

#### **Particular case: OCx fast enable:**

In One-pulse mode, the edge detection on TIx input sets the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{DELAY}$  min we can get.

If the user wants to output a waveform with the minimum delay, the OCxFE bit in the TIMx\_CCMRx register must be set. Then OCxREF (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

#### **17.3.11 Clearing the OCxREF signal on an external event**

The OCxREF signal can be cleared by setting the OCCS bit in the TIMx\_SMCR register. The OCxREF signal of a given channel can be pulled low when OCCS = 0 and a high level is applied on the ETRF input (OCxCE enable bit in the corresponding TIMx\_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. The OCxREF signal can be pulled low when OCCS = 1 and a high level is applied on the OCREF\_CLR (output from comparator, COMPx\_CSR[13:10]=1001 for TIM2 and COMPx\_CSR[13:10]=1011 for TIM3) signal.

This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

For example, the OCxREF signal can be connected to an external output. In this case, the ETR must be configured as follow:

- The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx\_SMCR register set to '00'.

- The external clock mode 2 must be disabled: bit ECE of the TIMx\_SMCR register set to '0'.
- The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

The figure below shows the behavior of the OCxREF signal corresponding to different OCxCE values when the ETRF input becomes High. In this example, the timer TIMx is programmed in PWM mode.

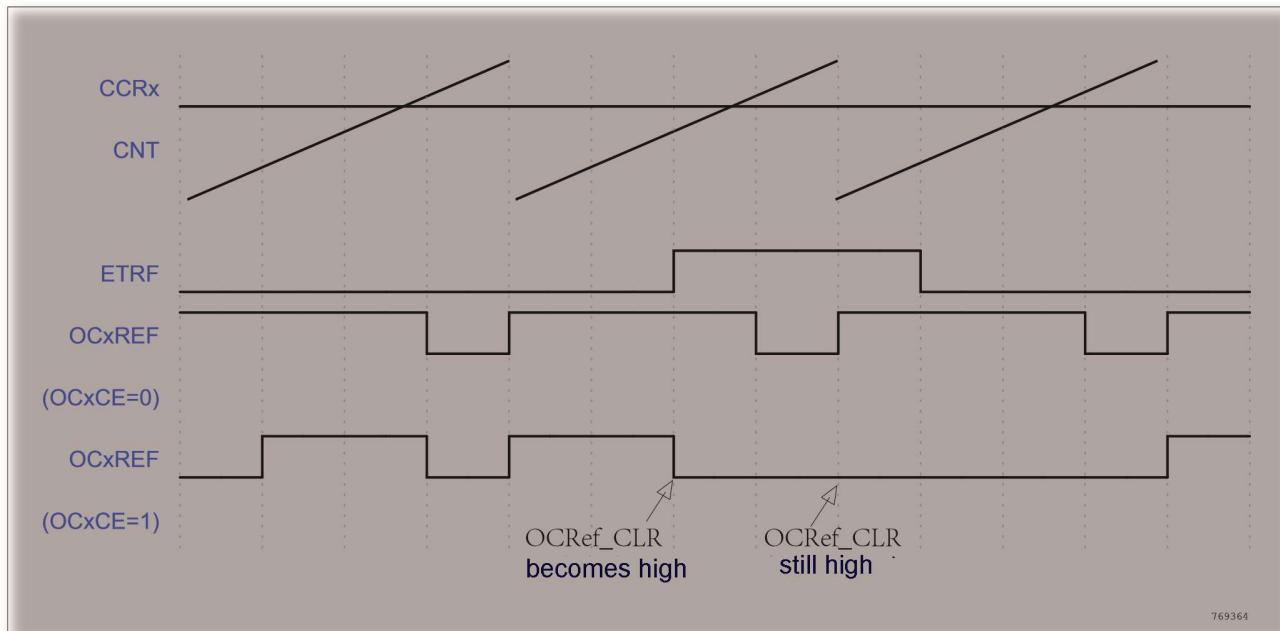


Figure 169. Clearing TIMx OCxREF

### 17.3.12 Encoder Interface mode

To select Encoder Interface mode, write SMS to '001' in the TIMx\_SMCR register if the counter is counting on TI2 edges only, SMS to '010' if it is counting on TI1 edges only and SMS to '011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx\_CCER register. When needed, the user can program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to table below. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx\_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence, the counter counts up or down, the DIR bit in the TIMx\_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx\_ARR register (0 to ARR or ARR down to 0 depending on the direction). So user must configure TIMx\_ARR before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The following table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 52. Counting direction versus encoder signals

Active edge	Level on opposite signal  (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI1FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset. The following figure gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points.

For this example we assume that the configuration is the following:

- C1S = '01' (TIMx\_CCMR1 register, IC1FP1 mapped on TI1).
- CC2S = '01' (TIMx\_CCMR2 register, IC2FP2 mapped on TI2).
- CC1P = '0' (TIMx\_CCER register, IC1FP1 non-inverted, IC1FP1 = TI1).
- CC2P = '0' (TIMx\_CCER register, IC2FP2 non-inverted, IC2FP2 = TI2).
- SMS = '011' (TIMx\_SMCR register, both inputs are active on both rising and falling edges)
- CEN = '1' (TIMx\_CR1 register, counter enabled).

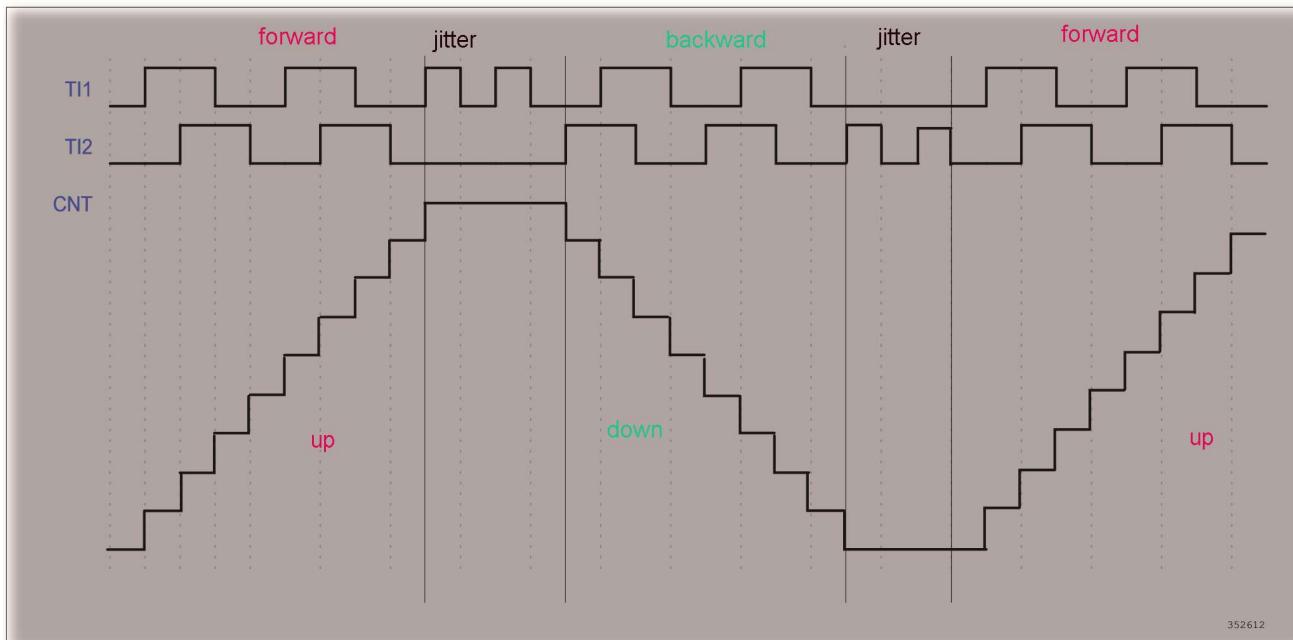


Figure 170. Example of counter operation in encoder interface mode

The following figure gives an example of counter behavior when IC1FP1 polarity is inverted (same configuration as above except CC1P = '1').

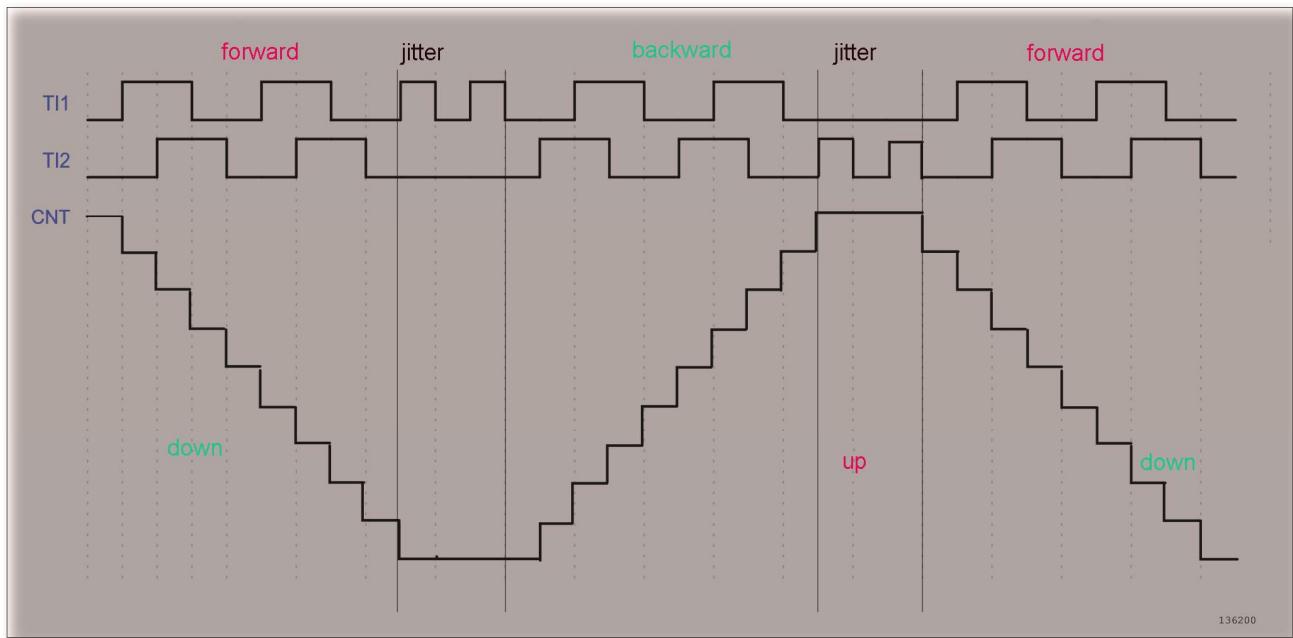


Figure 171. Example of encoder interface mode with IC1FP1 polarity inverted

The timer, when configured in Encoder Interface mode provides information on the sensor's current position. The user can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). It is also possible to read its value through a DMA request generated by a real-time clock.

### 17.3.13 Timer input XOR function

The TI1S bit in the TIMx\_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx\_CH1, TIMx\_CH2 and TIMx\_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. An example of this feature used to interface Hall sensors is given in Section 15.3.18.

### 17.3.14 Timers and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

#### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

- In the following example, the upcounter is cleared in response to a rising edge on TI1 input.
- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC1S bits select the input capture source only, i.e. CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P = 0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS = 101 in TIMx\_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR = 0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

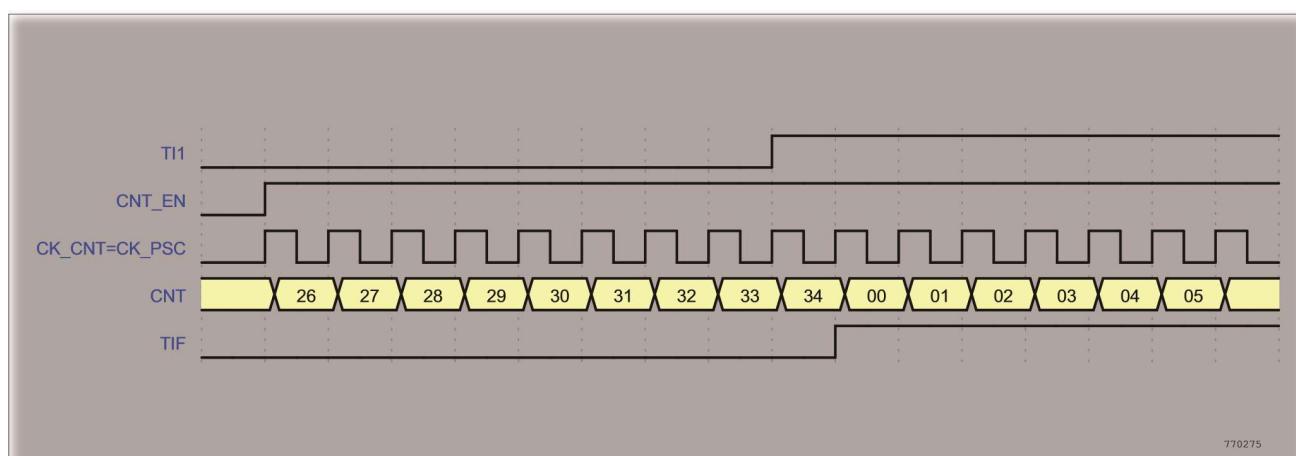


Figure 172. Control circuit in reset mode

### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so the user does not need to configure it. The CC1S bits select the input capture source only. Write CC1S = 01 in TIMx\_CCMR1 register. Write CC1P = 1 in TIMx\_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS = 101 in TIMx\_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set either when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

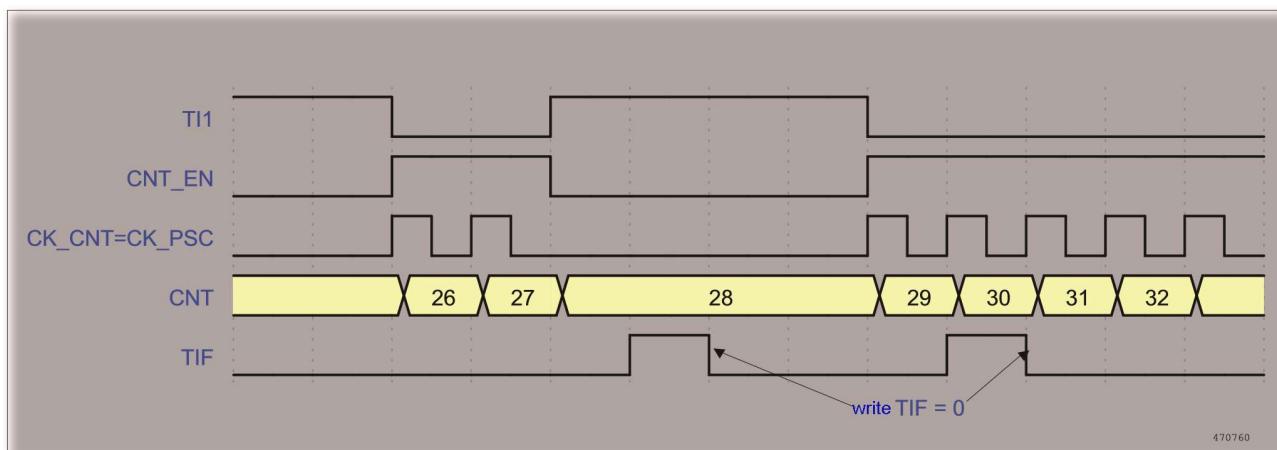


Figure 173. Control circuit in gated mode

### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC2S bits select the input capture source only. Write CC2S = 01 in TIMx\_CCMR1 register. Write CC1P = 1 in TIMx\_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIMx\_SMCR register. Select TI2 as the input source by writing TS = 110 in TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

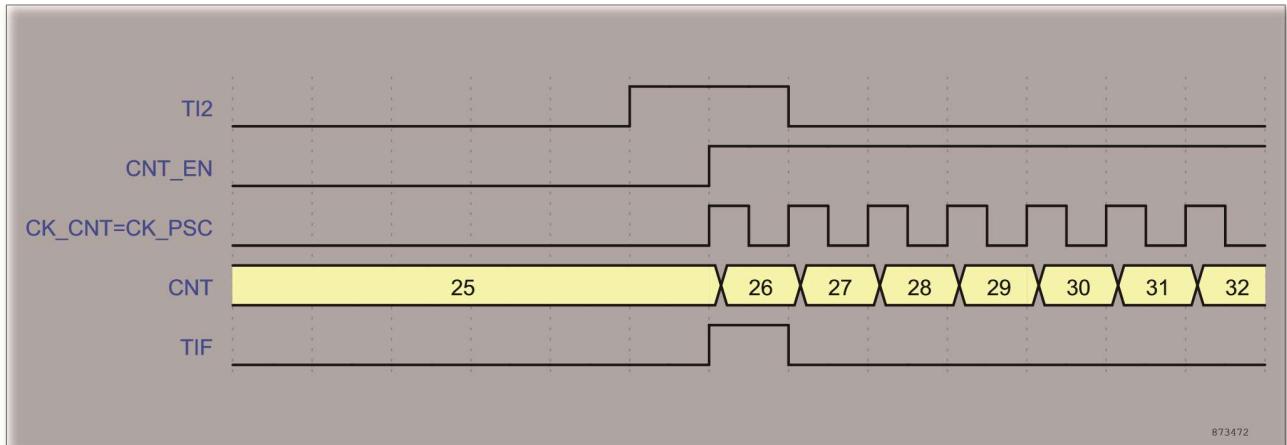


Figure 174. Control circuit in trigger mode

### Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx\_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

- Configure the external trigger input circuit by programming the TIMx\_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS = 00: prescaler disabled
  - ETP = 0: detection of rising edges on ETR and ECE = 1 to enable the external clock mode 2
- Configure the channel 1 as follows, to detect rising edges on TI:
  - IC1F = 0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S = 01 in TIMx\_CCMR1 register to select only the input capture source.
  - Write CC1P = 0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in trigger mode by writing SMS = 110 in the TIMx\_SMCR register. Select TI1 as the input source by writing TS = 101 in the TIMx\_SMCR register.

A rising edge on TI1 sets the TIF flag. The counter then counts on ETR rising edges. The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

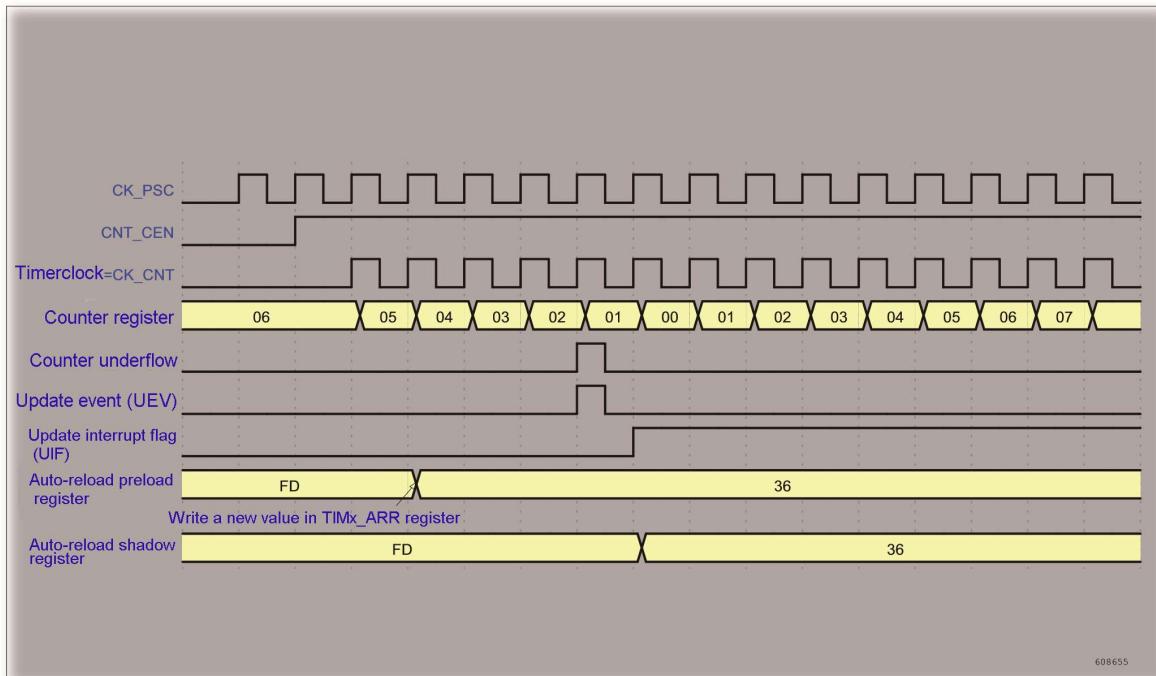


Figure 175. Control circuit in external clock mode 2 + trigger mode

### 17.3.15 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master mode, it can reset, start, stop or clock the counter of another Timer configured in Slave mode.

The figure below presents an overview of the trigger selection and the master mode selection blocks.

#### Using one timer as prescaler for another timer

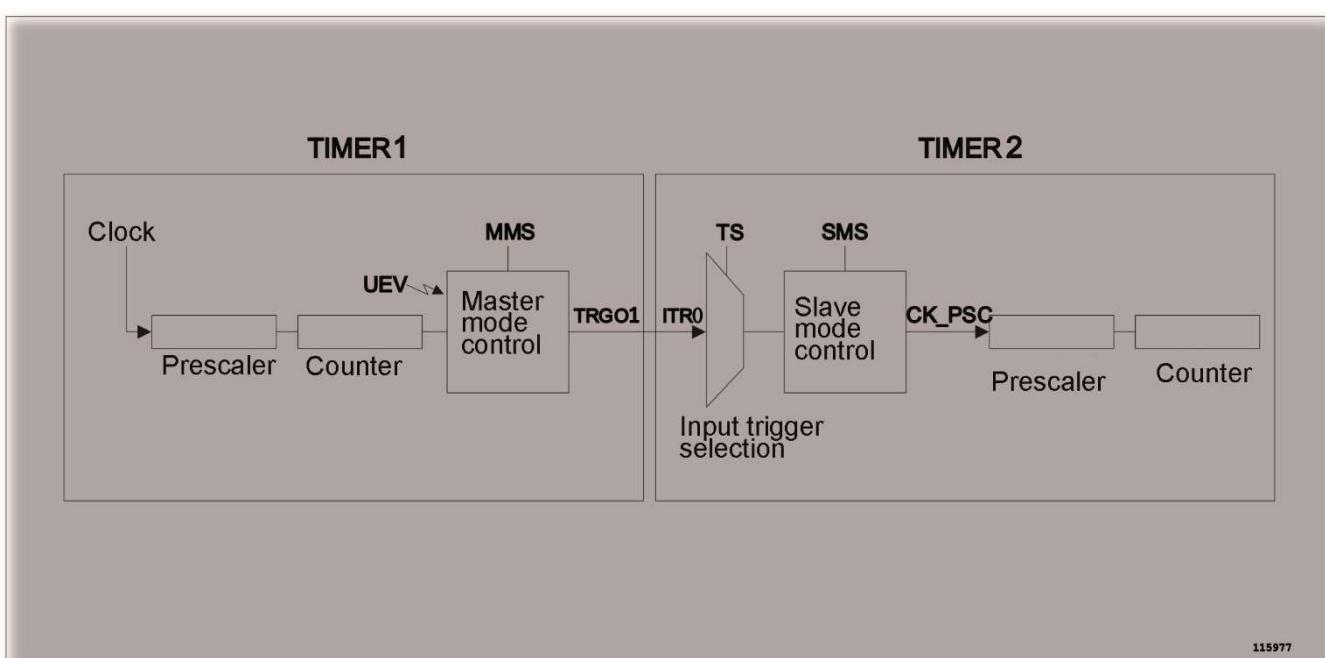


Figure 176. Master/Slave timer example

For example, the user can configure Timer 1 to act as a prescaler for Timer 2. Refer to the above diagram and perform the following operations:

- Configure Timer 1 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS = 010 in the TIM1\_CR2 register, a rising edge is output on TRGO1 each time an update event is generated.
- To connect the TRGO1 output of Timer 1 to Timer 2, Timer 2 must be configured in slave mode using ITR1 as internal trigger. You select this through the TS bits in the TIM2\_SMCR register (writing TS = 000).
- Then you put the slave mode controller in external clock mode 1 (write SMS = 111 in the TIM2\_SMCR register). This causes Timer 2 to be clocked by the rising edge of the periodic Timer 1 trigger signal (which corresponds to the Timer 1 counter overflow).
- Finally both timers must be enabled by setting their respective CEN bits (TIMx\_CR1 register).

Note: If OCx is selected on Timer 1 as trigger output (MMS = 1xx), its rising edge is used to clock the counter of Timer 2.

### Using one timer to enable another timer

In this example, we control the enable of Timer 2 with the output compare of Timer 1. Refer to Figure 177. Timer 2 counts on the divided internal clock only when OC1REF of Timer 1 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_CNT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

- Configure Timer 1 master mode to send its Output compare 1 Reference (OC1REF) signal as trigger output (MMS = 100 in the TIM1\_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1\_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS = 001 in the TIM2\_SMCR register).
- Configure Timer 2 in gated mode (SMS = 101 in TIM2\_SMCR register).
- Enable Timer 2 by writing '1' in the CEN bit (TIM2\_CR1 register).
- Enable Timer 1 by writing '1' in the CEN bit (TIM1\_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the Timer 2 counter enable signal.

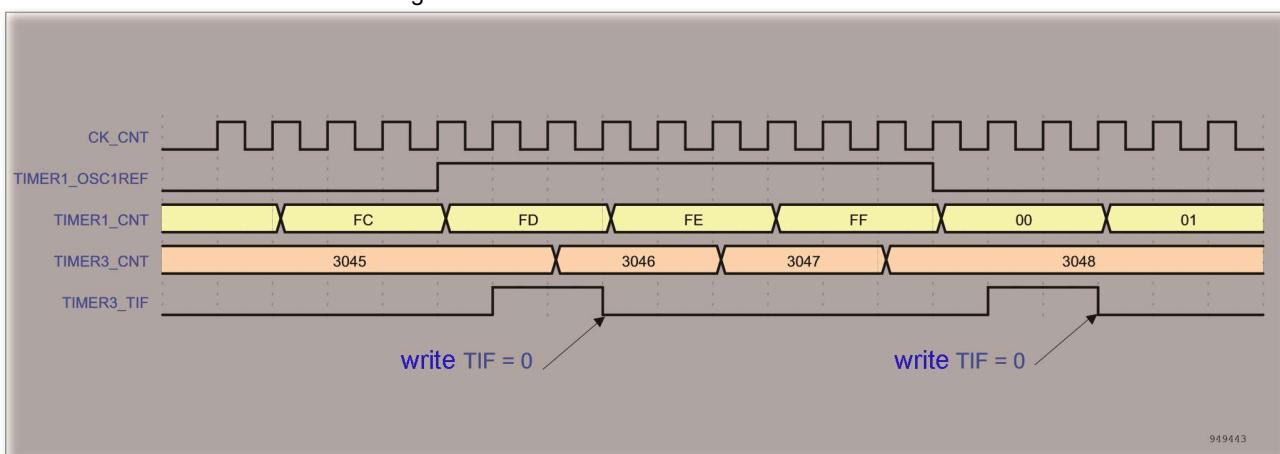


Figure 177. Gating timer 2 with OC1REF of timer 1

In the above example, the Timer 2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting Timer 1. You can then write any value you want in the timer counters. The timers can easily be reset by writing the UG bit in the TIMx\_EGR registers.

In the next example, we synchronize Timer 1 and Timer 2. Timer 1 is the master and starts from 0. Timer 2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer 2 stops when Timer 1 is disabled by writing '0' to the CEN bit in the TIM1\_CR1 register.

- Configure Timer 1 master mode to send its Output compare 1 Reference (OC1REF) signal as trigger output (MMS = 100 in the TIM1\_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1\_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS = 000 in the TIM2\_SMCR register).
- Configure Timer 2 in gated mode (SMS = 101 in TIM2\_SMCR register).
- Reset Timer 1 by writing '1' in UG bit (TIM1\_EGR register).
- Reset Timer 2 by writing '1' in UG bit (TIM2\_EGR register).
- Initialize Timer 2 to 0xE7 by writing '0xE7' in the timer 2 counter (TIM2\_CNT).
- Enable Timer 2 by writing '1' in the CEN bit (TIM2\_CR1 register).
- Start Timer 1 by writing '1' in the CEN bit (TIM1\_CR1 register).
- Stop Timer 1 by writing '0' in the CEN bit (TIM1\_CR1 register).

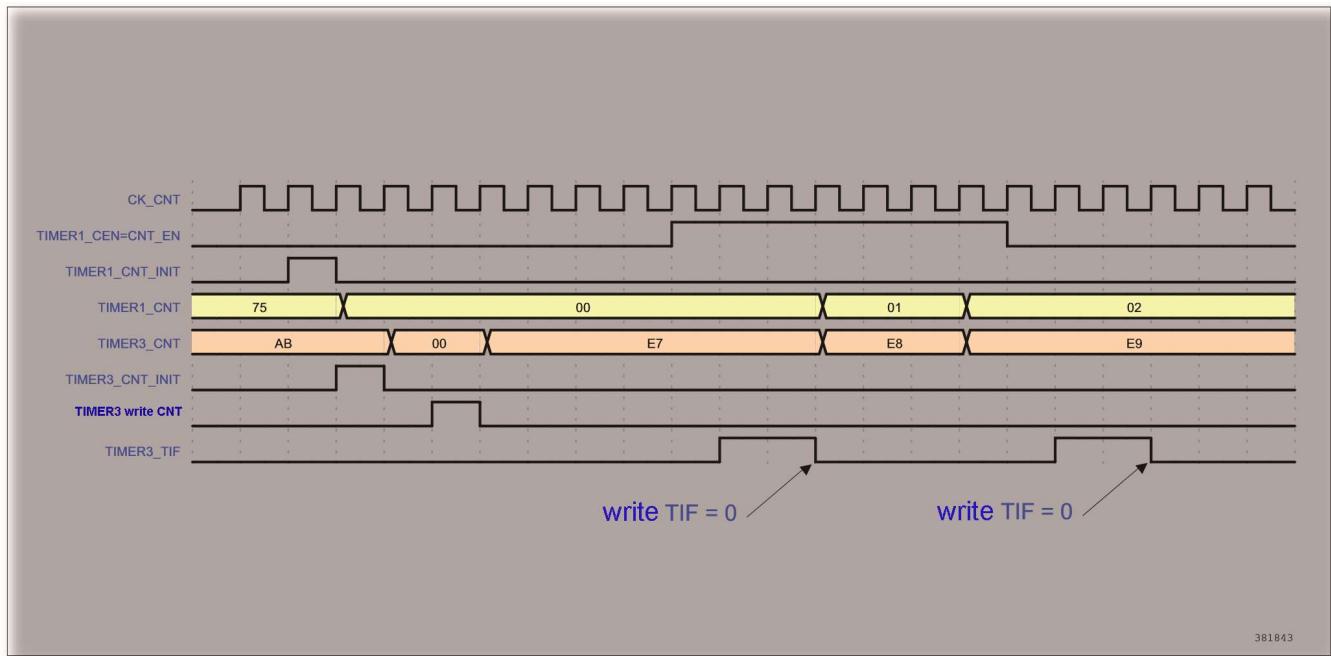


Figure 178. Gating timer 2 with enable of timer 1

### Using one timer to enable another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to Figure 179. Timer 2 starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal, its CEN bit is automatically set to '1' and the counter counts until we write '0' to the CEN bit in the TIM2\_CR1 register. Both

counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

- Configure Timer 1 master mode to send its Update Event (UEV) as trigger output (MMS = 010 in the TIM1\_CR2 register).
- Configure the Timer 1 period (TIM1\_ARR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS = 000 in the TIM2\_SMCR register).
- Configure Timer 2 in trigger mode (SMS = 110 in TIM2\_SMCR register).
- Start Timer 1 by writing '1' in the CEN bit (TIM1\_CR1 register).

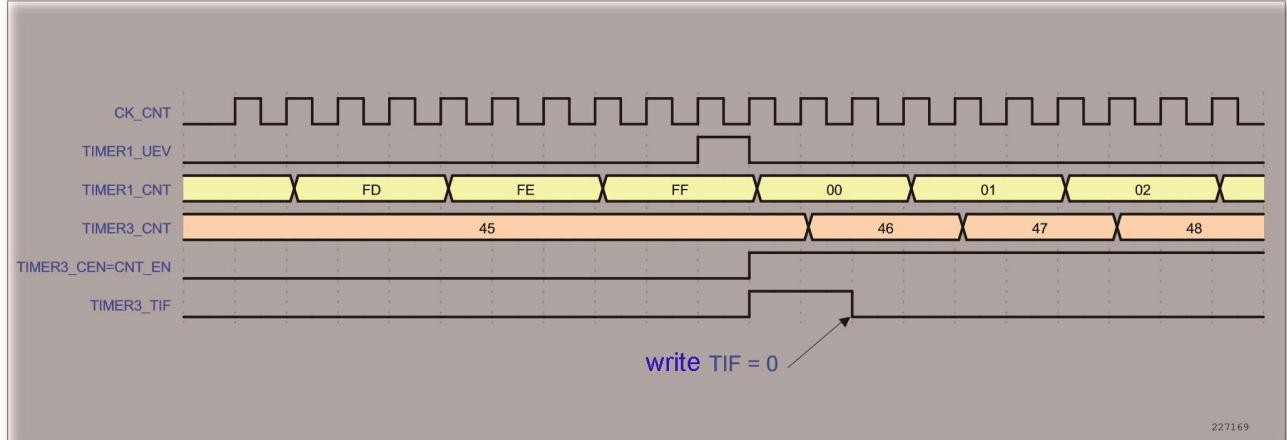


Figure 179. Triggering timer 2 with update of timer 1

As in the previous example, the user can initialize both counters before starting counting. The diagram below shows the behavior with the same configuration as 0 but in trigger mode instead of gated mode (SMS = 110 in the TIM2\_SMCR register).

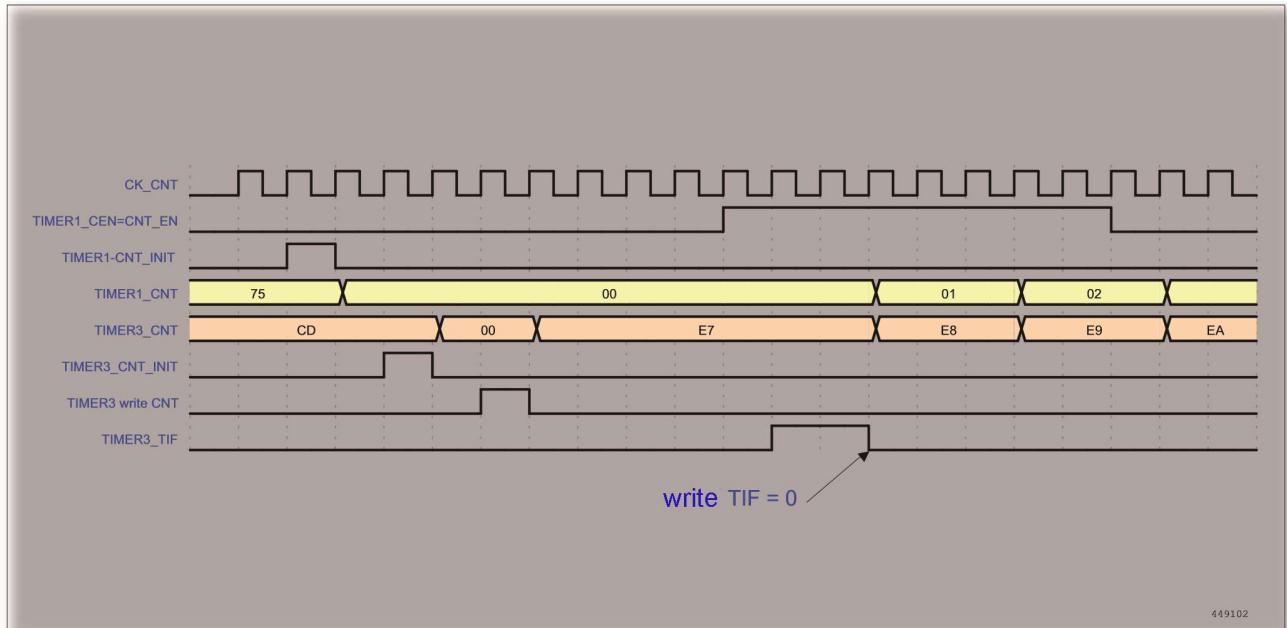


Figure 180. Triggering timer 2 with enable of timer 1

### Using one timer as prescaler for another timer

In this example, we use Timer 1 as prescaler for Timer 2. We configure as follows:

- Configure Timer 1 master mode to send its Update Event (UEV) as trigger output (MMS = 010 in the TIM1\_CR2 register). Then a periodic signal can be generated at each counter overflow.
- Configure the Timer 1 period (TIM1\_ARR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS = 000 in the TIM2\_SMCR register).
- Configure Timer 2 in external clock mode (SMS = 111 in TIM2\_SMCR register).
- Start Timer 2 by writing '1' in the CEN bit (TIM1\_CR2 register).
- Start Timer 1 by writing '1' in the CEN bit (TIM1\_CR1 register).

### Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of Timer 1 when its TI1 input rises, and the enable of Timer 2 with the enable of Timer 1. To ensure the counters are aligned, Timer 1 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to Timer 2):

- Configure Timer 1 master mode to send its Enable as trigger output (MMS = 001 in the TIM1\_CR2 register).
- Configure Timer 1 slave mode to get the input trigger from TI1 (TS = 100 in the TIM1\_SMCR register).
- Configure Timer 1 in trigger mode (SMS = 110 in the TIM1\_SMCR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS = 000 in the TIM2\_SMCR register).
- Configure Timer 2 in trigger mode (SMS = 110 in TIM2\_SMCR register).

When a rising edge occurs on TI1 (Timer 1), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but you can easily insert an offset between them by writing any of the counter registers (TIMx\_CNT). You can see from the following diagram that there is a delay between CNT\_EN and CK\_PSC on Timer 1 in the master/slave mode.

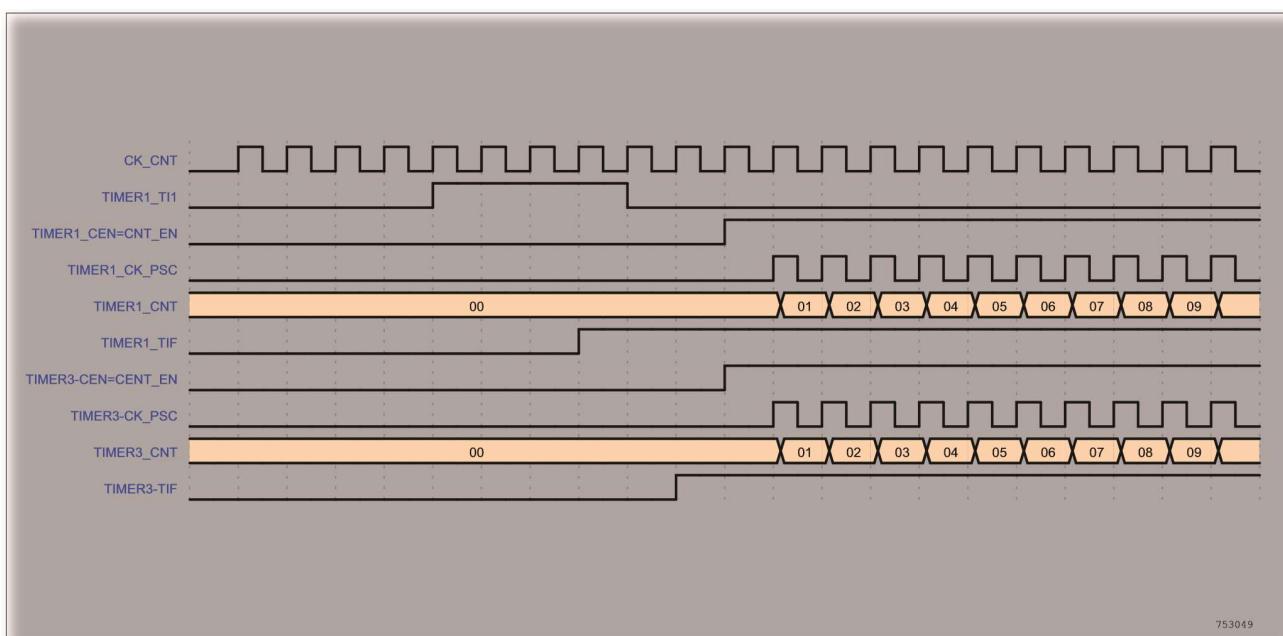


Figure 181. Triggering timer 1 and 2 with timer 1 TI1 input

### 17.3.16 Debug mode

When the microcontroller enters debug mode (CPU core - halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIMx\_STOP configuration bit in DBG module. For more details, refer to debug module section.

## 17.4 TIMx register description

---

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

Table 53. Overview of TIMx registers

Offset	Acronym	Register Name	Reset	Section
0x00	TIMx_CR1	Control register 1	0x00000000	Section 17.4.1
0x04	TIMx_CR2	Control register 2	0x00000000	Section 17.4.2
0x08	TIMx_SMCR	Slave mode control register	0x00000000	Section 17.4.3
0x0C	TIMx_DIER	DMA/interrupt enable register	0x00000000	Section 17.4.4
0x10	TIMx_SR	Status register	0x00000000	Section 17.4.5
0x14	TIMx_EGR	Event generate register	0x00000000	Section 17.4.6
0x18	TIMx_CCMR1	Capture/compare mode register 1	0x00000000	Section 17.4.7
0x1C	TIMx_CCMR2	Capture/compare mode register 2	0x00000000	Section 17.4.8
0x20	TIMx_CCER	Capture/compare enable register	0x00000000	Section 17.4.9
0x24	TIMx_CNT	Counter	0x00000000	Section 17.4.10
0x28	TIMx_PSC	Prescaler	0x00000000	Section 17.4.11
0x2C	TIMx_ARR	Auto-reload register	0x00000000	Section 17.4.12
0x34	TIMx_CCR1	Capture/compare register 1	0x00000000	Section 17.4.13
0x38	TIMx_CCR2	Capture/compare register 2	0x00000000	Section 17.4.14
0x3C	TIMx_CCR3	Capture/compare register 3	0x00000000	Section 17.4.15
0x40	TIMx_CCR4	Capture/compare register 4	0x00000000	Section 17.4.16
0x48	TIMx_DCR	DMA control register	0x00000000	Section 17.4.17
0x4C	TIMx_DMAR	DMA address for full transfer	0x00000000	Section 17.4.18

### 17.4.1 Control register 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Reserved		CKD	ARPE	CMS	DIR	OPM	URS	UDIS	CEN	

Bit	Field	Type	Reset	Description
15:10	Reserved			Reserved, read as 0

Bit	Field	Type	Reset	Description
9:8	CKD	rw	0x00	<p>Clock division</p> <p>These 2 bits indicate the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock used by the dead-time generators and the digital filters (ETR, TIx).</p> <p>00: <math>t_{DTS} = t_{CK\_INT}</math></p> <p>01: <math>t_{DTS} = 2 \times t_{CK\_INT}</math></p> <p>10: <math>t_{DTS} = 4 \times t_{CK\_INT}</math></p> <p>11: Reserved, do not program this value</p>
7	ARPE	rw	0x00	<p>Auto-reload preload enable</p> <p>0: TIMx_ARR register is not buffered</p> <p>1: TIMx_ARR register is buffered</p>
6:5	CMS	rw	0x00	<p>Center-aligned mode selection</p> <p>00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).</p> <p>01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.</p> <p>10: Center-aligned mode 2. The counter counts up and down alternatively. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.</p> <p>11: Center-aligned mode 3. The counter counts up and down alternatively. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.</p> <p>Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)</p>
4	DIR	rw	0x00	<p>Direction</p> <p>0: Counter used as upcounter</p> <p>1: Counter used as downcounter</p> <p>Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.</p>

3	OPM	rw	0x00	One pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit CEN).
Bit	Field	Type	Reset	Description
2	URS	rw	0x00	Update request source This bit is set by software to select the UEV event sources. 0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be: - Counter overflow/underflow - Setting the UG bit - Update generation through the slave mode controller 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.
1	UDIS	rw	0x00	Update disable This bit is set by software to enable/disable UEV event generation. 0: UEV enabled. The Update (UEV) event is generated by one of the following events: - Counter overflow/underflow - Setting the UG bit - Update generation through the slave mode controller Buffered registers are then loaded with their preload values. 1: UEV disabled. The Update event is not generated, shadow registers keep their values (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is sent from the slave mode controller.
0	CEN	rw	0x00	Counter enable 0: Counter disabled 1: Counter enabled Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

#### 17.4.2 Control register 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TI1S	MMS	CCDS	Reserved								
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 8	Reserved		Reserved, read as 0	
Bit	Field	Type	Reset	Description
7	TI1S	rw	0x00	<p>TI1 selection</p> <p>0: The TIMx_CH1 pin is connected to TI1 input</p> <p>1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)</p>
6:4	MMS	rw	0x00	<p>Master mode selection</p> <p>These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:</p> <p>000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.</p> <p>001: Enable - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).</p> <p>010: Update - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.</p> <p>011: Compare Pulse - The trigger output (TRGO) send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.</p> <p>100: Compare - OC1REF signal is used as trigger output (TRGO)</p> <p>101: Compare - OC2REF signal is used as trigger output (TRGO)</p> <p>110: Compare - OC3REF signal is used as trigger output (TRGO)</p> <p>111: Compare - OC4REF signal is used as trigger output (TRGO)</p>
3	CCDS	rw	0x00	<p>Capture/compare DMA selection</p> <p>0: CCx DMA request sent when CCx event occurs</p> <p>1: CCx DMA requests sent when update event occurs</p>

2: 0	Reserved	Reserved and read as 0.
------	----------	-------------------------

### 17.4.3 Slave mode control register (TIMx\_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF		MSM		TS		OCCS		SMS			

Bit	Field	Type	Reset	Description
15	ETP	rw	0x00	<p>External trigger polarity</p> <p>This bit selects whether ETR or inverted ETR is used for trigger operations.</p> <p>0: ETR is non-inverted, active at high level or rising edge</p> <p>1: ETR is inverted, active at low level or falling edge</p>
14	ECE	rw	0x00	<p>External clock enable</p> <p>This bit enables External clock mode 2.</p> <p>0: External clock mode 2 disabled</p> <p>1: External clock mode 2 enabled. The counter is clocked by any active rising edge on the ETRF signal.</p> <p>Note 1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS = 111 and TS = 111).</p> <p>Note 2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).</p> <p>Note 3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.</p>
13:12	ETPS	rw	0x00	<p>External trigger prescaler</p> <p>External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.</p> <p>00: Prescaler OFF</p> <p>01: ETRP frequency divided by 2</p> <p>10: ETRP frequency divided by 4</p> <p>11: ETRP frequency divided by 8</p>

Bit	Field	Type	Reset	Description
11:8	ETF	rw	0x00	<p>External trigger filter</p> <p>This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:</p> <ul style="list-style-type: none"> <li>0000: No filter, sampling is done at <math>f_{DTS}</math></li> <li>0001: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 2</li> <li>0010: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 4</li> <li>0011: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 8</li> <li>0100: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 6</li> <li>0101: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 8</li> <li>0110: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 6</li> <li>0111: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 8</li> <li>1000: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 6</li> <li>1001: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 8</li> <li>1010: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 5</li> <li>1011: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 6</li> <li>1100: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 8</li> <li>1101: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 5</li> <li>1110: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 6</li> <li>1111: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 8</li> </ul>
7	MSM	rw	0x00	<p>Master/slave mode</p> <p>0: No action</p> <p>1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.</p>

Bit	Field	Type	Reset	Description
6:4	TS	rw	0x00	<p>Trigger selection</p> <p>This bit-field selects the trigger input to be used to synchronize the counter.</p> <ul style="list-style-type: none"> <li>000: Internal Trigger 0 (ITR0)</li> <li>001: Internal Trigger 1 (ITR1)</li> <li>010: Internal Trigger 2 (ITR2)</li> <li>011: Internal Trigger 3 (ITR3)</li> <li>100: TI1 Edge Detector (TI1F_ED)</li> <li>101: Filtered Timer Input 1 (TI1FP1)</li> <li>110: Filtered Timer Input 2 (TI2FP2)</li> <li>111: External Trigger input (ETRF)</li> </ul> <p>See the table below for more details on ITRx.</p> <p>Note: These bits must be changed only when they are not used (e.g. when SMS = 000) to avoid wrong edge detections at the transition.</p>
3	OCCS	rw	0x00	<p>Output compare clear selection</p> <p>This bit is used to clear the comparator output in the PWM mode.</p> <ul style="list-style-type: none"> <li>1: The comparator output is considered as the clear signal</li> <li>0: The external trigger input is considered as the clear signal.</li> </ul>

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

2:0	SMS	rw	0x00	Slave mode selection When external signals are selected, the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control register description). 000: Slave mode disabled - if CEN = 1, then the prescaler is clocked directly by the internal clock. 001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level. 010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level. 011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input. 100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers. 101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled. 110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled. 111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter. Note: The gated mode must not be used if TI1F_EN is selected as the trigger input (TS = 100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.
-----	-----	----	------	---

Table 54. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	None	TIM3	None
TIM3	TIM1	TIM2	None	None

#### 17.4.4DMA/Interrupt enable register (TIMx\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE

Bit	Field	Type	Reset	Description
15	Reserved			Reserved, read as 0

Bit	Field	Type	Reset	Description
14	TDE	rw	0x00	Trigger DMA request enable 0: Trigger DMA request disabled 1: Trigger DMA request enabled
13	Reserved			Reserved and read as 0.
12	CC4DE	rw	0x00	Capture/Compare 4 DMA request enable 0: Capture/Compare 4 DMA request disabled 1: Capture/Compare 4 DMA request enabled
11	CC3DE	rw	0x00	Capture/Compare 3 DMA request enable 0: Capture/Compare 3 DMA request disabled 1: Capture/Compare 3 DMA request enabled
10	CC2DE	rw	0x00	Capture/Compare 2 DMA request enable 0: Capture/Compare 2 DMA request disabled 1: Capture/Compare 2 DMA request enabled
9	CC1DE	rw	0x00	Capture/Compare 1 DMA request enable 0: Capture/Compare 1 DMA request disabled 1: Capture/Compare 1 DMA request enabled
8	UDE	rw	0x00	Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled
7	Reserved			Reserved and read as 0.
6	TIE	rw	0x00	Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
5	Reserved			Reserved and read as 0.
4	CC4IE	rw	0x00	Capture/Compare 4 interrupt enable 0: Capture/Compare 4 interrupt disabled 1: Capture/Compare 4 interrupt enabled
3	CC3IE	rw	0x00	Capture/Compare 3 interrupt enable 0: Capture/Compare 3 interrupt disabled 1: Capture/Compare 3 interrupt enabled
2	CC2IE	rw	0x00	Capture/Compare 2 interrupt enable 0: Capture/Compare 2 interrupt disabled 1: Capture/Compare 2 interrupt enabled

Bit	Field	Type	Reset	Description
1	CC1IE	rw	0x00	Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt disabled 1: Capture/Compare 1 interrupt enabled
0	UIE	rw	0x00	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

#### 17.4.5 Status register (TIMx\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		CC4OF	CC3OF	CC2OF	CC1OF	Res.		TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF	

rc\_w0      rc\_w0      rc\_w0      rc\_w0           rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0

Bit	Field	Type	Reset	Description
15:13	Reserved			Reserved, read as 0
12	CC4OF	rc_w0	0x00	Capture/Compare 4 overcapture flag Refer to CC1OF description.
11	CC3OF	rc_w0	0x00	Capture/Compare 3 overcapture flag Refer to CC1OF description.
10	CC2OF	rc_w0	0x00	Capture/Compare 2 overcapture flag Refer to CC1OF description.
9	CC1OF	rc_w0	0x00	Capture/Compare 1 overcapture flag This flag is set to '1' by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'. 0: No overcapture has been detected. 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set to 1.
8: 7	Reserved			Reserved and read as 0.
6	TIF	rc_w0	0x00	Trigger interrupt flag This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software. 0: No trigger event occurred. 1: Trigger interrupt pending.
5	Reserved			Reserved and read as 0.

Bit	Field	Type	Reset	Description
4	CC4IF	rc_w	00x00	Capture/Compare 4 interrupt flag Refer to CC1IF description.
3	CC3IF	rc_w0	0x00	Capture/Compare 3 interrupt flag Refer to CC1IF description.
2	CC2IF	rc_w0	0x00	Capture/Compare 2 interrupt flag Refer to CC1IF description.
1	CC1IF	rc_w0	0x00	Capture/Compare 1 interrupt flag If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software. 0: No match. 1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register. 0: No input capture occurred 1: The counter value has been captured in (copied to) TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)
0	UIF	rc_w0	0x00	Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update occurred. 1: Update event pending. This bit is set by hardware when the registers are updated: - At overflow or underflow regarding the repetition downcounter value (update if REP_CNT = 0) and if the UDIS = 0 in the TIMx_CR1 register. - When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register. - When CNT is reinitialized by a trigger event (refer to the synchronous control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

#### 17.4.6 Event generate register (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								TG	COMG	CC4G	CC3G	CC2G	CC1G	UG	
								w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15:7	Reserved			Reserved, read as 0
6	TG	w	0x00	<p>Trigger generation</p> <p>This bit is set by software in order to generate a brake event, it is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA can occur if enabled.</p>
5	COMG	w	0x00	<p>Capture/Compare control update generation</p> <p>This bit can be set by software, it is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits.</p> <p>Note: This bit acts only on channels that have a complementary output.</p>
4	CC4G	w	0x00	Capture/Compare 4 generation Refer to CC1G description.
3	CC3G	w	0x00	Capture/Compare 3 generation Refer to CC1G description.
2	CC2G	w	0x00	Capture/Compare 2 generation Refer to CC1G description.
1	CC1G	w	0x00	<p>Capture/Compare 1 generation</p> <p>This bit is set by software in order to generate a capture/compare event, it is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: A capture/compare event is generated on channel CC1:</p> <p>If channel CC1 is configured as output: CC1IF flag is set, corresponding interrupt or DMA request is sent if enabled.</p> <p>If channel CC1 is configured as input: The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.</p>
0	UG	w	0x00	<p>Update generation</p> <p>This bit can be set by software, it is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: Reinitialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR = 1 (downcounting).</p>

### 17.4.7 Capture/compare mode register 1 (TIMx\_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CC<sub>x</sub>S bits. All the other bits of this register have a different function in output mode. For a given bit, OC<sub>xx</sub> describes its function when the channel is configured in output, IC<sub>xx</sub> describes its function when the channel is configured in input. So the user must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	11:	11	10	9	8	7	6	5	3	2	1	0
OC2CE	OC2M		OC2PE	OC2FE		CC2S	OC1CE	OC1M	OC1PE	OC1FE		CC1S		
	IC2F			IC2PSC				IC1F			IC1PSC			

#### Output compare mode:

Bit	Field	Type	Reset	Description
15	OC2CE	rw	0x00	Output compare 2 clear enable
14:12	OC2M	rw	0x00	Output compare 2 mode
11	OC2PE	rw	0x00	Output compare 2 preload enable
10	OC2FE	rw	0x00	Output compare 2 fast enable
9:8	CC2S	rw	0x00	Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register). Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).
7	OC1CE	rw	0x00	Output compare 1 clear enable 0: OC1REF is not affected by the ETRF input 1: OC1REF is cleared as soon as a high level is detected on ETRF input

Bit	Field	Type	Reset	Description
6:4	OC1M	rw	0x00	<p>Output compare 1 mode</p> <p>These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.</p> <p>000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the OC1REF.</p> <p>001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).</p> <p>010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).</p> <p>011: Toggle. OC1REF level toggles when TIMx_CCR1=TIMx_CNT.</p> <p>100: Force inactive level. OC1REF is forced low.</p> <p>101: Force active level. OC1REF is forced high.</p> <p>110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT &lt; TIMx_CCR1, else inactive. In downcounting, channel 1 is inactive (OC1REF = 0) as long as TIMx_CNT &gt; TIMx_CCR1, else active (OC1REF = 1).</p> <p>111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT &lt; TIMx_CCR1, else active. In downcounting, channel 1 is active as long as TIMx_CNT &gt; TIMx_CCR1, else inactive.</p> <p>Note 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).</p> <p>Note 2: In PWM mode 1 or 2, the OC1REF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.</p>
3	OC1PE	rw	0x00	<p>Output compare 1 preload enable</p> <p>0: Preload function of TIMx_CCR1 register disabled. TIMx_CCR1 register can be written at anytime, the new value is taken in account immediately.</p> <p>1: Preload function of TIMx_CCR1 register enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.</p> <p>Note 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).</p> <p>Note 2:</p> <p>The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.</p>

Bit	Field	Type	Reset	Description
2	OC1FE	rw	0x00	<p>Output compare 1 fast enable This bit is used to accelerate the effect of an event on the trigger input on the CC output.</p> <p>0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.</p> <p>1: An active edge on the trigger input acts like a compare match. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.</p>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC1 channel is configured as output</li> <li>01: CC1 channel is configured as input, IC1 is mapped on TI1</li> <li>10: CC1 channel is configured as input, IC1 is mapped on TI2</li> <li>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)</li> </ul> <p>Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).</p>

### Input capture mode:

Bit	Field	Type	Reset	Description
15:12	IC2F	rw	0x00	Input capture 2 filter
11:10	IC2PSC	rw	0x00	Input capture 2 prescaler
9:8	CC2S	rw	0x00	<p>Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC2 channel is configured as output.</li> <li>01: CC2 channel is configured as input, IC2 is mapped on TI2.</li> <li>10: CC2 channel is configured as input, IC2 is mapped on TI1.</li> <li>11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register).</li> </ul> <p>Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).</p>

Bit	Field	Type	Reset	Description
7:4	IC1F	rw	0x00	<p>Input capture 1 filter</p> <p>This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to generate a transition on the output:</p> <ul style="list-style-type: none"> <li>0000: No filter, sampling is done at <math>f_{DTS}</math></li> <li>1000: <math>f_{SAMPLING}=f_{DTS}/8, N = 6</math></li> <li>0001: <math>f_{SAMPLING}=f_{CK\_INT}, N = 2</math></li> <li>1001: <math>f_{SAMPLING}=f_{DTS}/8, N = 8</math></li> <li>0010: <math>f_{SAMPLING}=f_{CK\_INT}, N = 4</math></li> <li>1010: <math>f_{SAMPLING}=f_{DTS}/16, N = 5</math></li> <li>0011: <math>f_{SAMPLING}=f_{CK\_INT}, N = 8</math></li> <li>1011: <math>f_{SAMPLING}=f_{DTS}/16, N = 6</math></li> <li>0100: <math>f_{SAMPLING}=f_{DTS}/2, N = 6</math></li> <li>1100: <math>f_{SAMPLING}=f_{DTS}/16, N = 8</math></li> <li>0101: <math>f_{SAMPLING}=f_{DTS}/2, N = 8</math></li> <li>1101: <math>f_{SAMPLING}=f_{DTS}/32, N = 5</math></li> <li>0110: <math>f_{SAMPLING}=f_{DTS}/4, N = 6</math></li> <li>1110: <math>f_{SAMPLING}=f_{DTS}/32, N = 6</math></li> <li>0111: <math>f_{SAMPLING}=f_{DTS}/4, N = 8</math></li> <li>1111: <math>f_{SAMPLING}=f_{DTS}/32, N = 8</math></li> </ul>
3:2	IC1PSC	rw	0x00	<p>Input capture 1 prescaler</p> <p>This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).</p> <p>The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register). 00: no prescaler, capture is done each time an edge is detected on the capture input</p> <ul style="list-style-type: none"> <li>01: capture is done once every 2 events</li> <li>10: capture is done once every 4 events</li> <li>11: capture is done once every 8 events</li> </ul>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC1 channel is configured as output</li> <li>01: CC1 channel is configured as input, IC1 is mapped on TI1</li> <li>10: CC1 channel is configured as input, IC1 is mapped on TI2</li> <li>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)</li> </ul> <p>Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).</p>

### 17.4.8 Capture/compare mode register 2 (TIMx\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M		OC4PE	OC4FE			CC4S	OC3CE	OC3M		OC3PE	OC3FE		CC3S	
	IC4F			IC4PSC				IC3F		IC3PSC					

### Output Compare mode

Bit	Field	Type	Reset	Description
15	OC4CE	rw	0x00	Output compare 4 clear enable
14:12	OC4M	rw	0x00	Output compare 4 mode
11	OC4PE	rw	0x00	Output compare 4 preload enable
10	OC4FE	rw	0x00	Output compare 4 fast enable
9:8	CC4S	rw	0x00	Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the used input: 00: CC4 channel is configured as output 01: CC4 channel is configured as input, IC4 is mapped on TI4 10: CC4 channel is configured as input, IC4 is mapped on TI3 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register). Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).
7	OC3CE	rw	0x00	Output compare 3 clear enable
6:4	OC3M	rw	0x00	Output compare 3 mode
3	OC3PE	rw	0x00	Output compare 3 preload enable
2	OC3FE	rw	0x00	Output compare 3 fast enable

Bit	Field	Type	Reset	Description
1:0	CC3S	rw	0x00	<p>Capture/Compare 3 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC3 channel is configured as output</li> <li>01: CC3 channel is configured as input, IC3 is mapped on TI3</li> <li>10: CC3 channel is configured as input, IC3 is mapped on TI4</li> <li>11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register).</li> </ul> <p>Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).</p>

### Input Capture mode

Bit	Field	Type	Reset	Description
15:12	IC4F	rw	0x00	Input capture 4 filter
11:10	IC4PSC	rw	0x00	Input capture 4 prescaler
9:8	CC4S	rw	0x00	<p>Capture/Compare 4 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC4 channel is configured as output</li> <li>01: CC4 channel is configured as input, IC4 is mapped on TI4</li> <li>10: CC4 channel is configured as input, IC4 is mapped on TI3</li> <li>11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register).</li> </ul> <p>Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).</p>
7:4	IC3F	rw	0x00	Input capture 3 filter
3:2	IC3PSC	rw	0x00	Input capture 3 prescaler

Bit	Field	Type	Reset	Description
1:0	CC3S	rw	0x00	<p>Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC3 channel is configured as output</li> <li>01: CC3 channel is configured as input, IC3 is mapped on TI3</li> <li>10: CC3 channel is configured as input, IC3 is mapped on TI4</li> <li>11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register).</li> </ul> <p>Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).</p>

#### 17.4.9 Capture/compare enable register (TIMx\_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CC4P	CC4E	Res.	CC3P	CC3E	Res.	CC2P	CC2E	Res.	CC1P	CC1E				
	rw	rw		rw		rw									

Bit	Field	Type	Reset	Description
15: 14	Reserved			Reserved, always read as 0.
13	CC4P	rw	0x00	Capture/Compare 4 output polarity Refer to CC1P description.
12	CC4E	rw	0x00	Capture/Compare 4 output enable Refer to CC1E description.
11: 10	Reserved			Reserved and always read as 0.
9	CC3P	rw	0x00	Capture/Compare 3 output polarity Refer to CC1P description.
8	CC3E	rw	0x00	Capture/Compare 3 output enable Refer to CC1E description.
7: 6	Reserved			Reserved and always read as 0.
5	CC2P	rw	0x00	Capture/Compare 2 output polarity Refer to CC1P description.
4	CC2E	rw	0x00	Capture/Compare 2 output enable Refer to CC1E description.
3: 2	Reserved			Reserved and always read as 0.

Bit	Field	Type	Reset	Description
1	CC1P	rw	0x00	<p>Capture/Compare 1 output polarity CC1 channel configured as output: 0: OC1 active high 1: OC1 active low</p> <p>CC1 channel configured as input: This bit selects whether the inverted signal of IC1 or IC1 is used for trigger or capture operations. 0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted. 1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted.</p> <p>Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).</p>
0	CC1E	rw	0x00	<p>Capture/Compare 1 output enable CC1 channel configured as output: 0: Off - OC1 is not active. OC1 level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits. 1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits.</p> <p>CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the TIMx_CCR1 register or not. 0: capture disabled 1: capture enabled</p>

Table 55. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (OCx = 0, OCx_EN = 0)
1	OCx = OCxREF + Polarity, OCx_EN = 1

Note: The states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and the GPIO and AFIO registers.

#### 17.4.10 Counter (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:16	CNT	rw	0x0000	16 higher bits of the counter (High counter value)
15:0	CNT	rw	0x0000	16 lower bits of the counter (Low counter value)

#### 17.4.11 Prescaler (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description
15:0	PSC	rw	0x0000	<p>Prescaler value</p> <p>The counter clock frequency (CK_CNT) is equal to <math>f_{CK\_PSC}/(PSC + 1)</math>.</p> <p>PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through slave controller when configured in “reset mode”).</p>

#### 17.4.12 Auto-reload register (TIMx\_ARR)

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:16	ARR	rw	0x0000	High auto-reload value
15:0	ARR	rw	0x0000	<p>Low auto-reload value</p> <p>ARR is the value to be loaded in the actual auto-reload register.</p> <p>Refer to the Section 17.3.1 for more details about ARR update and behavior.</p> <p>The counter is blocked while the auto-reload value is null.</p>

### 17.4.13 Capture/compare register 1 (TIMx\_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:16	CCR1	rw	0x0000	High Capture/Compare 1 value (higher 16 bits)
15:0	CCR1	rw	0x0000	If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded immediately if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output. If channel CC1 is configured as input: CCR1 is the counter value transferred by the last input capture 1 event (IC1).

### 17.4.14 Capture/compare register 2 (TIMx\_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:16	CCR2	rw	0x0000	High Capture/Compare 2 value (higher 16 bits)
15:0	CCR2	rw	0x0000	<p>Low Capture/Compare 2 value (lower 16 bits)</p> <p>If channel CC2 is configured as output:</p> <p>CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).</p> <p>It is loaded immediately if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.</p> <p>If channel CC2 is configured as input:</p> <p>CCR2 is the counter value transferred by the last input capture 2 event (IC2).</p>

#### 17.4.15 Capture/compare register 3 (TIMx\_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:16	CCR3	rw	0x0000	High Capture/Compare 3 value (higher 16 bits)
15:0	CCR3	rw	0x0000	<p>Low Capture/Compare 3 value (lower 16 bits)</p> <p>If channel CC3 is configured as output:</p> <p>CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).</p> <p>It is loaded immediately if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output. If channel CC3 is configured as input:</p> <p>CCR3 is the counter value transferred by the last input capture 3 event (IC3).</p>

### 17.4.16 Capture/compare register 4 (TIMx\_CCR4)

Address offset: 0x40  
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:16	CCR4	rw	0x0000	High Capture/Compare 4 value (higher 16 bits)
15:0	CCR4	rw	0x0000	If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded immediately if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC4 output. If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4).

### 17.4.17 DMA control register (TIMx\_DCR)

Address offset: 0x48  
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.			DBL				Res.			DBA					
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15:13	Reserved			Reserved, always read as 0

Bit	Field	Type	Reset	Description
12:8	DBL	w	0x00	<p>DMA burst length</p> <p>This bit-field defines the length of DMA burst transfer (the timer recognizes a burst transfer when a write access is done to the TIMx_DMAR register), ie. the number of transfers by half-words (double bytes) or bytes:</p> <ul style="list-style-type: none"> <li>00000: 1 transfer</li> <li>00001: 2 transfers</li> <li>00010: 3 transfers...</li> <li>...10001: 18 transfers</li> </ul> <p>Example: Let us consider the following transfer: DBL = 7, DBA = TIM2_CR1</p> <ul style="list-style-type: none"> <li>- In this case, DBL = 7 and DBA = TIM2_CR1 represent the transfer is done starting from the address given in the following formula:  <math>(\text{TIMx\_CR1 address}) + \text{DBA} + (\text{DMA index})</math>, where DMA index = DBL,</li> </ul> <p><math>(\text{TIMx\_CR1 address}) + \text{DBA} + 7</math> provides the address to/from which the transfer is done, ie. the seventh register starting from the <math>(\text{TIMx\_CR1 address}) + \text{DBA}</math>. The following situations may occur according to the configuration of DMA data length:</p> <ul style="list-style-type: none"> <li>- If the data is configured in half-words (16 bits), then the data is transferred to all of seven registers.</li> <li>- If the data is configured in bytes, the data is still transferred to all of seven registers: the first register contains the first MSB byte, the second contains the first LSB byte, and so on. Therefore, users must specify the data length in the DMA transfer as for timers.</li> </ul>
7:5	Reserved			Reserved and always read as 0.
4:0	DBA	w	0x00	<p>DMA base address</p> <p>This bit-field defines the base-address for DMA burst transfer (when a write access is done through the TIMx_DMAR address). DMA is defined as an offset starting from the address of the TIMx_CR1 register.</p> <ul style="list-style-type: none"> <li>00000: TIMx_CR1</li> <li>00001: TIMx_CR2</li> <li>00010: TIMx_SMCR</li> <li>.....</li> </ul>

#### 17.4.18 DMA address for full transfer (TIMx\_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	

Bit	Field	Type	Reset	Description
15:0	DMAB	w	0x0000	<p>DMA register for burst accesses</p> <p>A write operation to the TIMx_DMAR register accesses the register located at the address:</p> <p>'TIMx_CR1 address' + DBA + DMA index, where 'TIMx_CR1 address' is the address of the control register 1 (TIMx_CR1),</p> <p>'DBA' is the base address configured in TIMx_DCR register,</p> <p>'DMA index' is the offset automatically controlled by DMA,</p> <p>and depends on the DBL configured in TIMx_DCR.</p>

# 18

# Basic Timer (TIM14)

## Basic Timer (TIM14)

### 18.1 TIM14 introduction

The basic timer TIM14 consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The basic timer TIM14 is completely independent, and does not share any resources.

### 18.2 TIM14 main features

- 16-bit auto-reload counter
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge-aligned Mode)
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

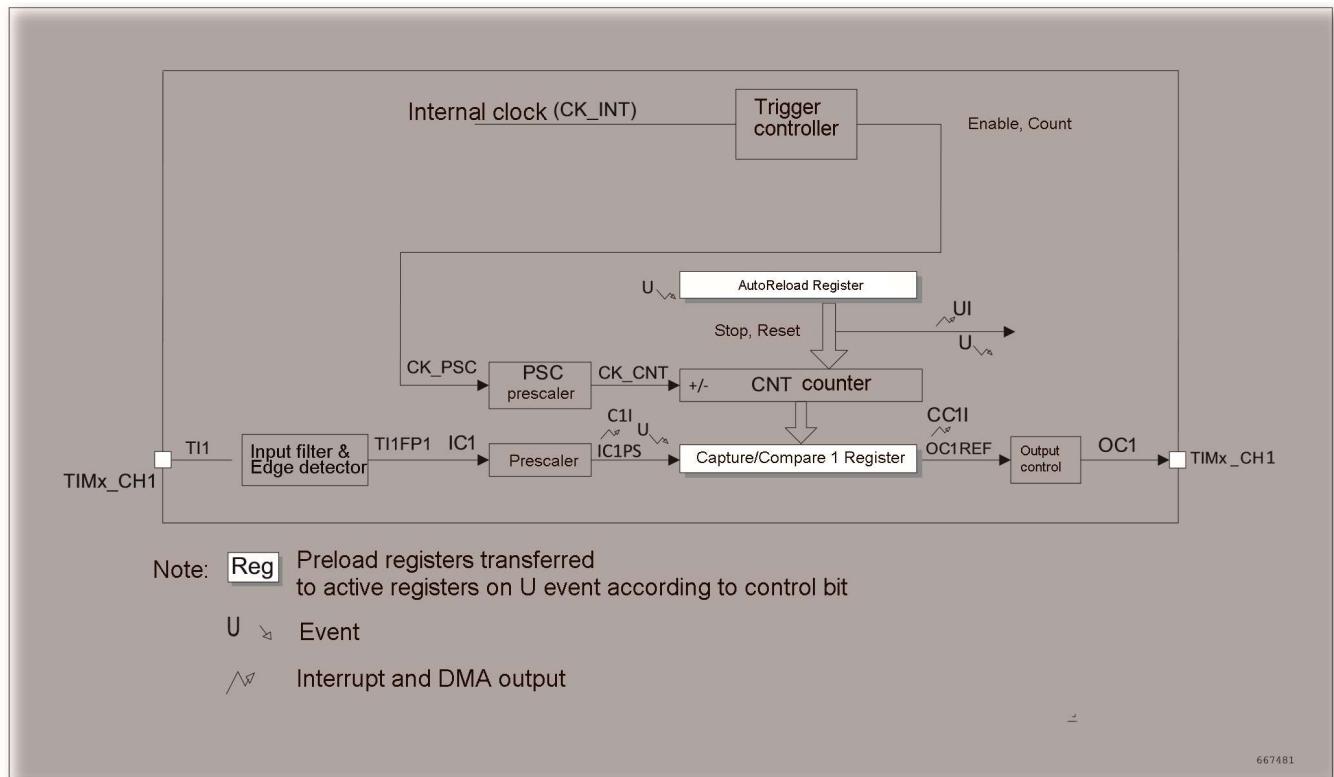


Figure 182. Basic timer block diagram

## 18.3 TIM14 functional description

### 18.3.1 Time-base unit

The main block of the programmable basic timer is a 16-bit counter with its related auto-reload register. The counter can count up. The counter clock can be divided by a prescaler. The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running. The time-base unit includes:

- Counter register (TIM14\_CNT)
- Prescaler register (TIM14\_PSC)
- Auto-reload register (TIM14\_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register immediately or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIM14\_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIM14\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIM14\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT\_EN is set 1 clock cycle after CEN.

### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM14\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

The figures below give some examples of the counter behavior when the prescaler ratio is changed on the fly.

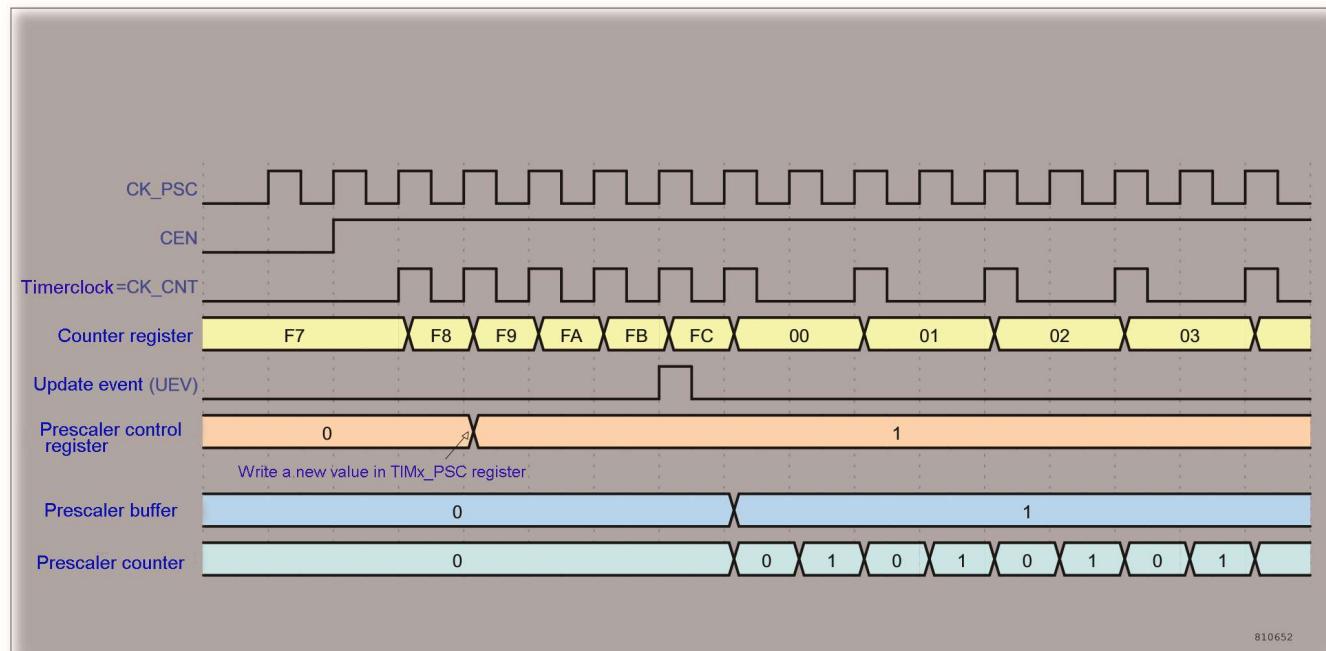


Figure 183. Counter timing diagram with prescaler division change from 1 to 2

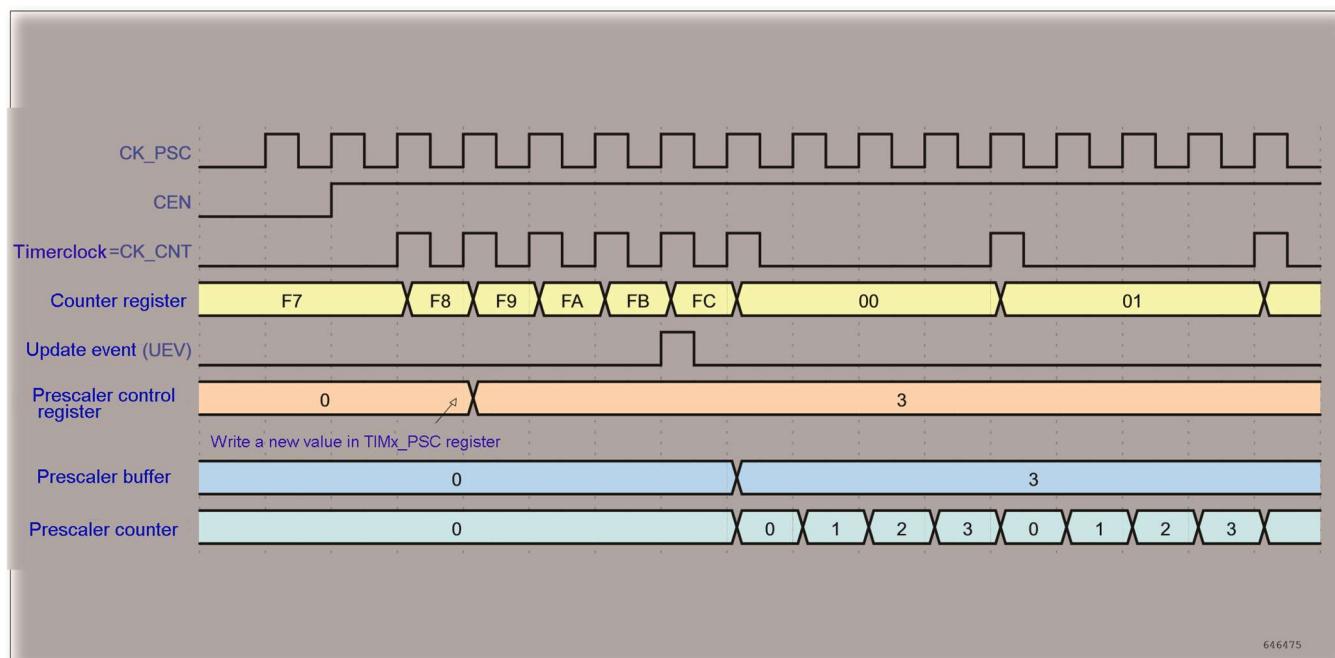


Figure 184. Counter timing diagram with prescaler division change from 1 to 4

### 18.3.2 Counting mode

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIM14\_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated by setting the UG bit in the TIM14\_EGR register.

The UEV event can be disabled by setting the UDIS bit in the TIM14\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change) when an update event should be generated. In addition, if the URS bit (update request selection) in TIM14\_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set by hardware (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter in the capture mode.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM14\_SR register) is set by hardware (depending on the URS bit).

- The auto-reload shadow register is updated with the preload value (TIM14\_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIM14\_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM14\_ARR = 0x36:

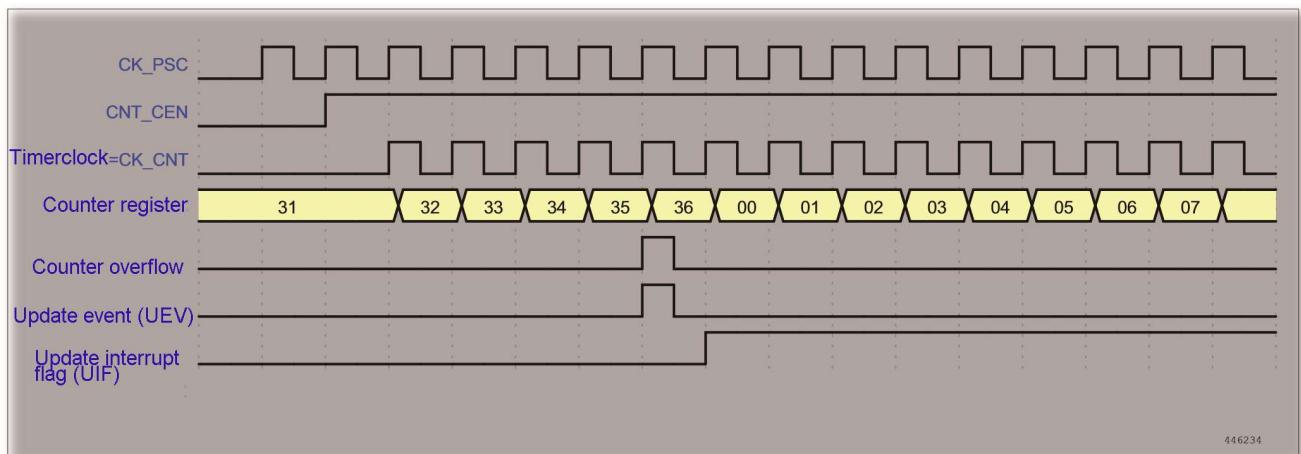


Figure 185. Counter timing diagram, internal clock divided by 1

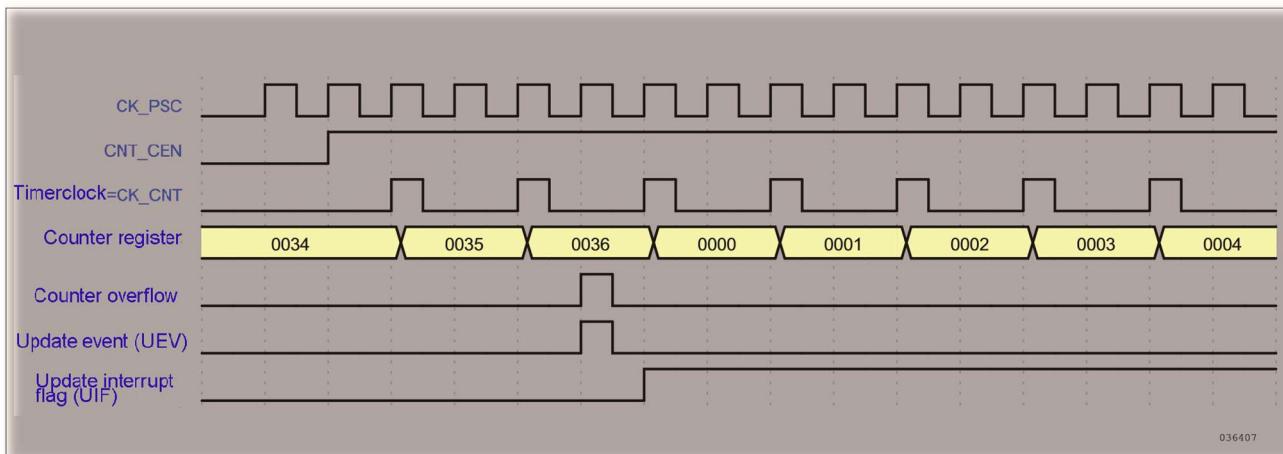


Figure 186. Counter timing diagram, internal clock divided by 2

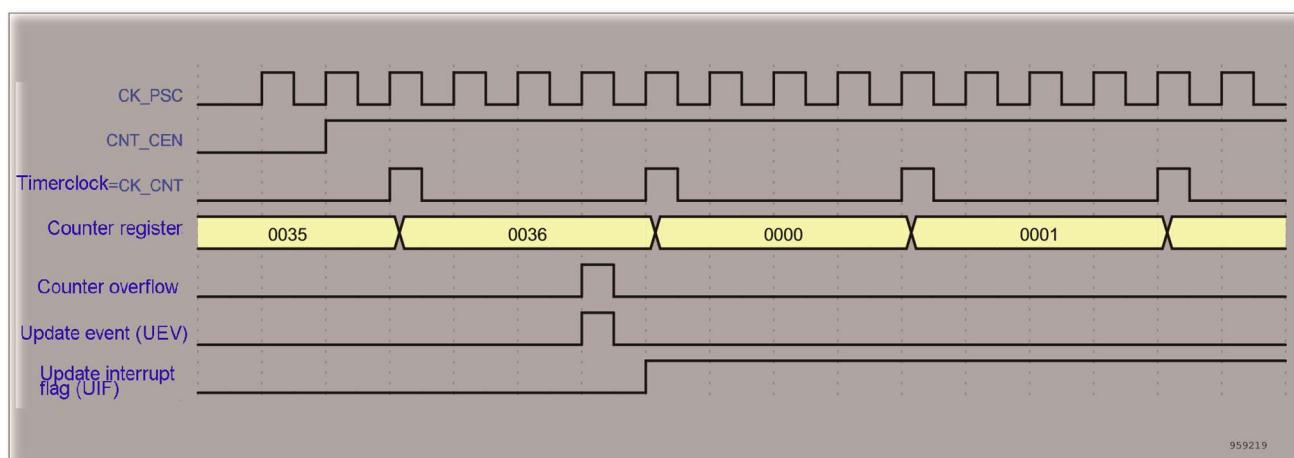


Figure 187. Counter timing diagram, internal clock divided by 4

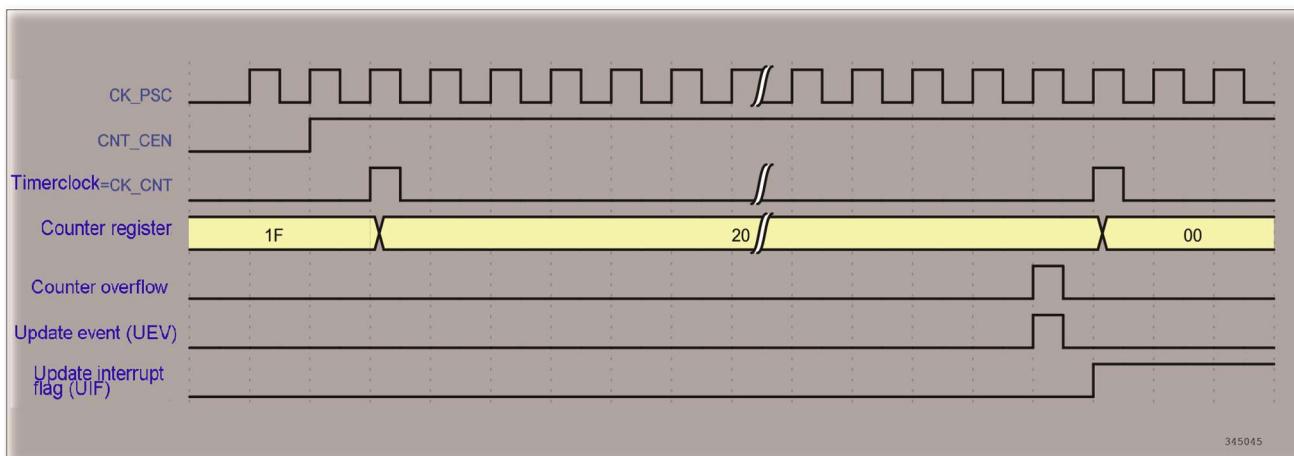


Figure 188. Counter timing diagram, internal clock divided by N

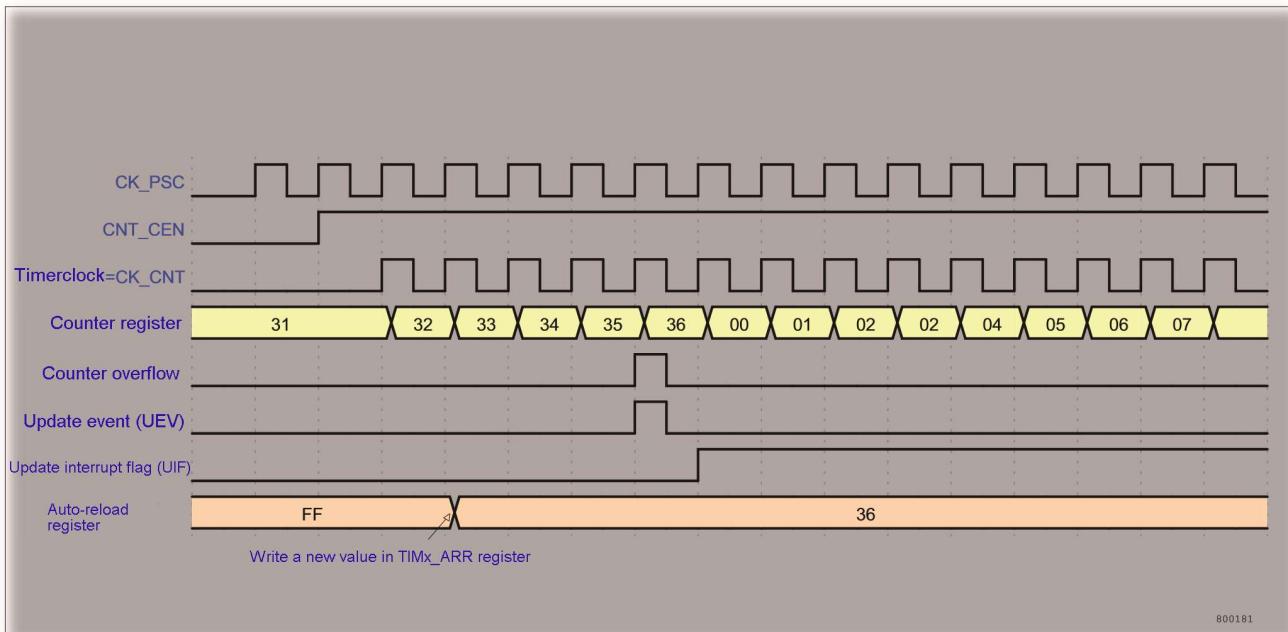


Figure 189. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)

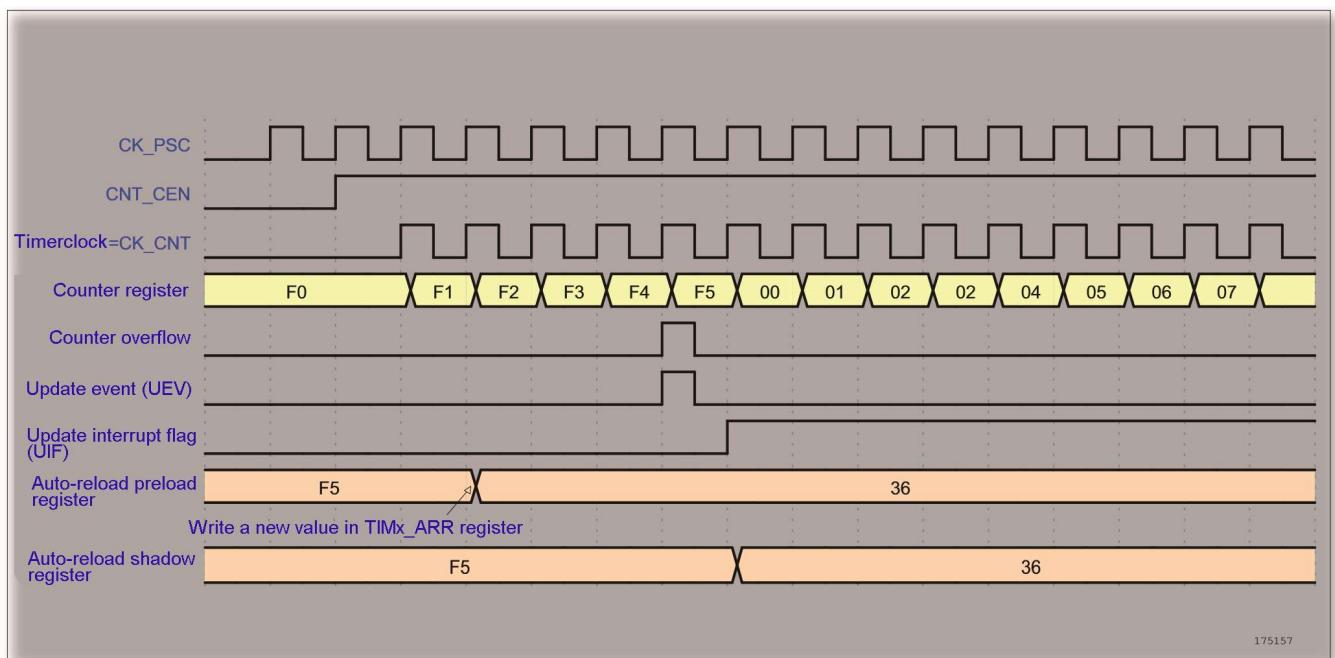


Figure 190. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)

### 18.3.3 Clock source

The counter clock is provided by the Internal clock (CK\_INT) source.

The CEN (in the TIM14\_CR1 register) and UG bits (in the TIM14\_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to '1', the prescaler is clocked by the internal clock CK\_INT.

The figure below shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

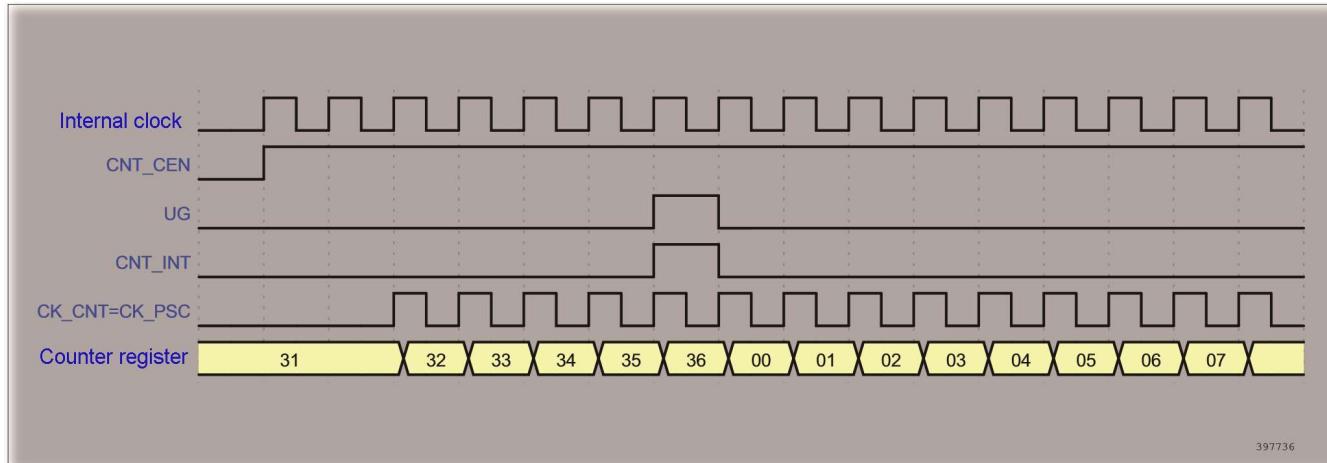


Figure 191. Control circuit in normal mode, internal clock divided by 1

### 18.3.4 Capture/compare channel

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figures give an overview of one Capture/Compare channel. The input stage samples the corresponding TI<sub>x</sub> input to generate a filtered signal TI<sub>x</sub>F. Then, an edge detector with polarity selection generates a signal (TI<sub>x</sub>FP<sub>x</sub>) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (IC<sub>x</sub>PS).

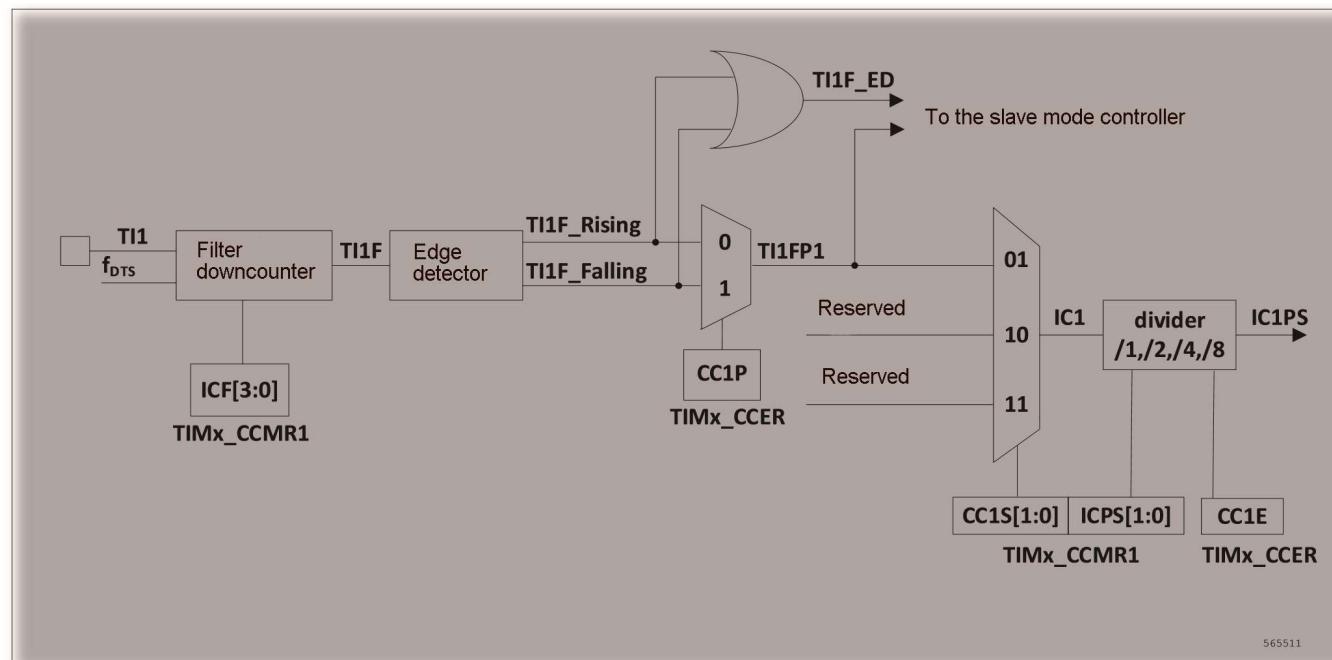


Figure 192. Capture/compare channel (example: channel 1 input stage)

The output stage generates an intermediate waveform which is then used for reference: OC<sub>x</sub>REF (active high). The polarity acts at the end of the chain.

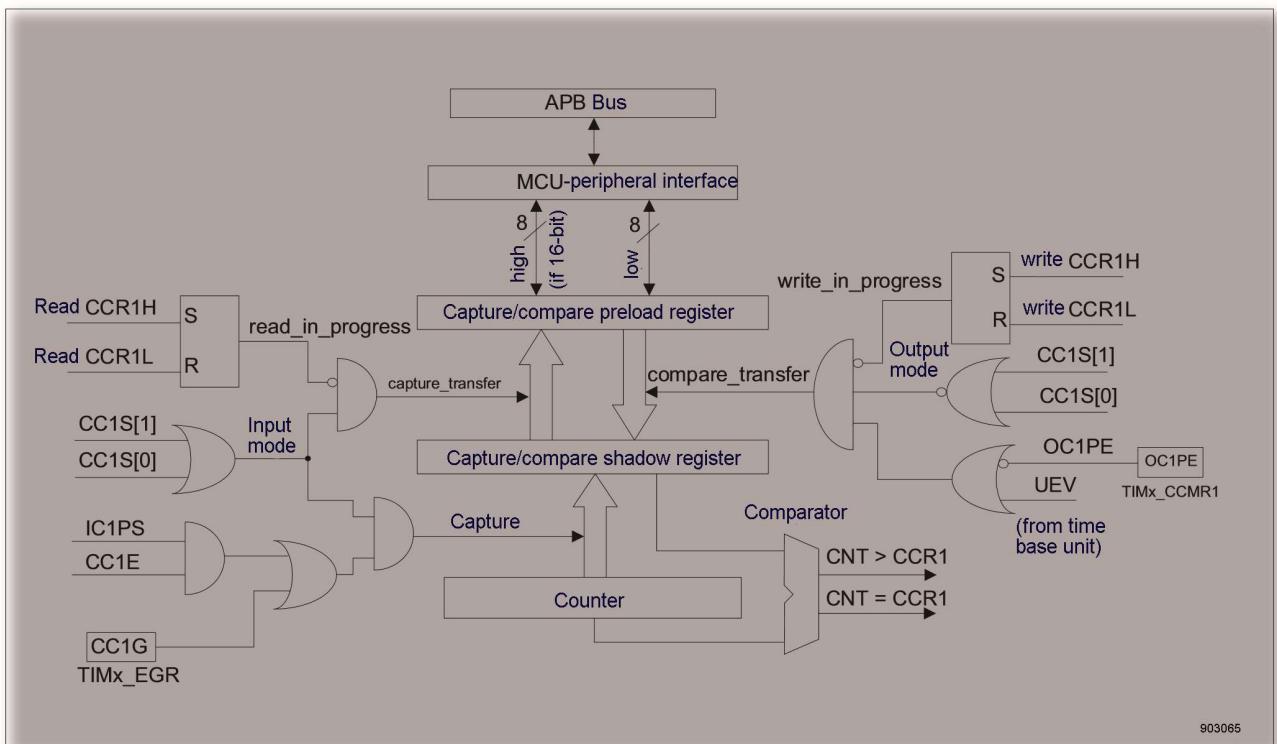


Figure 193. Capture/compare channel 1 main circuit

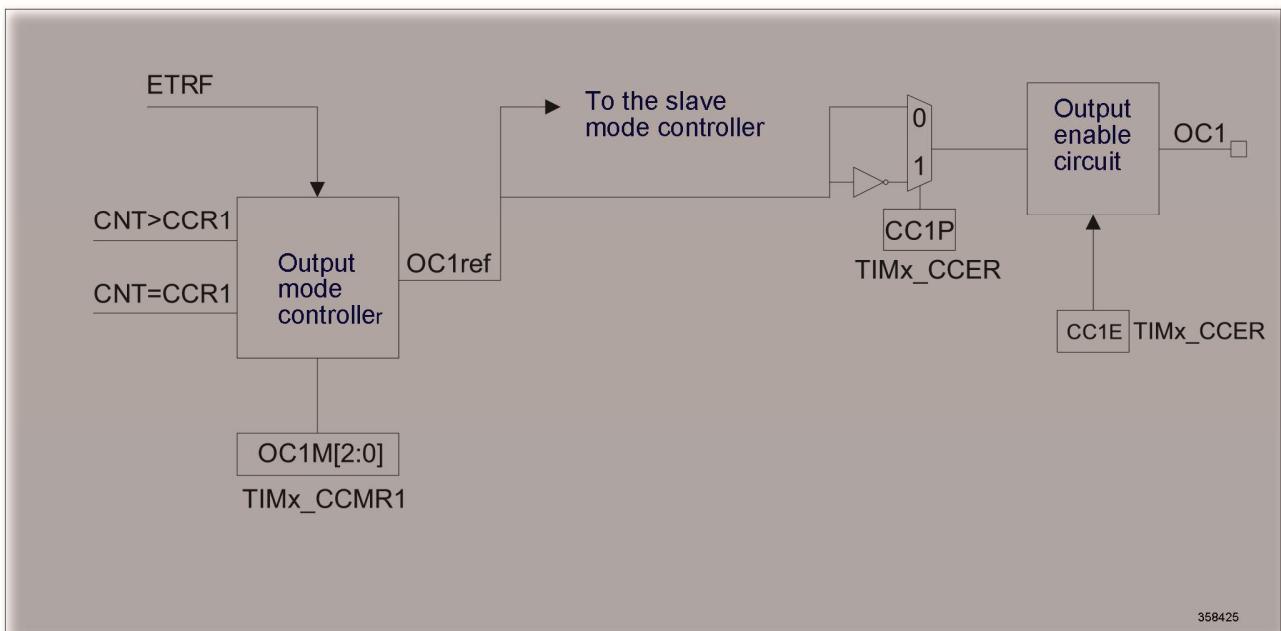


Figure 194. Output stage of capture/compare channel (channel 1)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register. In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 18.3.5 Input capture mode

In Input capture mode, the Capture/Compare registers (TIM14\_CCRx) are used to latch the value of the counter after an edge is detected by the corresponding IC<sub>x</sub> signal. When a capture occurs, the corresponding CC<sub>x</sub>IF flag (TIM14\_SR register) is set and an interrupt can be sent if it is enabled. If a capture occurs while the CC<sub>x</sub>IF flag was already high, then the over-capture flag CC<sub>x</sub>OF (TIM14\_SR register) is set. CC<sub>x</sub>IF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIM14\_CCRx register. CC<sub>x</sub>OF is cleared when written to '0'. The following example shows how to capture the counter value in TIM14\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIM14\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIM14\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIM14\_CCR1 register becomes read-only.
  2. Program the needed input filter duration with respect to the input signal (by programming ICxF bits in the TIM14\_CCMRx register if the input is a TI<sub>x</sub> input). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate an edge transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f<sub>DTS</sub> frequency). Then write IC1F bits to 0011 in the TIM14\_CCMR1 register.
  3. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIM14\_CCER register
  4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIM14\_CCMR1 register).
  5. Enable capture from the counter into the capture register by setting the CC1E bit to '1' in the TIM14\_CCER register.
  6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM14\_DIER register.
- When an input capture occurs:

- The TIM14\_CCR1 register gets the value of the counter on the active level transition. CC1IF flag is set (interrupt flag). CC1OF is also set to '1' if at least two consecutive captures occurred whereas the CC1IF flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: Input capture interrupt requests can be generated by software by setting the corresponding CC<sub>x</sub>G bit in the TIM14\_EGR register.

### 18.3.6 Forced output mode

In output mode (CC<sub>x</sub>S bits = 00 in the TIM14\_CCMRx register), each output compare signal (OC<sub>x</sub>REF and then OC<sub>x</sub>) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OC<sub>x</sub>REF/OC<sub>x</sub>) to its active level, the user just needs to write 101 in the OC<sub>x</sub>M bits in the corresponding TIM14\_CCMRx register. Thus OC<sub>x</sub>REF is forced high (OC<sub>x</sub>REF is always active high) and OC<sub>x</sub> gets opposite value to CC<sub>x</sub>P polarity bit. For example: CC<sub>x</sub>P = 0 (OC<sub>x</sub> active high) => OC<sub>x</sub> is forced to high level.

The OC<sub>x</sub>REF signal can be forced low by writing the OC<sub>x</sub>M bit to 100 in the TIM14\_CCMRx register. Anyway, the comparison between the TIM14\_CCRx shadow register and the counter is still performed

and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

### 18.3.7 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bit in the TIM14\_CCMRx register) and the output polarity (CCxP bit in the TIM14\_CCER register). The output pin can keep its level (OCxM = 000), be set active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIM14\_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIM14\_DIER register).

The TIM14\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM14\_CCMRx register. In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The synchronization resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

The output compare mode is configured as follows:

- Select the counter clock (internal, external, prescaler).
- Write the desired data in the TIM14\_ARR and TIM14\_CCRx registers.
- Set the CCxIE bit if an interrupt request is to be generated.
- Select the output mode,
  - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = 0 to disable preload
  - Write CCxP = 0 to select active high
  - Write CCxE = 1 to enable output
- Enable the counter by setting the CEN bit in the TIM14\_CR1 register.

The TIM14\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in the following figure.

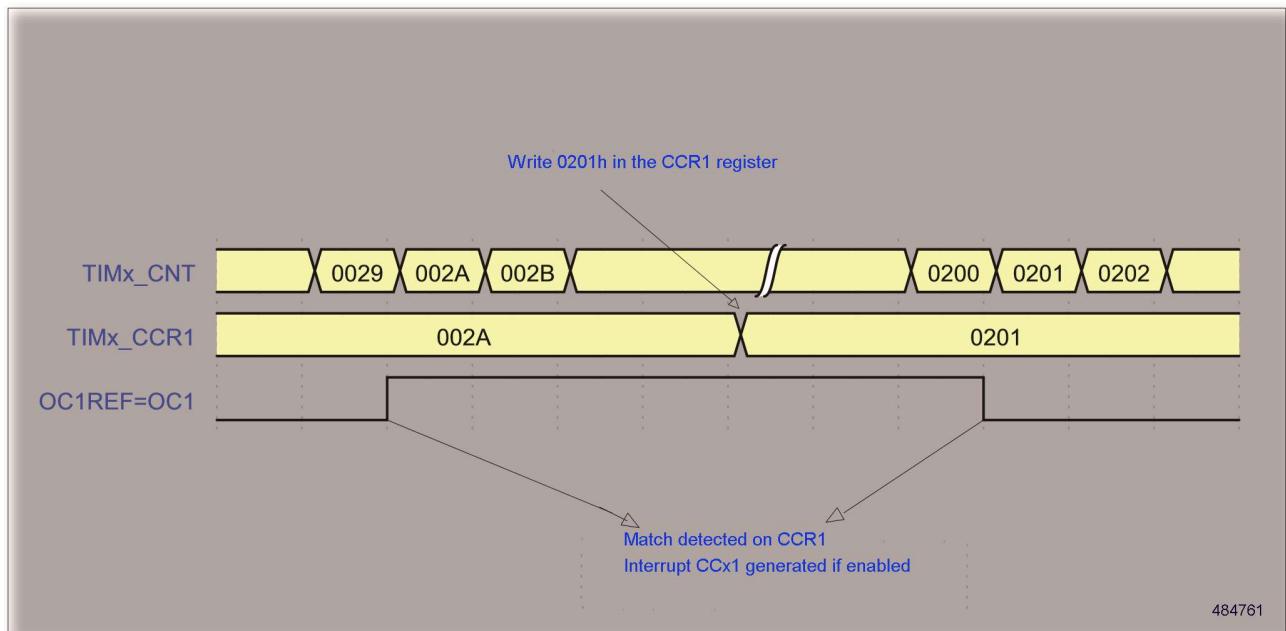


Figure 195. Output compare mode, toggle on OC1

### 18.3.8 PWM mode

Pulse Width Modulation mode allows generating a signal with a frequency determined by the value of the TIM14\_ARR register and a duty cycle determined by the value of the TIM14\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bit in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM14\_CCMRx register, and eventually the auto-reload preload register (in upcounting mode) by setting the ARPE bit in the TIM14\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user must initialize all the registers by setting the UG bit in the TIM14\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIM14\_CCER register. It can be programmed as active high or active low. OCx output is enabled by using the CCxE bit in the TIM14\_CCER register. For details, refer to the TIM14\_CCERx register description.

In PWM mode (1 or 2), TIM14\_CNT and TIM14\_CCRx are always compared to determine whether  $\text{TIM14\_CNT} \leq \text{TIM14\_CCRx}$ .

The reason is that this counter is counting up and can only generate PWM in edge-aligned mode.

### PWM edge-aligned mode

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as  $\text{TIM14\_CNT} < \text{TIM14\_CCRx}$ ; else it becomes low. If the compare value in TIM14\_CCRx is greater than the auto-reload value (in TIM14\_ARR), then OCxREF is held at '1'. If the compare value is 0 then OCxREF is held at '0'. The following figure shows some edge-aligned PWM waveforms in an example where TIM14\_ARR = 8.

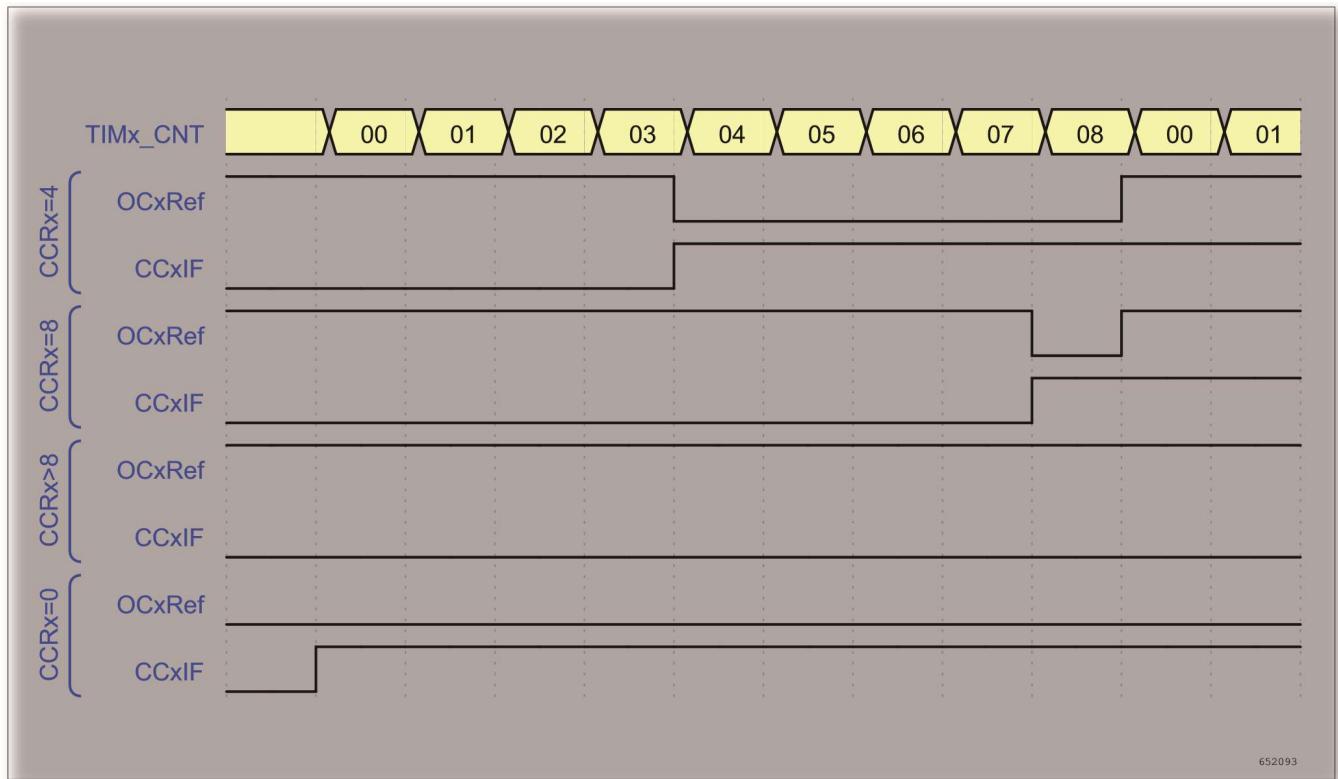


Figure 196. Edge-aligned PWM waveforms (ARR = 8)

### 18.3.9 Debug mode

When the microcontroller enters debug mode (Cortex®-M0 core halted), the TIM14 counter either continues to work normally or stops, depending on DBG\_TIMx\_STOP configuration bit in DBG module.

## 18.4 TIM14 register description

Table 56. Overview of TIM14 registers

Offset	Acronym	Register Name	Reset	Section
0x00	TIM14_CR1	Control register 1	0x00000000	Section 18.4.1
0x0C	TIM14_DIER	Interrupt enable register	0x00000000	Section 18.4.2
0x10	TIM14_SR	Status register	0x00000000	Section 18.4.3
0x14	TIM14_EGR	Event generate register	0x00000000	Section 18.4.4
0x18	TIM14_CCMR1	Capture/compare mode register 1	0x00000000	Section 18.4.5
0x20	TIM14_CCER	Capture/compare enable register	0x00000000	Section 18.4.6
0x24	TIM14_CNT	Counter	0x00000000	Section 18.4.7
0x28	TIM14_PSC	Prescaler	0x00000000	Section 18.4.8
0x2C	TIM14_ARR	Auto-reload register	0x00000000	Section 18.4.9

0x34	TIM14_CCR1	Capture/compare register 1	0x00000000	Section 18.4.10
0x44	TIM14_BDTR	Brake and dead-time register	0x00000000	Section 18.4.11

### 18.4.1 Control register 1 (TIM14\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CKD	ARPE	Reserved		URS	UDIS	CEN							
		rw	rw			rw	rw								

Bit	Field	Type	Reset	Description
15:10	Reserved			Reserved, read as 0
9:8	CKD	rw	0x00	<p>Clock division</p> <p>These 2 bits indicate the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling frequency used by the dead-time generators and the digital filters (ETR, TIx).</p> <p>00: <math>t_{DTS} = t_{CK\_INT}</math></p> <p>01: <math>t_{DTS} = 2 \times t_{CK\_INT}</math></p> <p>10: <math>t_{DTS} = 4 \times t_{CK\_INT}</math></p> <p>11: Reserved, do not program this value</p>
Bit	Field	Type	Reset	Description
7	ARPE	rw	0x00	<p>Auto-reload preload enable</p> <p>0: TIM14_ARR register is not buffered</p> <p>1: TIM14_ARR register is buffered</p>
6:3	Reserved			Reserved and read as 0.
2	URS	rw	0x00	<p>Update request source</p> <p>This bit is set by software to select the UEV event sources.</p> <p>0: The Update (UEV) event is generated by one of the following events if UEV is enabled:</p> <ul style="list-style-type: none"> <li>- Counter overflow</li> <li>- Setting the UG bit</li> <li>- Update generation through the slave mode controller</li> </ul> <p>1: Only counter overflow generates an update interrupt if UEV is enabled.</p>

1	UDIS	rw	0x00	Update disable This bit is set by software to enable/disable UEV event generation. 0: UEV enabled. The Update (UEV) event is generated by one of the following events: - Counter overflow - Setting the UG bit Shadow registers are updated subsequently. 1: UEV disabled. The Update event is not generated, shadow registers (ARR, PSC, CCRx) keep their values. However the counter and the prescaler are reinitialized if the UG bit is set.
0	CEN	rw	0x00	Counter enable 0: Counter disabled 1: Counter enabled

#### 18.4.2 Interrupt enable register (TIM14\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
														rw	rw

Bit	Field	Type	Reset	Description
15: 2	Reserved			Reserved, read as 0
1	CC1IE	rw	0x00	Capture/Compare 1 interrupt enable 0: CC1 interrupt disabled 1: CC1 interrupt enabled
Bit	Field	Type	Reset	Description
0	UIE	rw	0x00	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

#### 18.4.3 Status register (TIM14\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CC1OF	Reserved							CC1IF
							rc_w0								rc_w0 rc_w0

Bit	Field	Type	Reset	Description
15:10	Reserved			Reserved, read as 0

9	CC1OF	rc_w0	0x00	<p>Capture/Compare 1 overcapture flag</p> <p>This flag is set to '1' by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.</p> <p>0: No overcapture has been detected.</p> <p>1: The counter value has been captured in TIM14_CCR1 register while CC1IF flag was already set to 1.</p>
8: 2	Reserved		Reserved and read as 0.	
1	CC1IF	rc_w0	0x00	<p>Capture/Compare 1 interrupt flag</p> <p>If channel CC1 is configured as output:</p> <p>This flag is set by hardware when the counter matches the compare value. It is cleared by software.</p> <p>0: No match.</p> <p>1: The content of the counter TIM14_CNT matches the content of the TIM14_CCR1 register.</p> <p>When the contents of TIM14_CCR1 are greater than the contents of TIM14_ARR, the CC1IF bit goes high on the counter overflow.</p> <p>If channel CC1 is configured as input:</p> <p>This bit is set to '1' by hardware on a capture. It is cleared by software or by reading the TIM14_CCR1 register.</p> <p>0: No input capture occurred</p> <p>1: The counter value has been captured in (copied to) TIM14_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)</p>

Bit	Field	Type	Reset	Description
0	UIF	rc_w0	0x00	<p>Update interrupt flag</p> <p>This bit is set by hardware on an update event. It is cleared by software.</p> <p>0: No update occurred.</p> <p>1: Update interrupt pending. This bit is set by hardware when the registers are updated:</p> <ul style="list-style-type: none"> <li>- At overflow and if the UDIS = 0 in the TIM14_CR1 register;</li> <li>- When CNT is reinitialized by software using the UG bit in TIM14_EGR register, if URS=0 and UDIS=0 in the TIM14_CR1 register.</li> </ul>

#### 18.4.4 Event generation register (TIM14\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1G	UG
w														w	w

Bit	Field	Type	Reset	Description
15:2	Reserved			Reserved, read as 0
1	CC1G	w	0x00	<p>Capture/Compare 1 generation</p> <p>This bit is set by software in order to generate a capture/compare event, it is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: A capture/compare event is generated on channel CC1:</p> <ul style="list-style-type: none"> <li>If channel CC1 is configured as output: CC1IF flag is set, corresponding interrupt is sent if enabled.</li> <li>If channel CC1 is configured as input: The current value of the counter is captured in TIM14_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled.</li> <li>The CC1OF flag is set if the CC1IF flag was already high.</li> </ul>
0	UG	w	0x00	<p>Update generation</p> <p>This bit can be set by software, it is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: Reinitialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.</p>

#### 18.4.5 Capture/compare mode register 1 (TIM14\_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So the user must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							Res.		OC1M		OC1PE	OC1FE	CC1S		
							IC1F							IC1PSC	

rw rw rw rw rw rw rw

**Output Compare mode:**

Bit	Field	Type	Reset	Description
15: 7	Reserved			Reserved, read as 0

Bit	Field	Type	Reset	Description

6:4	OC1M	rw	0x00	<p>Output compare 1 mode</p> <p>These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.</p> <p>000: Frozen. The comparison between the output compare register TIM14_CCR1 and the counter TIMx_CNT has no effect on the OC1REF.</p> <p>001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIM14_CNT matches the capture/compare register 1 (TIM14_CCR1).</p> <p>010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIM14_CNT matches the capture/compare register 1 (TIM14_CCR1).</p> <p>011: Toggle. OC1REF level toggles when TIM14_CCR1=TIM14_CNT.</p> <p>100: Force inactive level. OC1REF is forced low.</p> <p>101: Force active level. OC1REF is forced high.</p> <p>110: PWM mode 1 - channel 1 is active as long as TIM14_CNT &lt; TIM14_CCR1, else inactive.</p> <p>111: PWM mode 2 - channel 1 is inactive as long as TIM14_CNT &lt; TIM14_CCR1, else active.</p> <p>Note: In PWM mode 1 or 2, the OC1REF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.</p>
-----	------	----	------	---

3	OC1PE	rw	0x00	<p>Output compare 1 preload enable</p> <p>0: Preload function of TIM14_CCR1 register disabled. TIMx_CCR1 register can be written at anytime, the new value is taken in account immediately.</p> <p>1: Preload function of TIM14_CCR1 register enabled. Read/Write operations access the preload register. TIM14_CCR1 preload value is loaded in the active register at each update event.</p> <p>Note: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIM14_CR1 register). Else the behavior is not guaranteed.</p>
---	-------	----	------	--

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

2	OC1FE	rw	0x00	<p>Output compare 1 fast enable</p> <p>This bit is used to accelerate the effect of an event on the trigger input on the CC output.</p> <p>0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.</p> <p>1: An active edge on the trigger input acts like a compare match. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.</p>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC1 channel is configured as output</li> <li>01: CC1 channel is configured as input, IC1 is mapped on TI1</li> <li>10: reserved</li> <li>11: reserved</li> </ul> <p>Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM14_CCER).</p>

### Input Capture mode:

Bit	Field	Type	Reset	Description
15:8	Reserved			Reserved, read as 0

Bit	Field	Type	Reset	Description

7:4	IC1F	rw	0x00	<p>Input capture 1 filter</p> <p>This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to generate a transition on the output:</p> <ul style="list-style-type: none"> <li>0000: No filter, sampling is done at <math>f_{DTS}</math></li> <li>1000: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 6</li> <li>0001: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 2</li> <li>1001: <math>f_{SAMPLING} = f_{DTS}/8</math>, N = 8</li> <li>0010: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 4</li> <li>1010: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 5</li> <li>0011: <math>f_{SAMPLING} = f_{CK\_INT}</math>, N = 8</li> <li>1011: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 6</li> <li>0100: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 6</li> <li>1100: <math>f_{SAMPLING} = f_{DTS}/16</math>, N = 8</li> <li>0101: <math>f_{SAMPLING} = f_{DTS}/2</math>, N = 8</li> <li>1101: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 5</li> <li>0110: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 6</li> <li>1110: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 6</li> <li>0111: <math>f_{SAMPLING} = f_{DTS}/4</math>, N = 8</li> <li>1111: <math>f_{SAMPLING} = f_{DTS}/32</math>, N = 8</li> </ul>
3:2	IC1PSC	rw	0x00	<p>Input capture 1 prescaler</p> <p>This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).</p> <p>The prescaler is reset as soon as CC1E = 0 (TIM14_CCER register).</p> <ul style="list-style-type: none"> <li>00: no prescaler, capture is done each time an edge is detected on the capture input</li> <li>01: capture is done once every 2 events</li> <li>10: capture is done once every 4 events</li> <li>11: capture is done once every 8 events</li> </ul>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC1 channel is configured as output</li> <li>01: CC1 channel is configured as input, IC1 is mapped on TI1</li> <li>10: reserved</li> <li>11: reserved</li> </ul> <p>Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM14_CCER).</p>

### 18.4.6 Capture/compare enable register (TIM14\_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1 NP	Res.	CC1P	CC1E
												rw	rw	rw	

Bit	Field	Type	Reset	Description
15:4	Reserved			Reserved, read as 0
3	CC1NP	rw	0x00	<p>Capture/Compare 1 complementary output polarity</p> <p>CC1 channel configured as output: CC1NP must be kept cleared (CC1NP = 0)</p> <p>CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define TI1FP1 polarity (refer to CC1P description).</p>
2	Reserved			Reserved and read as 0.
1	CC1P	rw	0x00	<p>Capture/Compare 1 output polarity</p> <p>CC1 channel configured as output:</p> <ul style="list-style-type: none"> <li>0: OC1 active high</li> <li>1: OC1 active low</li> </ul> <p>CC1 channel configured as input:</p> <p>CC1P bit selects TI1FP1 and TI2FP1 polarity for trigger or capture operations (whether IC1 or the inverted signal of IC1 is used).</p> <ul style="list-style-type: none"> <li>00: non-inverted/rising edge</li> <li>Capture is done on TIxFP1 rising edge (capture mode); TIxFP1 is not inverted.</li> <li>01: inverted/falling edge</li> <li>Capture is done on TIxFP1 falling edge (capture mode); TIxFP1 is inverted.</li> <li>10: reserved, do not use this configuration.</li> <li>11: non-inverted/both edges</li> </ul> <p>Capture is done on both TIxFP1 rising and falling edges (capture mode); TIxFP1 is not inverted.</p> <p>Note: These bits can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).</p>

Bit	Field	Type	Reset	Description
0	CC1E	rw	0x00	Capture/Compare 1 output enable CC1 channel configured as output: 0: Off - OC1 is not active. 1: On - OC1 signal is output on the corresponding output pin. CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the register TIM14_CCR1 or not. 0: capture disabled 1: capture enabled

Table 57. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (OCx = 0, OCx_EN = 0)
1	OCx = OCxREF + Polarity, OCx_EN = 1

Note: The states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and the GPIO registers.

#### 18.4.7 Counter (TIM14\_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

CNT																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description												
15:0	CNT	rw	0x0000	Counter value												

#### 18.4.8 Prescaler (TIM14\_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

rw																
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
15:0	PSC	rw	0x0000	<p>Prescaler value</p> <p>The counter clock frequency (CK_CNT) is equal to <math>f_{CK\_PSC}/(PSC + 1)</math>.</p> <p>PSC contains the value to be loaded in the active prescaler register at each update event.</p>

#### 18.4.9 Auto-reload register (TIM14\_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description
15:0	ARR	rw	0x0000	<p>Auto-reload value</p> <p>ARR is the value to be loaded in the actual auto-reload register.</p> <p>Refer to the section of Time-base unit for more details about ARR update and behavior.</p> <p>The counter is blocked while the auto-reload value is null.</p>

#### 18.4.10 Capture/compare register 1 (TIM14\_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description

15:0	CCR1	rw	0x0000	Capture/Compare 1 value If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded immediately if the preload feature is not selected in the TIM14_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIM14_CNT and signalled on OC1 output. If channel CC1 is configured as input: CCR1 is the counter value transferred by the last input capture 1 event (IC1).
------	------	----	--------	---

#### 18.4.11 Break and dead-time register (TIM14\_BDTR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE															Reserved

rw

Bit	Field	Type	Reset	Description
15	MOE	rw	0x00	MOE: Main output enable 0: outputs disabled. 1: outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM14_CCER register). See OC/OCN enable description for more details (Section 18.4.6: capture/compare enable register (TIM14_CCER)).
14:0	Reserved			Reserved and always read as 0.

# 19

# Basic Timer (TIM16/17)

## Basic Timer (TIM16/17)

### 19.1 TIM16/17 introduction

The basic timer TIM16/17 consists of a 16-bit auto-reload counter driven by a programmable prescaler. It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The basic timer TIM16/17 is completely independent, and does not share any resources.

### 19.2 Main features

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- 1 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Break input to put the timer’s output signals in reset state or in a known state
- Interrupt/DMA generation on the following events:
  - Update: Counter overflow
  - Input capture
  - Output compare
  - Break signal input

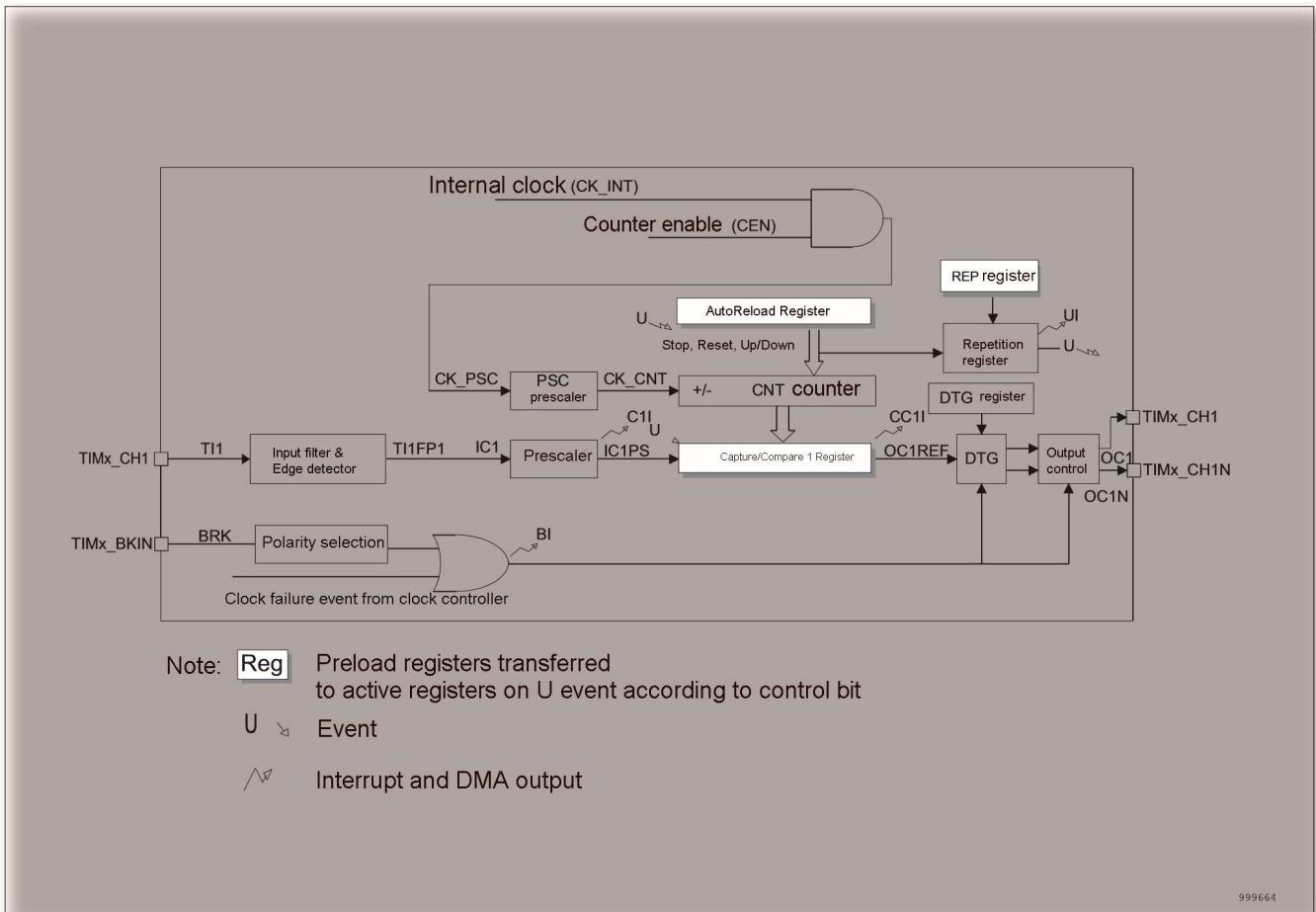


Figure 197. Basic timer TiM16 and TIM17 block diagram

## 19.3 Functional description

### 19.3.1 Time-base unit

The main block of the programmable basic timer is a 16-bit counter with its related auto-reload register. This counter is counting up. The counter clock can be divided by a prescaler. The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running. The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)
- Repetition counter register (TIMx\_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register immediately or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in the TIMx\_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal is set 1 clock cycle after CEN.

### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

The figures below give some examples of the counter behavior when the prescaler ratio is changed on the fly.

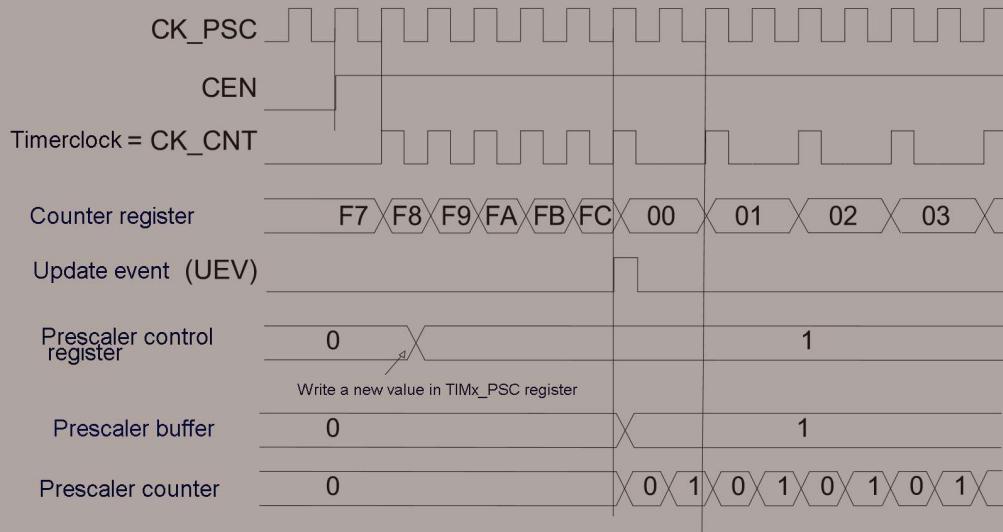


Figure 198. Counter timing diagram with prescaler division change from 1 to 2

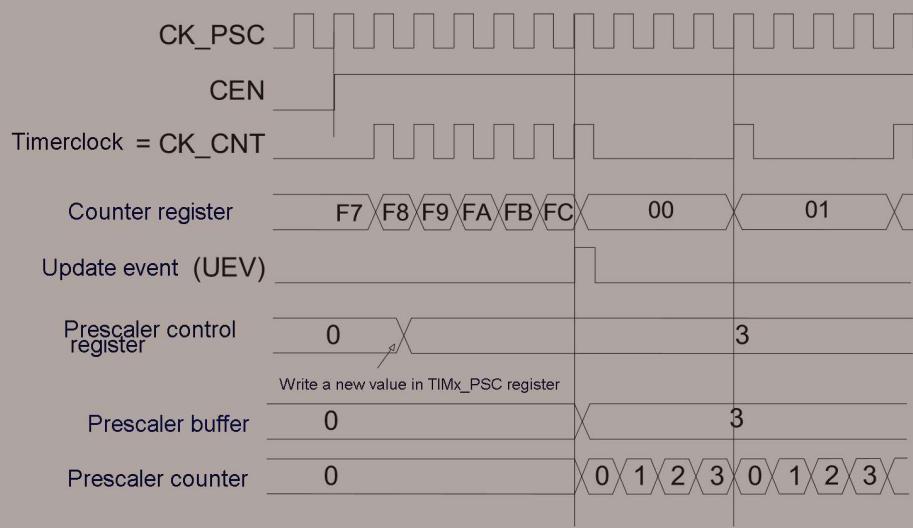


Figure 199. Counter timing diagram with prescaler division change from 1 to 4

### 19.3.2 Counting mode

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to '0'. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change) when an update event should be generated. In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set by hardware (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter in the capture mode. When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set by hardware (depending on the URS bit).

- The auto-reload shadow register is updated with the preload value (TIMx\_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR = 0x36.

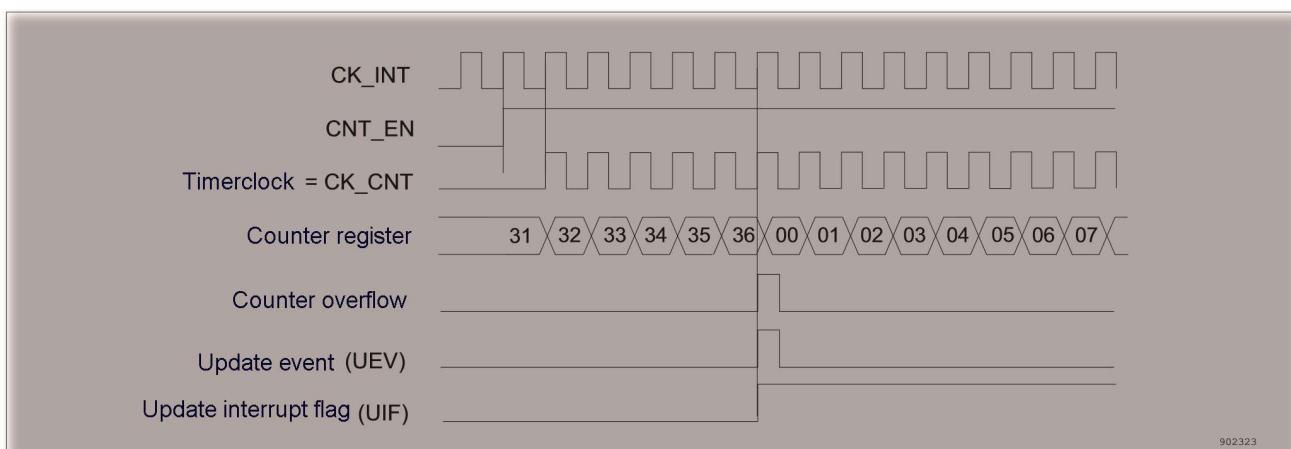


Figure 200. Counter timing diagram, internal clock divided by 1

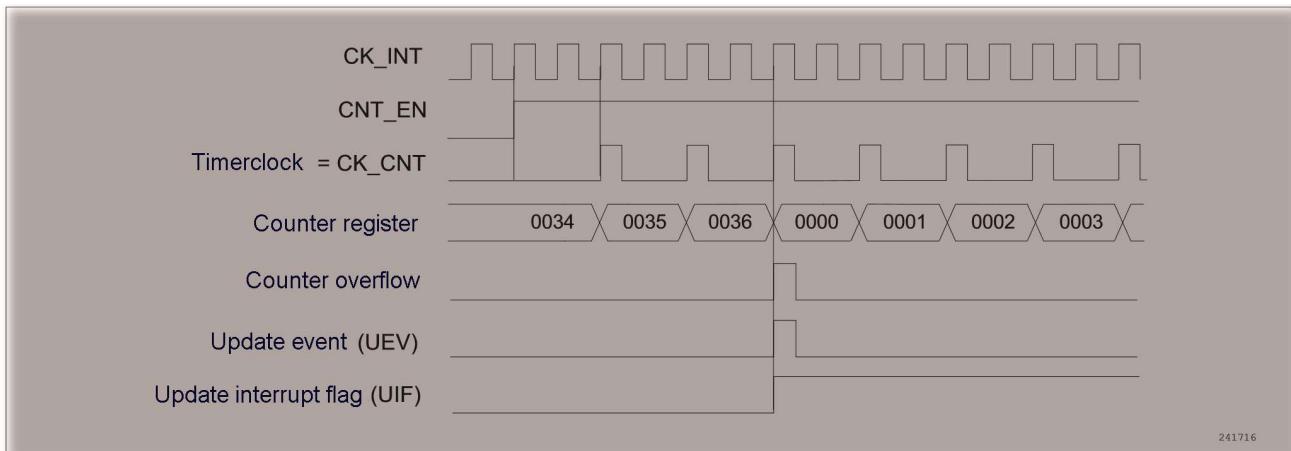


Figure 201. Counter timing diagram, internal clock divided by 2

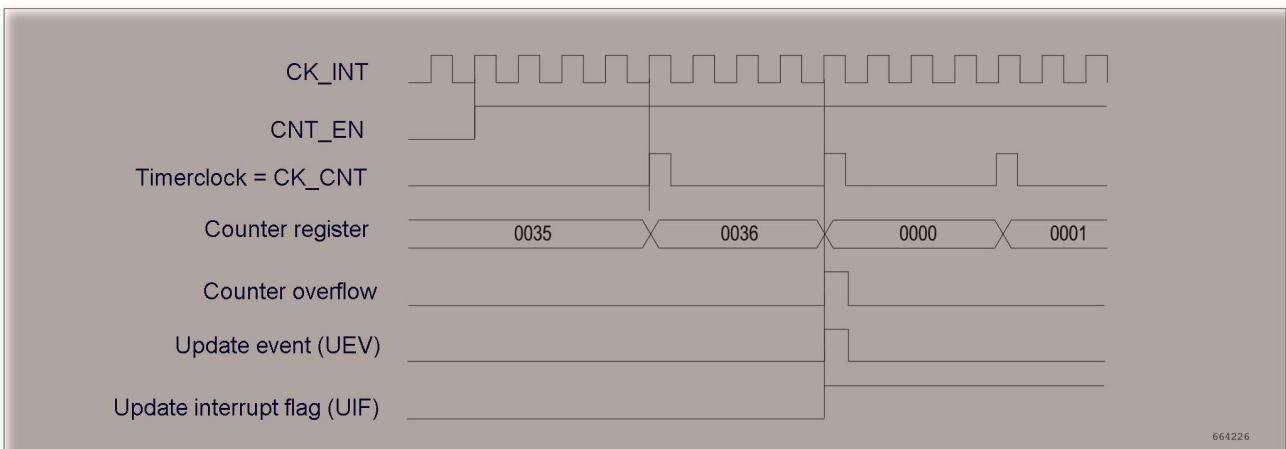


Figure 202. Counter timing diagram, internal clock divided by 4

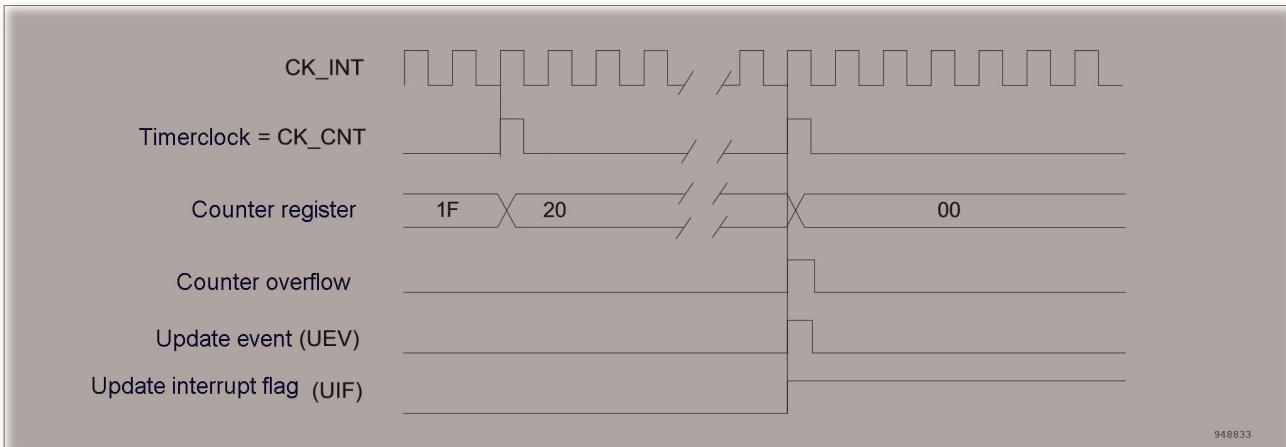


Figure 203. Counter timing diagram, internal clock divided by N

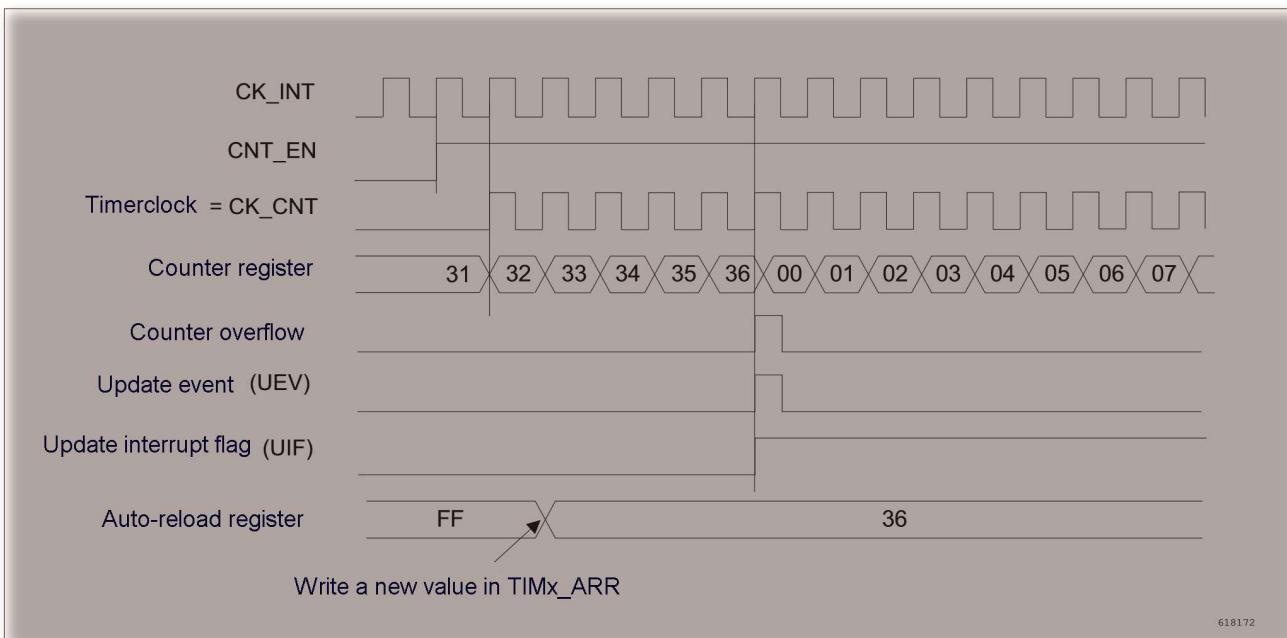


Figure 204. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)

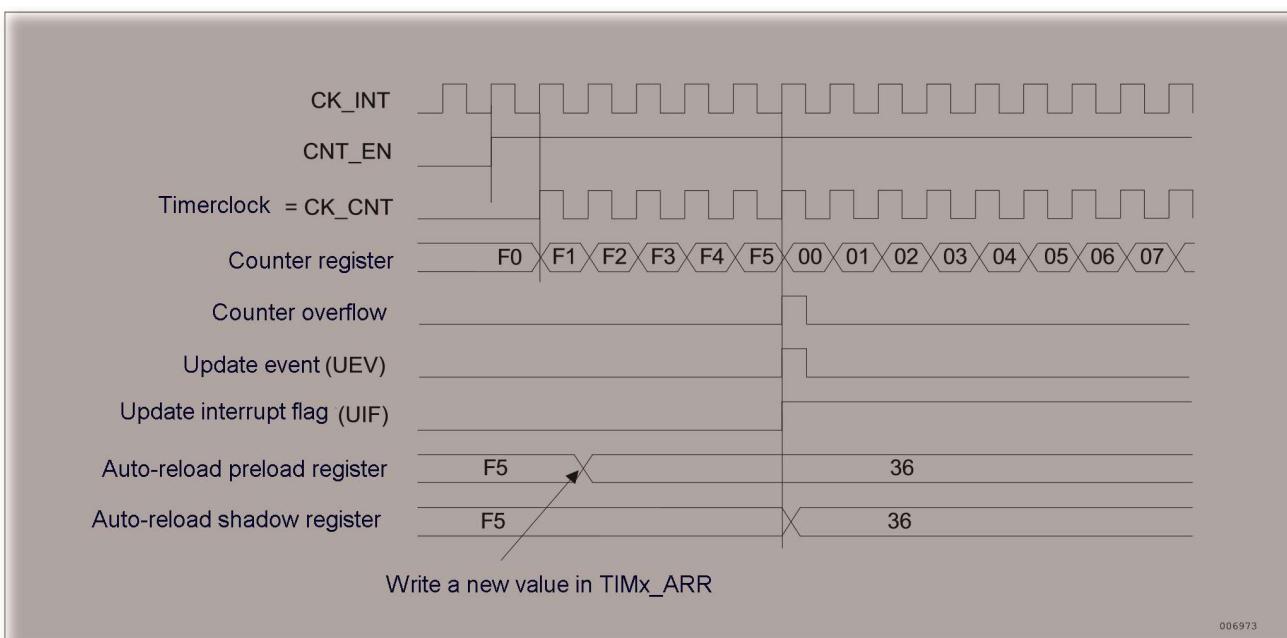
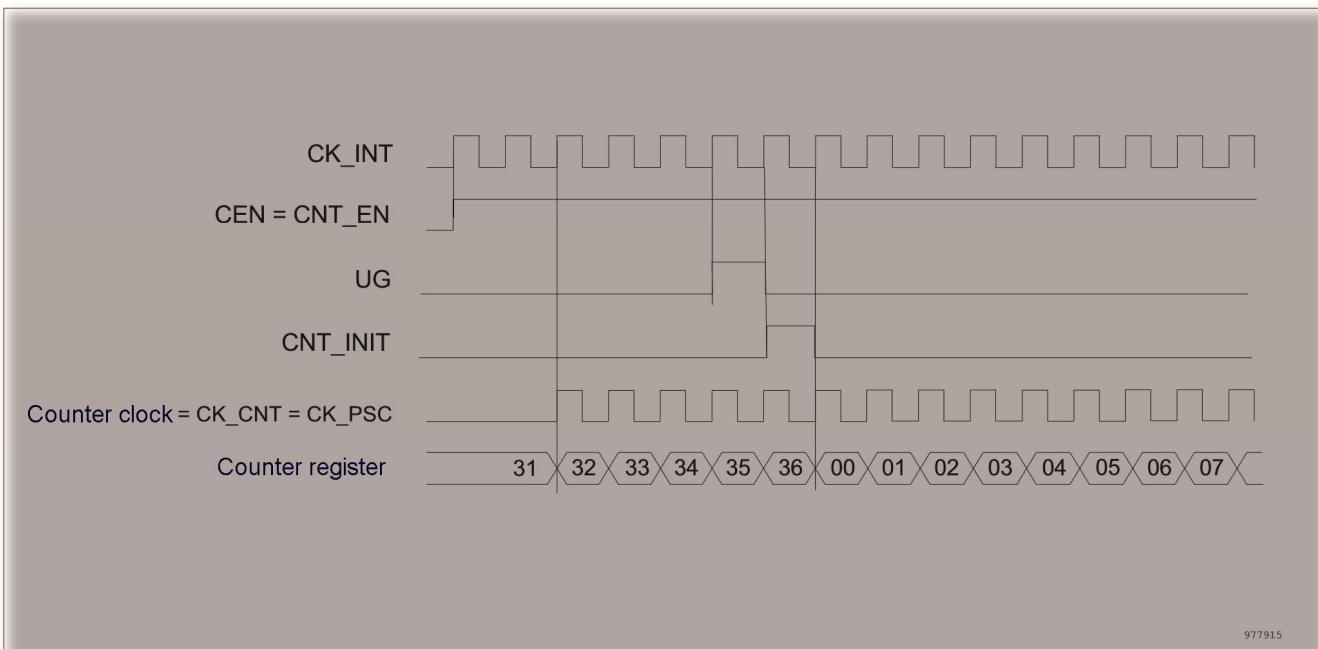


Figure 205. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)

### 19.3.3 Clock source

The counter clock is provided by the Internal clock (CK\_INT) source. The CEN (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to '1', the prescaler is clocked by the internal clock CK\_INT.

The figure below shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.



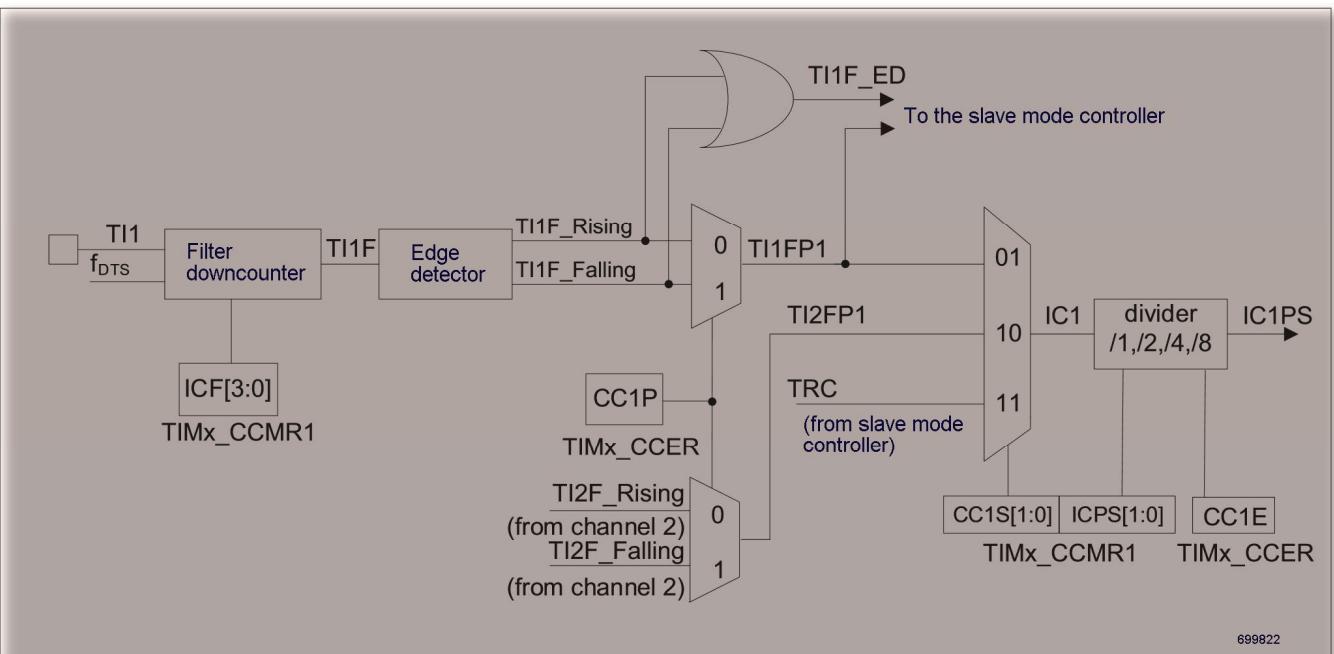
977915

Figure 206. Control circuit in normal mode, internal clock divided by 1

#### 19.3.4 Capture/compare channel

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control). The following figures give an overview of one Capture/Compare channel.

The input stage samples the corresponding TI<sub>x</sub> input to generate a filtered signal TI<sub>x</sub>F. Then, an edge detector with polarity selection generates a signal (TI<sub>x</sub>FP<sub>x</sub>) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (IC<sub>x</sub>PS).



699822

Figure 207. Capture/compare channel (example: channel 1 input stage)

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

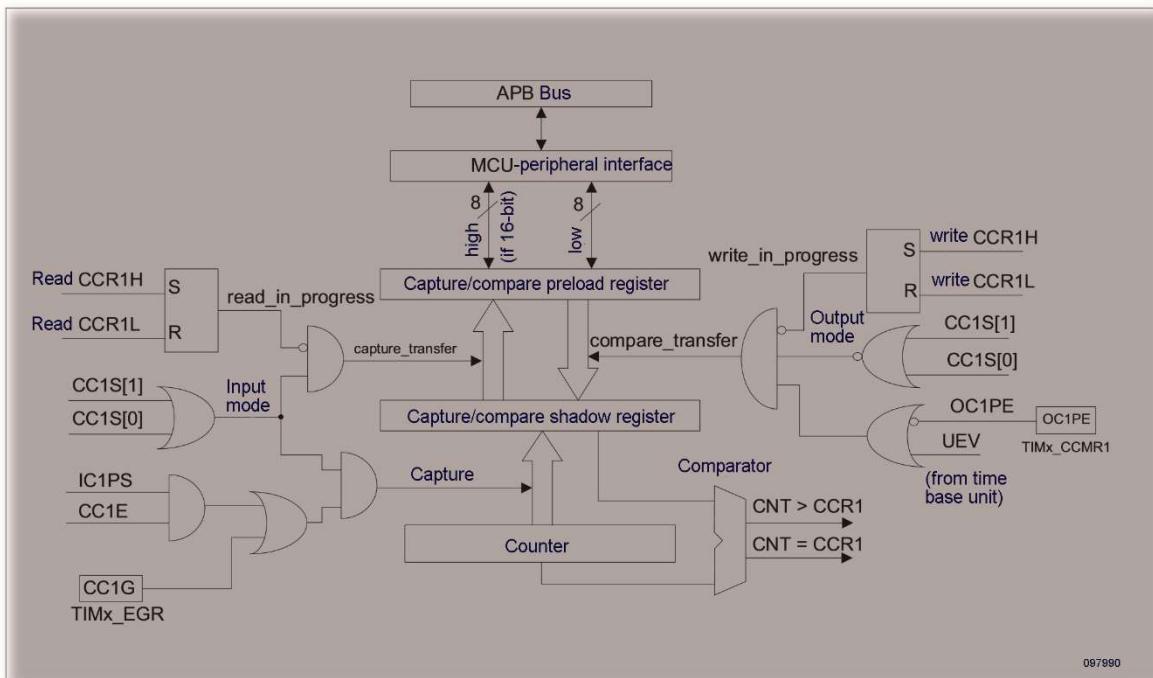


Figure 208. Capture/compare channel 1 main circuit

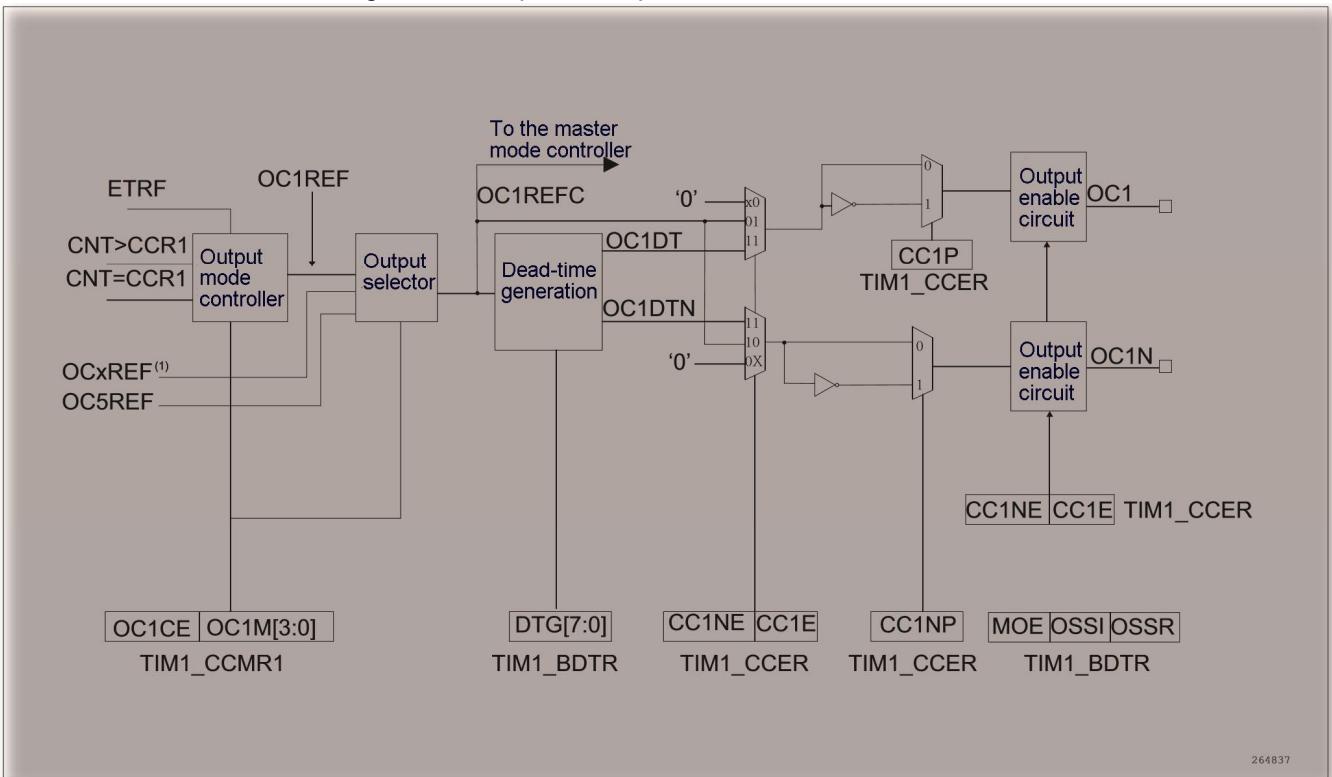


Figure 209. Output stage of capture/compare channel (channel 1)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register. In capture mode, captures are actually done in the shadow register, which is copied into the preload register. In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 19.3.5 Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx\_CCRx) are used to latch the value of the counter after an edge is detected by the corresponding IC<sub>x</sub> signal. When a capture occurs, the corresponding CC<sub>x</sub>IF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CC<sub>x</sub>IF flag was already high, then the over-capture flag CC<sub>x</sub>OF (TIMx\_SR register) is set. CC<sub>x</sub>IF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register.

CC<sub>x</sub>OF is cleared when written to '0'.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from '00', the channel is configured in input and the TM1\_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the input signal (by programming ICxF bits in the TIMx\_CCMRx register if the input is a TI<sub>x</sub> input). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate an edge transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f<sub>DTS</sub> frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx\_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit to '1' in the TIMx\_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active level transition.
- C1IF flag is set (interrupt flag). CC1OF is also set to '1' if at least two consecutive captures occurred whereas the CC1IF flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CC<sub>x</sub>G bit in the TIMx\_EGR register.

### 19.3.6 Forced output mode

In output mode (CC<sub>x</sub>S bits = 00 in the TIMx\_CCMRx register), each output compare signal (OC<sub>x</sub>REF and then OC<sub>x</sub>/OC<sub>x</sub>N) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCxREF is forced high (OCxREF is always active high) and OCx gets opposite value to CCxP polarity bit.

For example: CCxP = 0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bit to 100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 19.3.7 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bit in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM = 000), be set active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxE bit in the TIMx\_DIER register).
- Generates a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register. In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The synchronization resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

The output compare mode is configured as follows:

- Select the counter clock (internal, external, prescaler).
- Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
- Set the CCxE bit if an interrupt request is to be generated.
- Select the output mode :
  - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = 0 to disable preload register
  - Write CCxP = 0 to select active high polarity
  - Write CCxE = 1 to enable OCx output
- Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in the figure below.

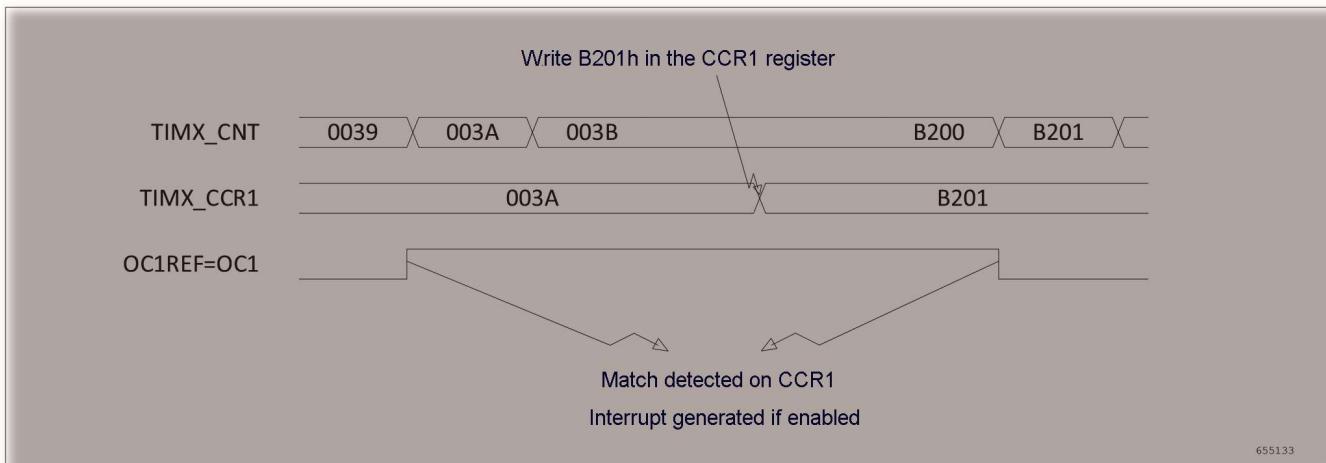


Figure 210. Output compare mode, toggle on OC1

### 19.3.8 PWM mode

Pulse Width Modulation mode allows generating a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bit in the TIMx\_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting mode) by setting the ARPE bit in the TIMx\_CR1 register. As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSS1 and OSSR bits in the TIMx\_CCER and TIMx\_BDTR registers. Refer to the TIMx\_CCERx register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether  $\text{TIMx\_CCR}_x \leq \text{TIMx\_CNT}$  or  $\text{TIMx\_CNT} \leq \text{TIMx\_CCR}_x$  (depending on the direction of the counter). This counter is counting up and can only generate PWM in edge-aligned mode. Refer to the upcounting mode.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as  $\text{TIMx\_CNT} < \text{TIMx\_CCR}_x$ ; else it becomes low. If the compare value in TIMx\_CCRx is greater than the auto-reload value (in TIMx\_ARR), then OCxREF is held at '1'. If the compare value is 0 then OCxREF is held at '0'. Figure 211 shows some edge-aligned PWM waveforms in an example where TIMx\_ARR = 8.

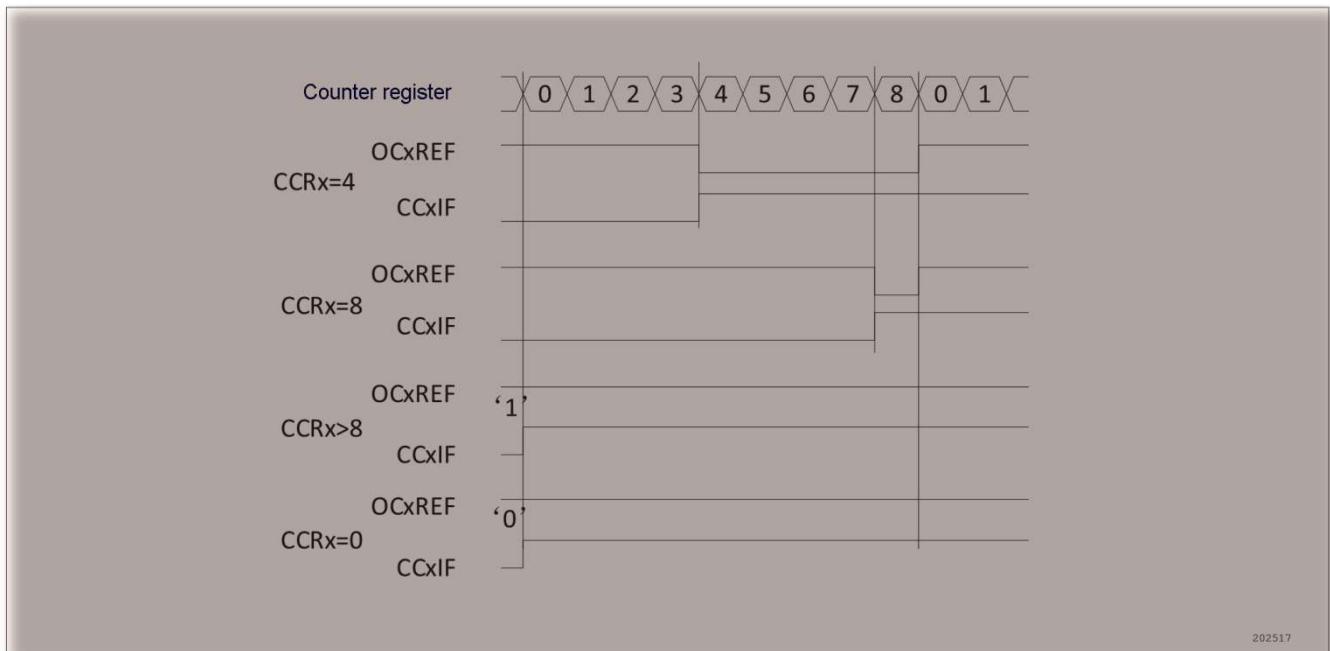


Figure 211. Edge-aligned PWM waveforms (ARR = 8)

### 19.3.9 Complementary outputs and dead-time insertion

The Basic timer TIM16/17 can output two complementary signals and manage the switching-off and the switching-on instants of the outputs. This time is generally known as dead-time and it has to be adjusted depending on the devices connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...).

User can select the polarity of the outputs (main output OC<sub>x</sub> or complementary OC<sub>xN</sub>) independently for each output. This is done by writing to the CC<sub>xP</sub> and CC<sub>xNP</sub> bits in the TIM<sub>x</sub>\_CCER register.

The complementary signals OC<sub>x</sub> and OC<sub>xN</sub> are activated by a combination of several control bits: the CC<sub>xE</sub> and CC<sub>xNE</sub> bits in the TIM<sub>x</sub>\_CCER register and the MOE, OIS<sub>x</sub>, OIS<sub>xN</sub>, OSSI and OSSR bits in the TIM<sub>x</sub>\_BDTR and TIM<sub>x</sub>\_CR2 registers. For more details, refer to Table 59: Output control bits for complementary OC<sub>x</sub> and OC<sub>xN</sub> channels with break feature. In particular, the dead-time is activated when switching to the IDLE state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CC<sub>xE</sub> and CC<sub>xNE</sub> bits, and the MOE bit if the break circuit is present. There is a 10-bit dead-time generator in each channel. From a reference waveform OC<sub>x</sub>REF, it generates 2 outputs OC<sub>x</sub> and OC<sub>xN</sub>. If OC<sub>x</sub> and OC<sub>xN</sub> are active high:

- The OC<sub>x</sub> output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OC<sub>xN</sub> output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OC<sub>x</sub> or OC<sub>xN</sub>) then the corresponding pulse is not generated. The following figures show the relationships between the output signals of the dead-time generator and the reference signal OC<sub>x</sub>REF (we suppose CC<sub>xP</sub> = 0, CC<sub>xNP</sub> = 0, MOE = 1, CC<sub>xE</sub> = 1 and CC<sub>xNE</sub> = 1 in these examples).

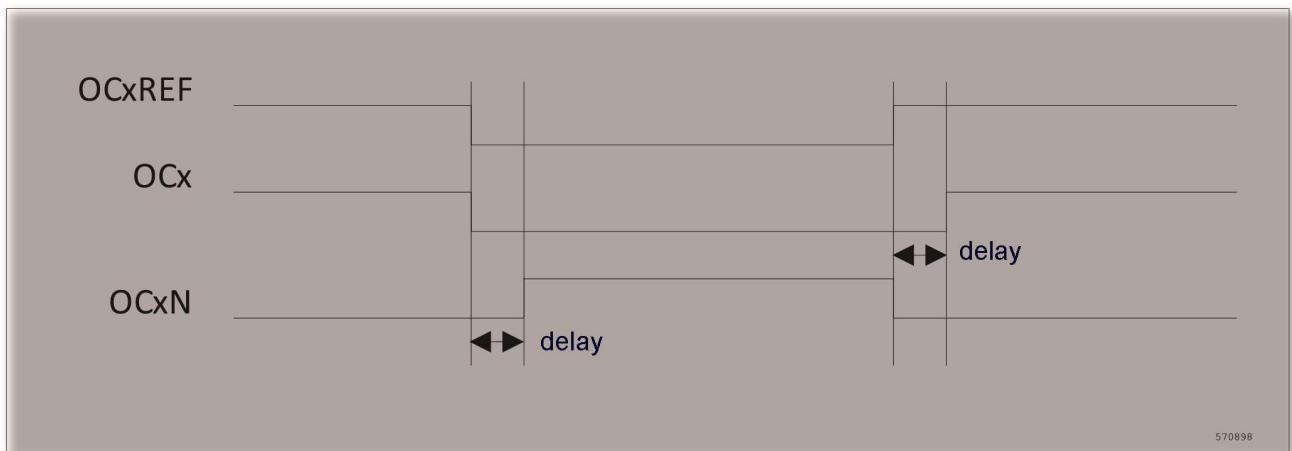


Figure 212. Complementary output with dead-time insertion

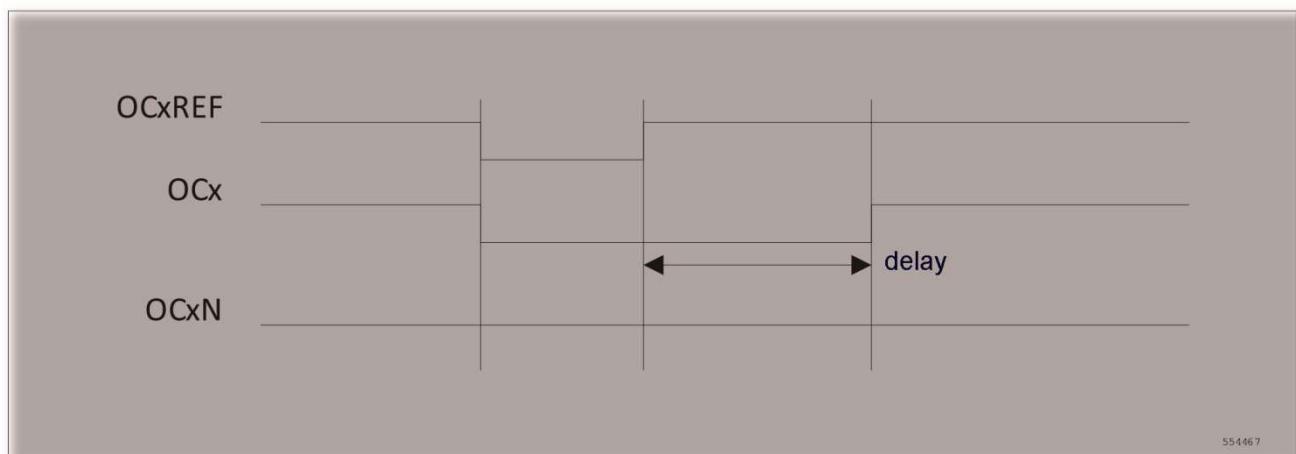


Figure 213. Dead-time waveforms with delay greater than the negative pulse

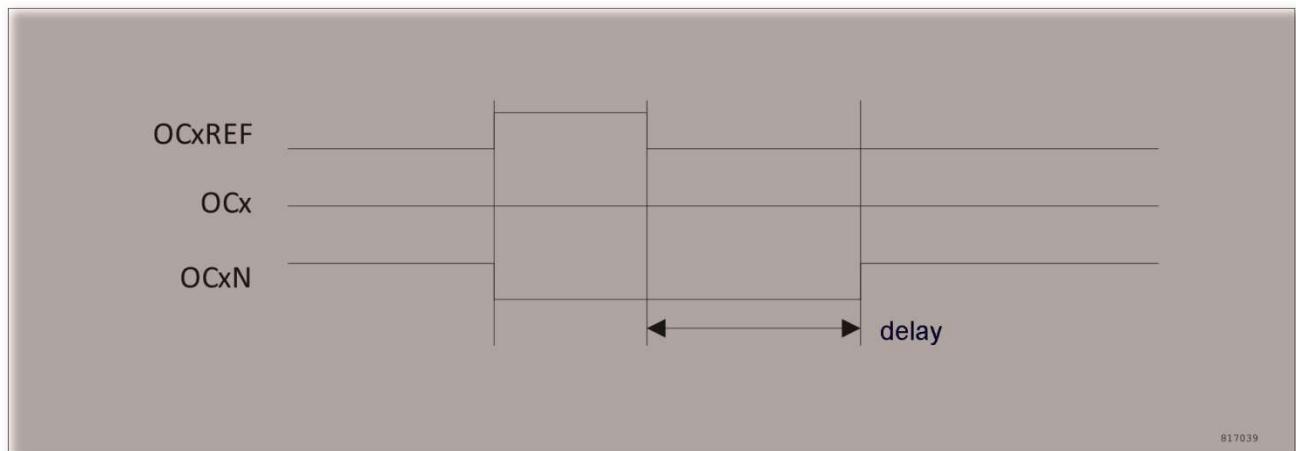


Figure 214. Dead-time waveforms with delay greater than the positive pulse

The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx\_BDTR register. Refer to the delay calculation in the Section 19.4.12 TIM16/17 brake and dead-time register (TIMx\_BDTR).

#### Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx\_CCER register.

This allows the user to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled ( $CCxE = 0$ ,  $CCxNE = 1$ ), it is not complemented and becomes active as soon as OCxREF is high. For example, if  $CCxNP = 0$  then  $OCxN = OCxREF$ . On the other hand, when both OCx and OCxN are enabled ( $CCxE = CCxNE = 1$ ), OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

### 19.3.10 Using the break function

When using the break function, the output enable signals and inactive levels are modified according to corresponding control bits (MOE, OSS1 and OSSR bits in the TIMx\_BDTR register, OISx and OISxN bits in the TIMx\_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to output control bits for complementary OCx and OCxN channels with break feature in the table 59. The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller. For details, refer to the Clock Security System section. When exiting from reset, the break circuit is disabled and the MOE bit is low. User can enable the break function by setting the BKE bit in the TIMx\_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx\_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is written to 1 whereas it was low, a delay (dummy instruction) must be inserted before reading it correctly. This is because the user writes an asynchronous signal, but reads a synchronous signal.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or in reset state (selected by the OSS1 bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx\_CR2 register as soon as MOE=0. If OSS1 = 0 then the timer releases the enable output else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck\_tim clock cycles).
  - If OSS1 = 0 then the timer releases the enable outputs, else the enable outputs remain; or the enable outputs become high as soon as one of the CCxE or CCxNE bits is high.
- An interrupt can be generated if the BIE bit in the TIMx\_DIER register is set and the break status flag (BIF bit in the TIMx\_SR register) is set to '1'. A DMA request can be generated if the BDE bit in the TIMx\_DIER register is set.
- If the AOE bit in the TIMx\_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx\_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows freezing the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The user can choose from three levels of protection selected by the LOCK bits in the TIMx\_BDTR register. Refer to Section 19.4.12: break and dead-time register (TIM16/17\_BDTR). The LOCK bits can be written only once after an MCU reset.

The figure below shows an example of behavior of the outputs in response to a break:

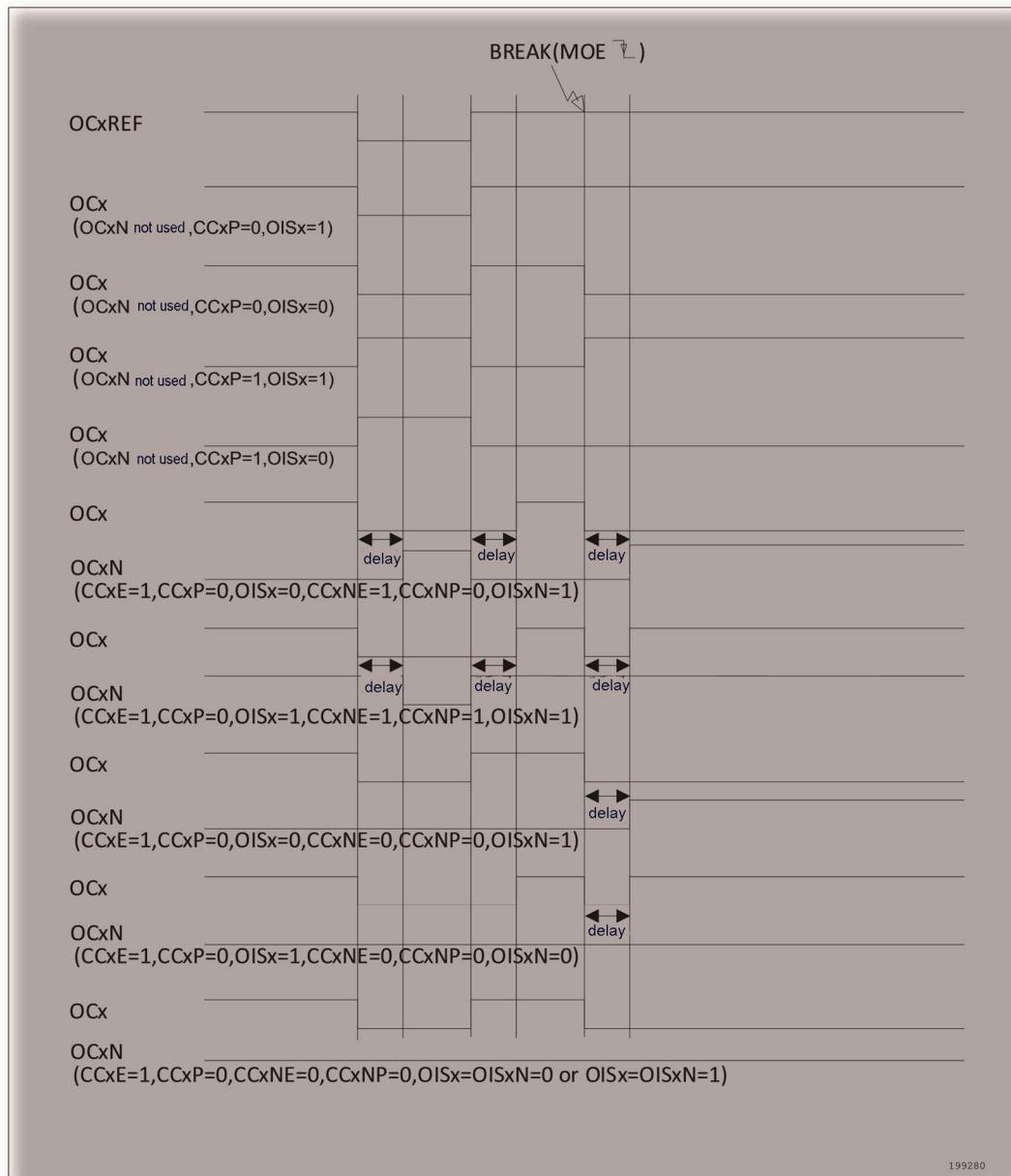


Figure 215. Output behavior in response to a break

### 19.3.11 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay. Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select One-pulse mode by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV. A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx    ARR (in particular, 0 < CCRx)

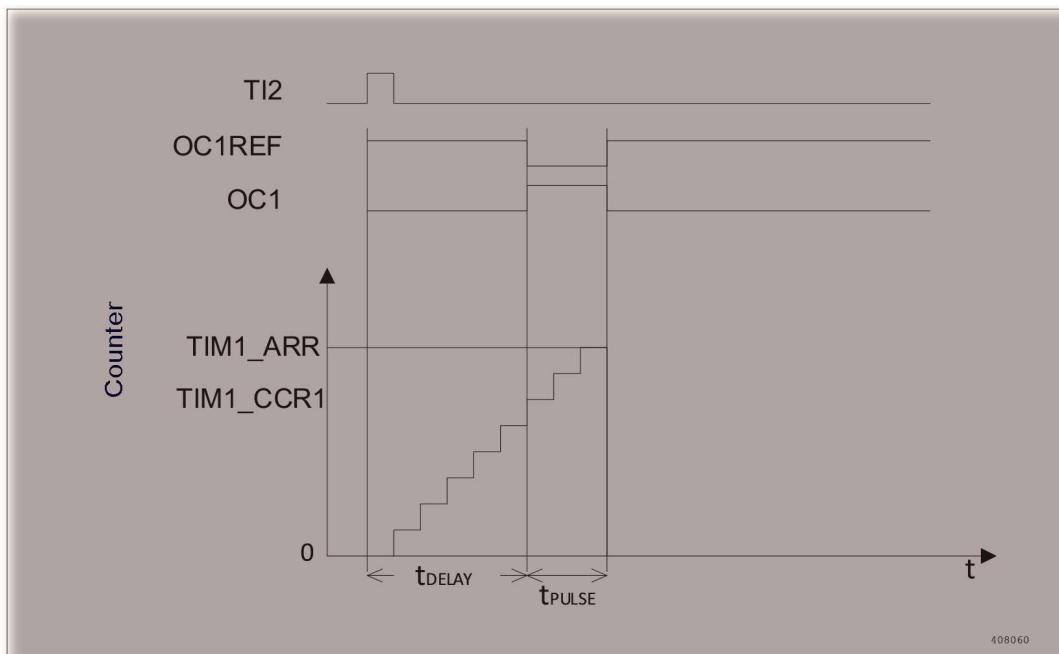


Figure 216. Example of one pulse mode

For example the user may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S to '01' in the TIMx\_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P to '0' in the TIMx\_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS to '110' in the TIMx\_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).
- The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).
- The  $t_{DELAY}$  is defined by the value written in the TIMx\_CCR1 register.

- The  $t_{PULSE}$  is defined by the difference between the auto-reload value and the compare value ( $\text{TIMx\_ARR} - \text{TIMx\_CCR1}$ ).
- Let us say the user wants to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this, enable PWM mode 2 by writing  $\text{OC1M} = 111$  in the  $\text{TIMx\_CCMR1}$  register. The user can optionally enable the preload registers by writing  $\text{OC1PE} = 1$  in the  $\text{TIMx\_CCMR1}$  register and  $\text{ARPE}$  in the  $\text{TIMx\_CR1}$  register. In this case the compare value must be written in the  $\text{TIMx\_CCR1}$  register, the auto-reload value in the  $\text{TIMx\_ARR}$  register, generate an update by setting the  $\text{UG}$  bit and wait for external trigger event on  $\text{TI2}$ .  $\text{CC1P}$  is written to '1' in this example.

In our example, the  $\text{DIR}$  and  $\text{CMS}$  bits in the  $\text{TIMx\_CR1}$  register should be low. The user only wants one pulse, so '1' must be written in the  $\text{OPM}$  bit in the  $\text{TIMx\_CR1}$  register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When  $\text{OPM}$  bit in the  $\text{TIMx\_CR1}$  register is set to '0', so the Repetitive Mode is selected.

### 19.3.12 Debug mode

When the microcontroller enters debug mode (Cortex®-M0 core halted), the  $\text{TIMx}$  counter either continues to work normally or stops, depending on  $\text{DBG\_TIMx\_STOP}$  configuration bit in  $\text{DBG}$  module.

## 19.4 Register description

Table 58. Overview of TIM16/17 registers

Offset	Acronym	Register Name	Reset	Section
0x00	TIM16/17_CR1	TIM16/17 control register 1	0x00000000	Section 19.4.1
0x04	TIM16/17_CR2	TIM16/17 control register 2	0x00000000	Section 19.4.2
0x0C	TIM16/17_DIER	TIM16/17 interrupt enable register	0x00000000	Section 19.4.3
0x10	TIM16/17_SR	TIM16/17 status register	0x00000000	Section 19.4.4
0x14	TIM16/17_EGR	TIM16/17 event generate register 1	0x00000000	Section 19.4.5
0x18	TIM16/17_CCMR1	TIM16/17 capture/compare mode register 1	0x00000000	Section 19.4.6
0x20	TIM16/17_CCER	TIM16/17 capture/compare enable register	0x00000000	Section 19.4.7
0x24	TIM16/17_CNT	TIM16/17 counter	0x00000000	Section 19.4.8
0x28	TIM16/17_PSC	TIM16/17 prescaler register	0x00000000	Section 19.4.9
0x2C	TIM16/17_ARR	TIM16/17 auto-reload register	0x00000000	Section 19.4.10
0x30	TIM16/17_RCR	TIM16/17 repetition counter register	0x00000000	??
0x34	TIM16/17_CCR1	TIM16/17 capture/compare register 1	0x00000000	Section 19.4.11
0x44	TIM16/17_BDTR	TIM16/17 brake and dead-time time register	0x00000000	Section 19.4.12
0x48	TIM16/17_DCR	TIM16/17 DMA control register	0x00000000	Section 19.4.13
0x4C	TIM16/17_DMAR	TIM16/17 full transfer address register	0x00000000	Section 19.4.14

### 19.4.1 TIM16/17 control register 1 (TIM16/17\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Reserved		CKD	ARPE		Reserved	OPM	URS	UDIS	CEN	

rw      rw      rw                                                                                           rw

Bit	Field	Type	Reset	Description
15:10	Reserved			Reserved, read as 0
9:8	CKD	rw	0x00	<p>Clock division</p> <p>This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the sampling frequency used by the digital filters (ETR, TIx).</p> <p>00: <math>t_{DTS} = t_{CK\_INT}</math></p> <p>01: <math>t_{DTS} = 2 \times t_{CK\_INT}</math></p> <p>10: <math>t_{DTS} = 4 \times t_{CK\_INT}</math></p> <p>11: reserved</p>
7	ARPE	rw	0x00	<p>Auto-reload preload enable</p> <p>0: TIMx_ARR register is not buffered</p> <p>1: TIMx_ARR register is buffered</p>
6:4	Reserved			Reserved and always read as 0.
3	OPM	rw	0x00	<p>One pulse mode</p> <p>0: Counter is not stopped at update event.</p> <p>1: Counter stops counting at the next update event (clearing the bit CEN).</p>
2	URS	rw	0x00	<p>Update request source</p> <p>This bit is set by software to select the UEV event sources.</p> <p>0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:</p> <ul style="list-style-type: none"> <li>- Counter overflow/underflow</li> <li>- Setting the UG bit</li> <li>- Update generation through the slave mode controller</li> </ul> <p>1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.</p>

Bit	Field	Type	Reset	Description
1	UDIS	rw	0x00	<p>Update disable</p> <p>This bit is set by software to enable/disable UEV event generation.</p> <p>0: UEV enabled. The Update (UEV) event is generated by one of the following events:</p> <ul style="list-style-type: none"> <li>- Counter overflow/underflow</li> <li>- Setting the UG bit</li> <li>- Update generation through the slave mode controller</li> </ul> <p>Buffered registers are then loaded with their preload values.</p> <p>1: UEV disabled. The Update event is not generated, shadow registers keep their values (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is sent from the slave mode controller.</p>
0	CEN	rw	0x00	<p>Counter enable</p> <p>0: Counter disabled</p> <p>1: Counter enabled</p> <p>CEN is automatically cleared when an update event occurs in one pulse mode.</p> <p>Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.</p>

#### 19.4.2 TIM16/17 control register 2 (TIM16/17\_CR2)

Address offset: 0x04

Reset value: 0x0000															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				OIS1N	OIS1	Reserved				CCDS	CCUS	Res.	CCPC		
				rw	rw					rw	rw				rw

Bit	Field	Type	Reset	Description
15:10	Reserved			Reserved, read as 0
9	OIS1N	rw	0x00	<p>Output Idle state 1 (OC1N output)</p> <p>0: OC1N = 0 after a dead-time when MOE = 0</p> <p>1: OC1N = 1 after a dead-time when MOE = 1</p> <p>Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).</p>
8	OIS1	rw	0x00	<p>Output Idle state 1 (OC1 output)</p> <p>0: OC1 = 0 (after a dead-time if OC1N is implemented) when MOE = 0</p> <p>1: OC1 = 1 (after a dead-time if OC1N is implemented) when MOE = 1</p> <p>Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).</p>

Bit	Field	Type	Reset	Description
7:4	Reserved			Reserved and read as 0.
3	CCDS	rw	0x00	Capture/compare DMA selection 0: CCx DMA request sent when CCx event occurs 1: CCx DMA requests sent when update event occurs
2	CCUS	rw	0x00	Capture/compare control update selection 0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only 1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on TRGI. Note: This bit acts only on channels that have a complementary output.
1	Reserved			Reserved and read as 0.
0	CCPC	rw	0x00	Capture/compare preloaded control 0: CCxE, CCxNE and OCxM bits are not preloaded 1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set. Note: This bit acts only on channels that have a complementary output.

#### 19.4.3 TIM16/17 interrupt enable register (TIM16/17\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CC1 DE	UD E	BIE	Res.	COM IE	Reserved			CC1 IE	UIE
						rw	rw	rw		rw				rw	rw

Bit	Field	Type	Reset	Description
15:10	Reserved			Reserved, read as 0
9	CC1DE	rw	0x00	Capture/Compare 1 DMA request enable 0: CC1 DMA request disabled 1: CC1 DMA request enabled
8	UDE	rw	0x00	Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled
7	BIE	rw	0x00	Break interrupt enable 0: Break interrupt disabled 1: Break interrupt enabled
6	Reserved			Reserved and read as 0.

Bit	Field	Type	Reset	Description
5	COMIE	rw	0x00	COM interrupt enable 0: COM interrupt disabled 1: COM interrupt enabled
4:2	Reserved			Reserved and read as 0.
1	CC1IE	rw	0x00	Capture/Compare 1 interrupt enable 0: CC1 interrupt disabled 1: CC1 interrupt enabled
0	UIE	rw	0x00	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

#### 19.4.4 TIM16/17 status register (TIM16/17\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Reserved	CC1 OF	Res.	BIF	Res.	COM IF		Reserved		CC1 IF	UIF
						rc_w0		rc_w0		rc_w0			rc_w0		rc_w0

Bit	Field	Type	Reset	Description
15:10	Reserved			Reserved, read as 0
9	CC1OF	rc_w0	0x00	Capture/Compare 1 overcapture flag This flag is set to '1' by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'. 0: No overcapture has been detected. 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set to 1.
8	Reserved			Reserved and read as 0.
7	BIF	rc_w0	0x00	Break interrupt flag This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active. 0: No break event occurred. 1: An active level has been detected on the break input.
6	Reserved			Reserved and read as 0.
5	COMIF	rc_w0	0x00	COM interrupt flag This flag is set by hardware on COM event (when capture/compare control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software. 0: No COM event occurred 1: COM interrupt pending.

Bit	Field	Type	Reset	Description
4:2	Reserved			Reserved and read as 0.
1	CC1IF	rc_w0	0x00	<p>Capture/Compare 1 interrupt flag            If channel CC1 is configured as output:            This flag is set by hardware when the counter matches the compare value. It is cleared by software.            0: No match.            1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register.            When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.</p> <p>If channel CC1 is configured as input:            This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.            0: No input capture occurred            1: The counter value has been captured in (copied to) TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)</p>
0	UIF	rc_w0	0x00	<p>Update interrupt flag            This bit is set by hardware on an update event. It is cleared by software.            0: No update occurred.            1: Update interrupt pending. This bit is set by hardware when the registers are updated:            - When CNT is reinitialized by software using the UG bit in TIM14_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.            - When CNT is reinitialized by a trigger event, if URS=0 and UDIS=0 in the TIMx_CR1 register.</p>

#### 19.4.5 TIM16/17 event generation register 1 (TIM16/17\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BG	Res.	COMG	Reserved		CC1G	UG		

Bit	Field	Type	Reset	Description
15:8	Reserved			Reserved, read as 0
7	BG	w	0x00	<p>Break generation</p> <p>This bit is set by software in order to generate a brake event, it is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: A break event is generated. MOET bit is cleared and TIF flag is set in the TIMx_SR register. Related interrupt or DMA can occur if enabled.</p>
6	Reserved			Reserved and read as 0.
5	COMG	w	0x00	<p>Capture/Compare control update generation</p> <p>This bit can be set by software, and automatically cleared by hardware.</p> <p>0: No action</p> <p>1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits</p> <p>Note: This bit acts only on channels that have a complementary output.</p>
4:2	Reserved			Reserved and read as 0.
1	CC1G	w	0x00	<p>Capture/Compare 1 generation</p> <p>This bit is set by software in order to generate a capture/compare event, and it is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: A capture/compare event is generated on channel CC1</p> <p>If channel CC1 is configured as output: CC1IF flag is set, corresponding interrupt or DMA is sent if enabled. The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.</p>
0	UG	w	0x00	<p>Update generation</p> <p>This bit can be set by software, and it is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: Reinitialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).</p>

#### 19.4.6 TIM16/17 capture/compare mode register 1 (TIM16/17\_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in output mode. For a given bit, OC<sub>xx</sub> describes its function when the channel is configured in output, IC<sub>xx</sub> describes its function when the channel is configured in input. So the user must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Res.	OC1M		OC1 PE	OC1 FE	CC1S		
IC1F								IC1PSC							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

### Output compare mode:

Bit	Field	Type	Reset	Description
15:7	Reserved			Reserved, always read as 0

Bit	Field	Type	Reset	Description
6:4	OC1M	rw	0x00	<p>Output compare 1 mode</p> <p>These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active levels depend on CC1P and CC1PN bits.</p> <p>000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the OC1REF. (this mode is used to generate a timing base)</p> <p>001: Set channel 1 to active level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).</p> <p>010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).</p> <p>011: Toggle. OC1REF level toggles when TIMx_CCR1=TIMx_CNT.</p> <p>100: Force inactive level. OC1REF is forced low.</p> <p>101: Force active level. OC1REF is forced high.</p> <p>110: PWM mode 1 - channel 1 is active as long as TIMx_CNT &lt; TIMx_CCR1, else inactive.</p> <p>111: PWM mode 2 - channel 1 is inactive as long as TIMx_CNT &lt; TIMx_CCR1, else active.</p> <p>Note 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output). Note 2: In PWM mode 1 or 2, the OC1REF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.</p>
3	OC1PE	rw	0x00	<p>Output compare 1 preload enable</p> <p>0: Preload function of TIMx_CCR1 register disabled. TIMx_CCR1 register can be written at anytime, the new value is taken in account immediately.</p> <p>1: Preload function of TIMx_CCR1 register enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.</p> <p>Note 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output). Note 2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.</p>

Bit	Field	Type	Reset	Description
2	OC1FE	rw	0x00	<p>Output compare 1 fast enable</p> <p>This bit is used to accelerate the effect of an event on the trigger input on the CC output.</p> <p>0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.</p> <p>1: An active edge on the trigger input acts like a compare match. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.</p>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC1 channel is configured as output;</li> <li>01: CC1 channel is configured as input, IC1 is mapped on TI1;</li> <li>10: CC1 channel is configured as input, IC1 is mapped on TI2;</li> <li>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register).</li> </ul> <p>Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).</p>

### Input capture mode:

Bit	Field	Type	Reset	Description
15:8	Reserved			Reserved, always read as 0

Bit	Field	Type	Reset	Description
7:4	IC1F	rw	0x00	<p>Input capture 1 filter</p> <p>This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to generate a transition on the output:</p> <ul style="list-style-type: none"> <li>0000: No filter, sampling is done at <math>f_{DTS}</math></li> <li>0001: <math>f_{SAMPLING}=f_{CK\_INT}</math>, <math>N = 2</math></li> <li>0010: <math>f_{SAMPLING}=f_{CK\_INT}</math>, <math>N = 4</math></li> <li>0011: <math>f_{SAMPLING}=f_{CK\_INT}</math>, <math>N = 8</math></li> <li>0100: <math>f_{SAMPLING}=f_{DTS}</math>, <math>N = 6</math></li> <li>0101: <math>f_{SAMPLING}=f_{DTS}</math>, <math>N = 8</math></li> <li>0110: <math>f_{SAMPLING}=f_{DTS}</math>, <math>N = 6</math></li> <li>0111: <math>f_{SAMPLING}=f_{DTS}/4</math>, <math>N = 8</math></li> <li>1000: <math>f_{SAMPLING}=f_{DTS}/8</math>, <math>N = 6</math></li> <li>1001: <math>f_{SAMPLING}=f_{DTS}/8</math>, <math>N = 8</math></li> <li>1010: <math>f_{SAMPLING}=f_{DTS}/16</math>, <math>N = 5</math></li> <li>1011: <math>f_{SAMPLING}=f_{DTS}/16</math>, <math>N = 6</math></li> <li>1100: <math>f_{SAMPLING}=f_{DTS}/16</math>, <math>N = 8</math></li> <li>1101: <math>f_{SAMPLING}=f_{DTS}/32</math>, <math>N = 5</math></li> <li>1110: <math>f_{SAMPLING}=f_{DTS}/32</math>, <math>N = 6</math></li> <li>1111: <math>f_{SAMPLING}=f_{DTS}/32</math>, <math>N = 8</math></li> </ul>
3:2	IC1PSC	rw	0x00	<p>Input capture 1 prescaler</p> <p>This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).</p> <ul style="list-style-type: none"> <li>00: no prescaler, capture is done each time an edge is detected on the capture input</li> <li>01: capture is done once every 2 events</li> <li>10: capture is done once every 4 events</li> <li>11: capture is done once every 8 events</li> </ul>
1:0	CC1S	rw	0x00	<p>Capture/Compare 1 selection</p> <p>This bit-field defines the direction of the channel (input/output) as well as the used input:</p> <ul style="list-style-type: none"> <li>00: CC1 channel is configured as output;</li> <li>01: CC1 channel is configured as input, IC1 is mapped on TI1;</li> <li>10: CC1 channel is configured as input, IC1 is mapped on TI2;</li> <li>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register).</li> </ul> <p>Note: CC1S bits are writable only when the channel is OFF</p>

(CC1E = 0 in TIMx\_CCER).

**19.4.7 TIM16/17 capture/compare enable register (TIM16/17\_CCER)**

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1 NP	CC1 NE	CC1P	CC1E
												rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:4	Reserved			Reserved, always read as 0
3	CC1NP	rw	0x00	Capture/Compare 1 complementary output polarity 0: OC1N active high 1: OC1N active low
2	CC1NE	rw	0x00	Capture/Compare 1 complementary output enable 0: Off - OC1N is not active. 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS2 and CC1E bits.
1	CC1P	rw	0x00	Capture/Compare 1 output polarity CC1 channel configured as output: 0: OC1 active high 1: OC1 active low CC1 channel configured as input: CC1P/CC1NP bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations (whether IC1 or the inverted signal of IC1 is used). 00: non-inverted/rising edge Capture is done on TI1FP1 rising edge (capture mode); TI1FP1 is not inverted. 01: inverted/falling edge edge Capture is done on TI1FP1 falling edge (capture mode); TI1FP1 is inverted. 10: reserved, do not use this configuration. 11: non-inverted/both edges Capture is done on both TI1FP1 rising and falling edges (capture mode); TI1FP1 is not inverted. Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit	Field	Type	Reset	Description
0	CC1E	rw	0x00	<p>Capture/Compare 1 output enable</p> <p>CC1 channel configured as output:</p> <p>0: Off - OC1 is not active.</p> <p>1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.</p> <p>CC1 channel configured as input:</p> <p>This bit determines if a capture of the counter value can actually be done into the TIMx_CCR1 register or not.</p> <p>0: capture disabled</p> <p>1: capture enabled</p>

Table 59. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states <sup>(1)</sup>	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	0	0	0	Output Disabled (not driven by the timer), OCx = 0, OCx_EN = 0	Output Disabled (not driven by the timer), OCxN = 0, OCxN_EN = 0
		0	0	1	Output Disabled (not driven by the timer), OCx = 0, OCx_EN = 0	OCxREF + Polarity, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		0	1	0	OCxREF + Polarity, OCx = OCxREF xor CCxP, OCx_EN = 1	Output Disabled (not driven by the timer), OCxN = 0, OCxN_EN = 0
		0	1	1	OCxREF + Polarity + dead-time, OCx_EN = 1	Complementary to OCxREF + Polarity + dead-time, OCxN_EN = 1
		1	0	0	Output Disabled (not driven by the timer), OCx = CCxP, OCx_EN = 0	Output Disabled (not driven by the timer), OCxN = CCxNP, OCxN_EN = 0
		1	0	1	Off-State (output enabled with inactive state) OCx = CCxP, OCx_EN = 1	OCxREF + Polarity, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		1	1	0	OCxREF + Polarity, OCx = OCxREF xor CCxP, OCx_EN = 1	Off-State (output enabled with inactive state) OCxN = CCxNP, OCxN_EN = 1
		1	1	1	OCxREF + Polarity + dead-time, OCx_EN = 1	Complementary to OCxREF + Polarity + dead-time, OCxN_EN = 1
0	0	X	0	0	Output Disabled (not driven by the timer),	

Control bits					Output states <sup>(1)</sup>		
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state	
0	0	0	0	1	Asynchronously: OCx = CCxP, OCx_EN = 0, OCxN = CCxNP, OCxN_EN = 0;		
			1	0	Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state.		
			1	1			
	1		0	0	Off-State (output enabled with inactive state)		
			0	1	Asynchronously: OCx = CCxP, OCx_EN = 1, OCxN = CCxNP, OCxN_EN = 1;		
			1	0	Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state.		
			1	1			

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO and AFIO registers.

#### 19.4.8 TIM16/17 counter (TIM16/17\_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description
15:0	CNT	rw	0x0000	Counter value

#### 19.4.9 TIM16/17 prescaler (TIM16/17\_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bit	Field	Type	Reset	Description
15:0	PSC	rw	0x0000	Prescaler value  The counter clock frequency (CK_CNT) is equal to $f_{CK\_PSC}/(PSC + 1)$ .  PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through slave controller when configured in "reset mode").

### 19.4.10 TIM16/17 auto-reload register (TIM16/17\_ARR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

ARR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:0	ARR	rw	0x0000	<p>Auto-reload value</p> <p>ARR is the value to be loaded in the actual auto-reload register.</p> <p>Refer to the Section 19.3.1 for more details about ARR update and behavior.</p> <p>The counter is blocked while the auto-reload value is null.</p>

### 19.4.11 TIM16/17 capture/compare register 1 (TIM16/17\_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:0	CCR1	rw	0x0000	<p>Capture/Compare 1 value</p> <p>If channel CC1 is configured as output:</p> <p>CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded immediately if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE).</p> <p>Else the preload value is copied in the active capture/compare 1 register when an update event occurs.</p> <p>The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC1 output.</p> <p>If channel CC1 is configured as input:</p> <p>CCR1 is the counter value transferred by the last input capture 1 event (IC1).</p>

### 19.4.12 TIM16/17 break and dead-time register (TIM16/17\_BDTR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK	DTG									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits AOE, BKP, BKE, OSSR and OSSI and DTG [7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

Bit	Field	Type	Reset	Description
15	MOE	rw	0x00	<p>Main output enable</p> <p>This bit is cleared asynchronously by hardware as soon as the break input is active. It is cleared as '0' by software or automatically set to '1' depending on the AOE bit configuration. It is acting only on the channels which are configured in output.</p> <p>0: OC and OCN outputs are disabled or forced to idle state. 1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).</p> <p>See OC/OCN enable description for more details (Section 19.4.7: TIM16/17 capture/compare enable register (TIM16/17_CCER)).</p>
14	AOE	rw	0x00	<p>Automatic output enable</p> <p>0: MOE can be set only by software 1: MOE can be set by software or automatically set at the next update event (if the break input is not active)</p> <p>Note: This bit can not be modified as long as LOCK level '1 has been programmed (LOCK bits in TIMx_BDTR register).</p>
13	BKP	rw	0x00	<p>Break polarity</p> <p>0: Break input BRK is active low 1: Break input BRK is active high</p> <p>Note: This bit can not be modified as long as LOCK level '1 has been programmed (LOCK bits in TIMx_BDTR register).</p> <p>Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.</p>
12	BKE	rw	0x00	<p>Break enable</p> <p>0: Break inputs (BRK and CSS clock failure event) disabled 1: Break inputs (BRK and CSS clock failure event) enabled</p> <p>Note: This bit can not be modified as long as LOCK level '1 has been programmed (LOCK bits in TIMx_BDTR register).</p> <p>Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.</p>

Bit	Field	Type	Reset	Description
11	OSSR	rw	0x00	<p>Off-state selection for Run mode</p> <p>This bit is used when MOE = 1 on channels having a complementary output. OSSR is not implemented if no complementary output is implemented in the timer. See OC/OCN enable description for more details (Section19.4.7: TIM1 capture/compare enable register (TIMx_CCER)).</p> <p>0: When the timer is inactive, OC/OCN outputs are disabled (OC/OCN enable output signal = 0).</p> <p>1: When the timer is inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1. Then, OC/OCN enable output signal = 1</p> <p>Note: This bit can not be modified as long as LOCK level '2 has been programmed (LOCK bits in TIMx_BDTR register).</p>
10	OSSI	rw	0x00	<p>Off-state selection for Idle mode</p> <p>This bit is used when MOE = 0 on channels configured as outputs.</p> <p>See OC/OCN enable description for more details (Section 19.4.7: TIM1 capture/compare enable register (TIMx_CCER)).</p> <p>0: When the timer is inactive, OC/OCN outputs are disabled (OC/OCN enable output signal = 0);</p> <p>1: When the timer is inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1. Then, OC/OCN enable output signal = 1.</p> <p>Note: This bit can not be modified as long as LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).</p>
9:8	LOCK	rw	0x00	<p>Lock configuration</p> <p>These bits offer a write protection against software errors.</p> <p>00: LOCK OFF - No bit is write protected;</p> <p>01: LOCK Level 1 = DTG, BKE, BKP and AOE bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register can no longer be written;</p> <p>10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits;</p> <p>11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.</p> <p>Note: The LOCK bits can be written only once after the system reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.</p>

Bit	Field	Type	Reset	Description
7:0	DTG	rw	0x00	<p>Dead-time generator setup This bit-field defines the duration of the dead-time inserted between the complementary outputs. Assuming that DT corresponds to this duration.</p> <p>DTG[7: 5] = 0xx:  <math>DT = (DTG[7: 0] + 1) \times t_{dtg}, t_{dtg} = t_{DTS}; DTG[7: 5] = 10x:</math>  <math>DT = (DTG[5: 0] + 1 + 64) \times t_{dtg}, t_{dtg} = 2 \times t_{DTS}; DTG[7: 5] = 110:</math>  <math>DT = (DTG[4: 0] + 1 + 32) \times t_{dtg}, t_{dtg} = 8 \times t_{DTS}; DTG[7: 5] = 111:</math>  <math>DT = (DTG[4: 0] + 1 + 32) \times t_{dtg}, t_{dtg} = 16 \times t_{DTS};</math></p> <p>Example if <math>t_{DTS}=125\text{ns}</math> (8MHz), dead-time possible values are:  125ns to 15875 ns by 125 ns steps,  16<math>\mu\text{s}</math> to 31750ns by 250ns steps,  32<math>\mu\text{s}</math> to 63<math>\mu\text{s}</math> by 1<math>\mu\text{s}</math> steps,  64<math>\mu\text{s}</math> to 126<math>\mu\text{s}</math> by 2<math>\mu\text{s}</math> steps</p> <p>Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 have been programmed (LOCK bits in TIMx_BDTR register).</p>

#### 19.4.13 TIM16/17 DMA control register (TIM16/17\_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DBL				Reserved				DBA			
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15:13	Reserved			Reserved, always read as 0
12:8	DBL	w	0x00	<p>DMA burst length This bit-field defines the length of DMA burst transfer (the timer recognizes a burst transfer when a write access is done to the TIMx_DMAR register), ie. the number of transferred bytes:</p> <p>00000: 1 byte  00001: 2 bytes  00010: 3 bytes  .....  10001: 18 bytes</p>
7:5	Reserved			Reserved and always read as 0.

Bit	Field	Type	Reset	Description
4:0	DBA	w	0x00	<p>DMA base address</p> <p>This bit-field defines the base-address for DMA transfers (when a write access is done through the TIMx_DMAR address). DMA is defined as an offset starting from the address of the TIMx_CR1 register.</p> <p>Example:</p> <ul style="list-style-type: none"> <li>00000: TIMx_CR1</li> <li>00001: TIMx_CR2</li> <li>00010: TIMx_SMCR</li> <li>.....</li> </ul> <p>Example: Let us consider the following transfer: DBL = 7 and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.</p>

#### 19.4.14 TIM16/17 address for full transfer (TIM16/17\_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15:0	DMAB	w	0x0000	<p>DMA register for burst accesses</p> <p>A write operation to the TIMx_DMAR register accesses the register located at the address:</p> <p>(TIMx_CR1 address) + (DBA + DMA index) x 4, where TIMx_CR1 address is the address of the control register 1 (TIMx_CR1), DBA is the base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and depends on DBL configured in TIMx_DCR.</p>

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers with the DMA transferring half words into the CCRx registers. This is done in the following steps:

1. Configure the corresponding DMA channel as follows:

- (a) DMA channel peripheral address is the DMAR register address
- (b) DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers
- (c) Number of data to transfer = 3 (See note below)
- (d) Circular mode disabled

2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.

3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

Note: This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice, the number of data to transfer should be 6. A buffer in the RAM should contain data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

## 20

# Independent Watchdog (IWDG)

## Independent Watchdog (IWDG)

### 20.1 Independent watchdog (IWDG) introduction

The device has embedded an independent watchdog which offers a combination of high safety level, timing accuracy and flexibility of use. The independent watchdog serves to detect and resolve malfunctions due to software failure, and to trigger system reset when the counter reaches a given timeout value.

The IWDG is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints.

### 20.2 IWDG main features

- Free-running downcounter
- Clocked from an independent oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x0000 is reached

### 20.3 IWDG functional description

The figure below shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0xCCCC in the Key register (IWDG\_KR), the counter starts counting down from the reset value of 0xFFFF. When it reaches the end of count value (0x000), a reset signal is generated (IWDG\_RESET).

Whenever the key value 0xAAAA is written in the IWDG\_KR key register, the IWDG\_RLR value is reloaded in the counter and the watchdog reset is prevented.

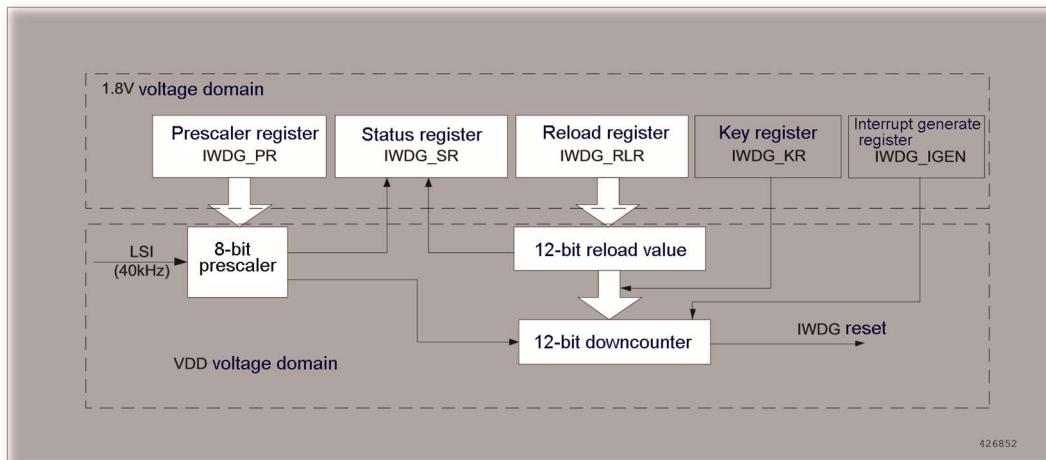


Figure 217. Independent watchdog block diagram

Note: The watchdog function is implemented in the  $V_{DD}$  voltage domain, still functional in Stop and Standby modes.

Table 60. IWDG timeout period (input clock at 40 KHz (LSI))

Prescale Coefficient	PR[2:0] bits	Min timeout RL[11:0]=0x000	Max timeout
/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

Note: These timings are given for a 40 kHz clock but the MCU internal oscillator frequency can vary from 30KHz to 60Khz.

Although this may be true that the oscillator frequency is accurate, the precise timing still depends on the phase difference between the APB interface clock and the oscillator clock. As a consequence, one complete oscillator cycle will always be uncertain.

### 20.3.1 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bytes (refer to the section of Embedded Flash Memory), the watchdog is automatically enabled at power-on, and will generate a system reset unless the Key register is written by the software before the counter reaches end of count.

### 20.3.2 Register access protection

Write access to the IWDG\_PR, IWDG\_RLR, and IWDG\_IGEN registers is protected. To modify them, first write the code 0x5555 in the IWDG\_KR register. A write access to this register with a different value will break the sequence and register access will be

protected again. This implies that it is the same case of the reload operation (writing 0xAAAA).

A status register is available to indicate that an update of the prescaler or the down-counter value is on going.

### 20.3.3 Debug mode

When the microcontroller enters debug mode (CPU core halted), the IWDG counter either continues to work normally or stops, depending on DBG\_IWDG\_STOP configuration bit in DBG module. For more details, refer to debug module section.

## 20.4 IWDG register description

Table 61. Overview of IWDG registers

Offset	Acronym	Register Name	Reset	Section
0x00	IWDG_KR	Key register	0x00000000	Section 20.4.1
0x04	IWDG_PR	Prescaler register	0x00000000	Section 20.4.2
0x08	IWDG_RLR	Reload register	0x00000FFF	Section 20.4.3
0x0C	IWDG_SR	Status register	0x00000000	Section 20.4.4
0x10	IWDG_CR	Control register	0x00000000	Section 20.4.5
0x14	IWDG_IGEN	Interrupt generate register	0x00000FFF	Section 20.4.6
0x18	IWDG_CNT	Counter register	0x00000000	Section 20.4.7

### 20.4.1 Key register (IWDG\_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

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

Bit	Field	Type	Reset	Description
31:16	Reserved			always read as 0.
15:0	KEY	w	0x0000	Key value (write-only register, read as 0x0000) These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0. Writing the key value 0x5555 to enable access to the

IWDG\_PR, IWDG\_RLR, IWDG\_IGEN and IWDG\_CR[IRQ\_SEL] registers. Writing the key value 0xCCCC starts the watchdog.

### 20.4.2 Prescaler register (IWDG\_PR)

Address offset: 0x04

Reset value: 0x0000 0000 (reset by Standby mode)

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

Reserved															
----------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved															PR
----------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

rw	rw	rw
----	----	----

Bit	Field	Type	Reset	Description
31:3	Reserved			always read as 0.
2:0	PR	rw	0x00	<p>Prescaler divider</p> <p>These bits are write access protected. They are set to select the prescaler dividers of the counter clock. The PVU bit in the IWDG_SR register must be reset in order to be able to change the prescaler dividers.</p> <p>000: prescaler divider = 4 100: prescaler divider = 64            001: prescaler divider = 8 101: prescaler divider = 128            010: prescaler divider = 16 110: prescaler divider = 256            011: prescaler divider = 32 111: prescaler divider = 256</p> <p>Note: Reading this register returns the prescaler value from the V<sub>DD</sub> voltage domain.</p> <p>This value may not be valid if a write operation is ongoing on this register. For this reason, the value read from this register is valid only when the PUV bit in the IWDG_SR register is reset.</p>

### 20.4.3 Reload register (IWDG\_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

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

Reserved															
----------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	RL
----------	----

rw rw

Bit	Field	Type	Reset	Description
31:12	Reserved			always read as 0.
11:0	RL	rw	0xFFFF	<p>Watchdog counter reload value</p> <p>These bits are write access protected. They serve to define the reload value of the watchdog counter. This value will be loaded to the counter each time 0xAAAA is written in the IWDG_KR register. Subsequently, the watchdog counter counts down from this value.</p> <p>The watchdog timeout period is a function of this value and the clock prescaler.</p> <p>Note: Reading this register returns the prescaler value from the V<sub>DD</sub> voltage domain. This value may not be valid if a write operation is ongoing on this register.</p> <p>For this reason, the value read from this register is valid only when the RUV bit in the IWDG_SR register is reset.</p>

#### 20.4.4 Status register (IWDG\_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												UPDATE	IVU	RVU	PVU
												r	r	r	r

Bit	Field	Type	Reset	Description
31:4	Reserved			always read as 0.
3	UPDATE	r	0x00	<p>Watchdog reload value update flag</p> <p>This bit is set by hardware to indicate that 0xAAAA is written into the IWDG_KR register. It is reset by hardware when the reload value of the watchdog is written into the counter and the counter is updated.</p>

2	IVU	r	0x00	Watchdog Interrupt Generate value update This bit is set by hardware to indicate that an update of the interrupt generate value is ongoing. It is reset by hardware when the interrupt generate value update operation is completed in the V <sub>DD</sub> voltage domain (takes up to 5 oscillator cycles at 40KHz). The interrupt generate value can be updated only when IVU bit is reset.
---	-----	---	------	---

Bit	Field	Type	Reset	Description
1	RVU	r	0x00	Watchdog counter reload value update This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V <sub>DD</sub> voltage domain (takes up to 5 oscillator cycles at 40KHz). The reload value can be updated only when RVU bit is reset.
0	PVU	r	0x00	Watchdog prescaler value update This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler value update operation is completed in the V <sub>DD</sub> voltage domain (takes up to 5 oscillator cycles at 40KHz). The prescaler value can be updated only when RVU bit is reset.

Note: If several reload values, prescaler values or interrupt generate values are used by application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value and to wait until IVU bit is reset before changing the interrupt generate value. However, after updating the prescaler and/or the reload value, it is not necessary to wait until RVU or PVU is reset before continuing code execution (even in case of low-power mode entry, the write operation is taken into account and will complete).

#### 20.4.5 IWDG control (IWDG\_CR) register

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:2	Reserved			Reserved, always read as 0

1	IRQ_CLR	rw	0x00	IWDG interrupt clear 1: interrupt cleared 0: no action
0	IRQ_SEL	rw	0x00	IWDG overflow operation select 1: interrupt generation enabled after overflow 0: reset generation enabled after overflow

#### 20.4.6 IWDG interrupt generate register (IWDG\_IGEN)

Address offset: 0x14

Reset value: 0x0000 0FFF

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				IGEN											
rw				rw											

Bit	Field	Type	Reset	Description
31:12	Reserved			Reserved, always read as 0.
11:0	IGEN	rw	0xFFF	IWDG Interrupt Generate value) These bits are write access protected. They serve to define the interrupt threshold of the watchdog counter. An interrupt is generated when the watchdog counter counts down until it reaches the threshold.

#### 20.4.7 IWDG counter register (IWDG\_CNT)

Address offset: 0x18

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT								PS							

Bit	Field	Type	Reset	Description
31:20	Reserved			Reserved and always read as 0.

19:8	CNT	r	0x00	Current value of IWDG counter.
7:0	PS	r	0x00	Current value of IWDG prescaler counter.

# 21

# Window Watchdog (WWDG)

## Window Watchdog (WWDG)

### 21.1 WWDG introduction

The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated by the reload counter when the downcounter value is greater than the window value. This implies that the counter must be refreshed in a limited time window.

### 21.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset:
  - Reset (if watchdog activated) when the downcounter value becomes less than 0x40.
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window.
- The early wakeup interrupt (EWI) is triggered when the downcounter reaches 0x40 (if enabled and the watchdog activated). It is applied to reload counter to avoid WWDG reset.

### 21.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG\_CR register) and when the 7-bit (T[6:0]) downcounter rolls over from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter value is greater than the value stored in the window register, then a reset is generated.

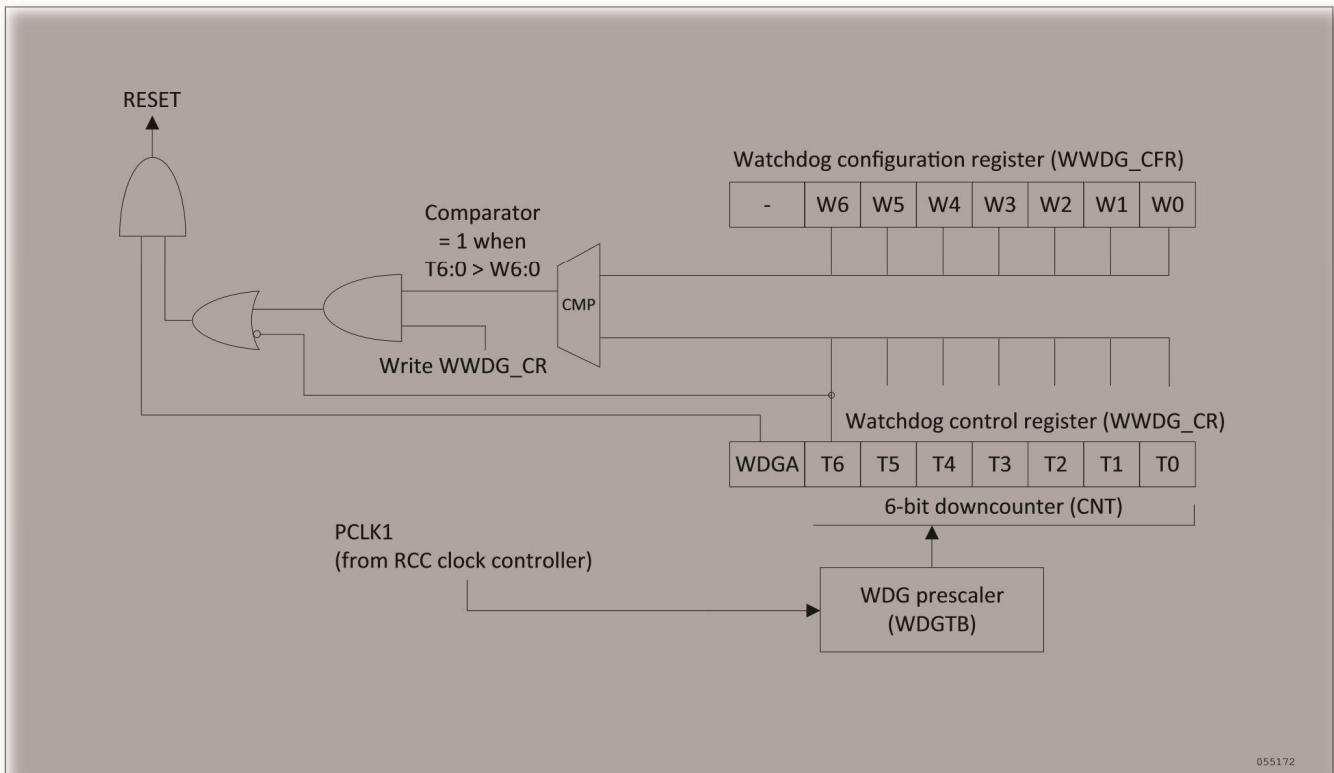


Figure 218. Watchdog block diagram

The application program must write in the WWDG\_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WWDG\_CR register must be between 0xFF and 0xC0.

- Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG\_CR register, then it cannot be disabled again except by a reset.

- Controlling the downcounter

This downcounter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG\_CR register.

The Configuration register (WWDG\_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F. The above figure describes the operation process of the window register.

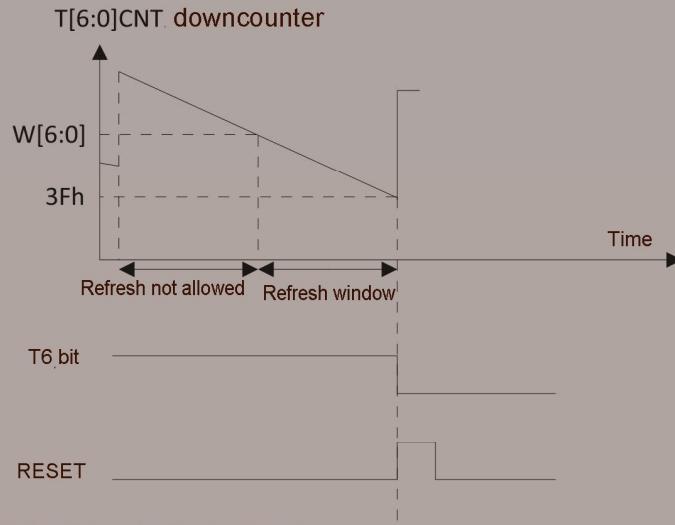
Another way to reload counter is to use the Early Wakeup Interrupt (EWI). The EWI interrupt is enabled by setting the WEI bit in the WWDG\_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to set a counter value greater than 0x40, so as to avoid WWDG reset. The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG\_SR register.

Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

## 21.4 How to program the watchdog timeout

The following figure shows the linear relationship between the increments loaded into the watchdog counter (CNT) and the time delay of the watchdog (in mS). This figure can be used as a reference for quick calculations, regardless of the time deviation. Please use the formula provided in the figure below when a higher accuracy is required.

Warning: When writing to the WWDG\_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.



The formula to calculate the timeout value is given by:

$$T_{\text{WWDG}} = T_{\text{PCLK1}} \times 4096 \times 2^{\text{WDGTB}} \times (T[5:0]+1) \quad (\text{mS})$$

where:

$T_{\text{WWDG}}$ : WWDG timeout

$T_{\text{PCLK1}}$ : APB1 clock period measured in ms

Minimum and maximum timeout values at PCLK1 = 36MHz

WDGTB	Min timeout value	Max timeout value
0	113µS	7.28mS
1	227µS	14.56mS
2	455µS	29.12mS
3	910µS	58.25mS

399420

Figure 219. Window watchdog timing diagram

## 21.5 Debug mode

When the microcontroller enters debug mode (CPU core halted), the WWDG counter either continues to work normally or stops, depending on `DBG_WWDG_STOP` configuration bit in DBG module. For more details, refer to debug module section.

## 21.6 WWDG register description

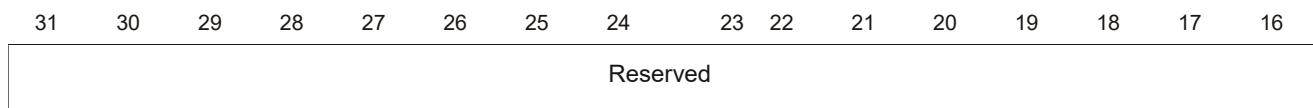
Table 62. Overview of WWDG registers

Offset	Acronym	Register Name	Reset	Section
0x00	WWDG_CR	Control register	0x0000007F	Section 21.6.1
0x04	WWDG_CFGR	Configuration register	0x0000007F	Section 21.6.2
0x08	WWDG_SR	Status register	0x00000000	Section 21.6.3

### 21.6.1 Control register (WWDG\_CR)

Address offset: 0x00

Reset value: 0x0000 007F



Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7	WDGA	rw	0x00	Activation bit This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset. 0: Watchdog disabled 1: Watchdog enabled
6:0	T	rw	0x7F	7 - bit counter (MSB to LSB) These bits contain the value of the watchdog counter. It is decremented by 1 every (4096 x 2 <sup>WDGTB</sup> ) PCLK1 cycles. A watchdog reset is produced when the counter value is decremented from 40h to 3Fh (T6 becomes cleared).

## 21.6.2 Configuration register (WWDG\_CFGR)

Address offset: 0x04

Reset value: 0x0000 007F

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

Bit	Field	Type	Reset	Description
31:10	Reserved			Reserved, always read as 0.
9	EWI	rw	0x00	Early wakeup interrupt When set, an interrupt occurs whenever the counter reaches the value 40h. This interrupt is only cleared by hardware after a reset.
8:7	WDGTB	rw	0x00	Timer base The time base of the prescaler can be modified as follows: 00: CK Counter Clock (PCLK1 div 4096) div 1 01: CK Counter Clock (PCLK1 div 4096) div 2 10: CK Counter Clock (PCLK1 div 4096) div 4 11: CK Counter Clock (PCLK1 div 4096) div 8
6:0	WINDOW	rw	0x7F	7-bit window value These bits contain the window value to be compared to the downcounter.

## 21.6.3 Status register (WWDG\_SR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

Bit	Field	Type	Reset	Description
31:1	Reserved			Reserved, always read as 0.
0	EWIF	rc_w0	0x00	<p>Early wakeup interrupt flag</p> <p>This bit is set by hardware when the counter has reached the value 40h. It must be cleared by software by writing '0'.</p> <p>A write of '1' in the bit has no effect. This bit is also set if the interrupt is not enabled.</p>

## 22

# Serial Peripheral Interface (SPI)

## Serial Peripheral Interface (SPI)

### 22.1 SPI introduction

The serial peripheral interface (SPI) generally supports communications across the boards, e.g. expanding serial Flash and ADC. Many devices from IC manufacturers are compatible with SPIs.

The SPI allows full-duplex, synchronous, serial communication of MCU with external devices. The communication can be implemented by the application software through the status query and SPI interrupt.

### 22.2 Main features

- Fully compatible with SPI of Motorola
- Supports DMA request
- Full-duplex synchronous transfers on three lines
- 16-bit programmable baud rate generator
- Master and Slave modes supported
- 8-byte FIFO transmission and reception
- Max SPI clock is PCLK/2 in the master mode (PCLK is the APB clock); max SPI clock is PCLK/4 in the slave mode
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Operations of one master and multiple slaves are supported
- Simultaneous transmission and reception of 1- to 32-bit data
- 1- to 32-bit data transmission and reception are only supported in the LSB mode instead of the MSB mode, except for the 8-bit data
- Transmission and reception buffer, 8 bits of data size respectively, are supported
- Interrupt trigger capability
  - Tx buffer is empty
  - Tx buffer and Tx shift register are both empty
  - Tx end underflow
  - Reception of effective bytes
  - Rx buffer overflow
  - Rx buffer full
  - Reception of a specified number of bytes in the master mode

## 22.3 SPI functional description

### 22.3.1 General description

The block diagram of the SPI is shown in the diagram below.

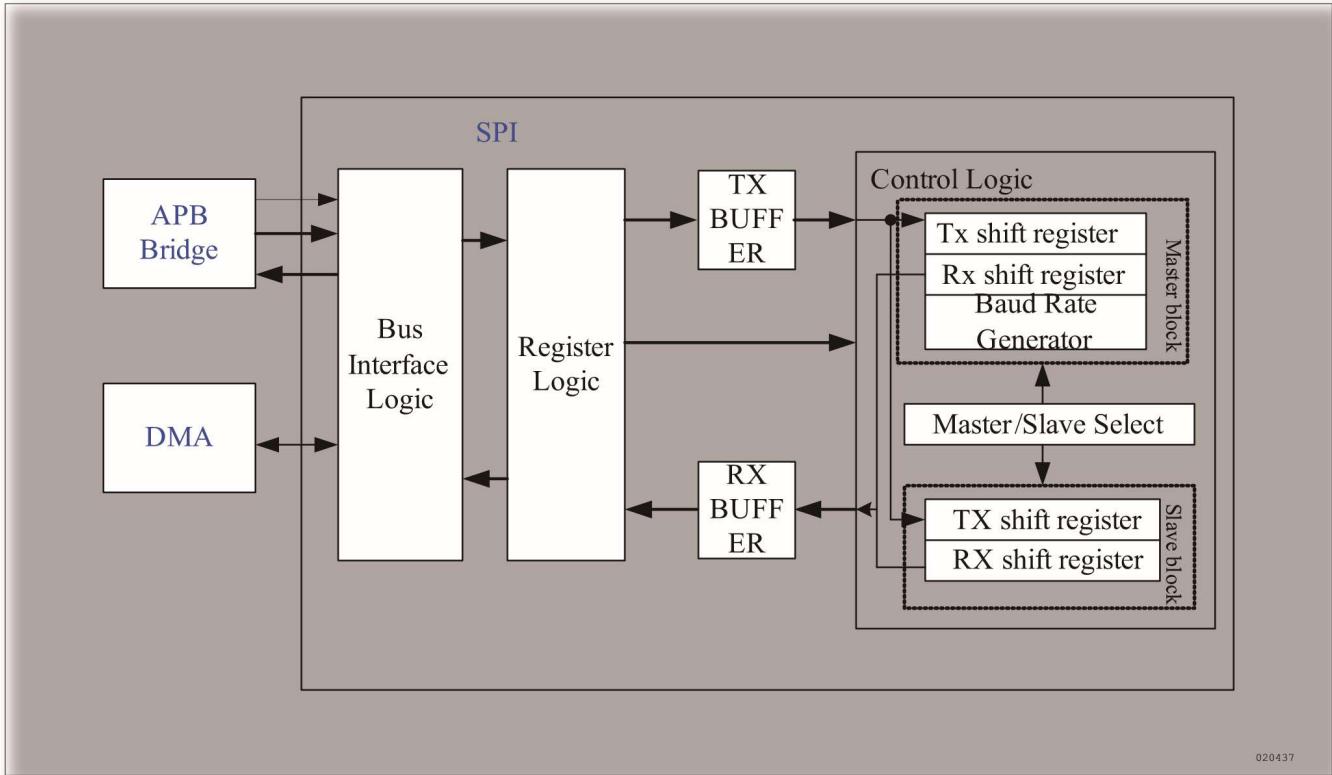


Figure 220. SPI block diagram

SPI supports simultaneous transmission and reception of 1- to 32-bit data. SPI can be configured in Slave mode or in Master mode in a Master environment. Four possible timing relationships may be chosen by configuring the clock polarity CPOL and clock phase CPHA. The programmable data order is MSB-first or LSB-first shifting.

The same clock is adopted in the transmission and reception. Data is output on the rising edge or falling edge of the clock, while latched on the opposite effective edge of SCLK. SPI serves to swap data, thus the data must be read at the end of the transfer, even though it is not significant data. In the SPI mode, the clock phases and polarities of the Master and the Slave which communicates with the Master must be identical.

Usually, the SPI is connected to external devices through four pins:

- MISO: Master In/Slave Out pin. This pin can be used to transmit data in slave mode and receive data in master mode.
- MOSI: Master Out/Slave In pin. This pin can be used to transmit data in master mode and receive data in slave mode.
- SCK: Serial Clock output for SPI masters and input for SPI slaves.
- NSS: Slave select. This is an optional pin to select a master/slave device. This pin acts as a ‘chip select’ to let the SPI master communicate with specific slaves individually and to avoid contention on the data lines. Slave NSS pins can be driven as standard IO ports on

the master device. The NSS pin may also be used as an output if enabled and driven low if the SPI is in master configuration. In this manner, all SPI NSS pins connected to the Master NSS pin see a low level.

An example of interconnections between a single master and a single slave is illustrated in the following figure.

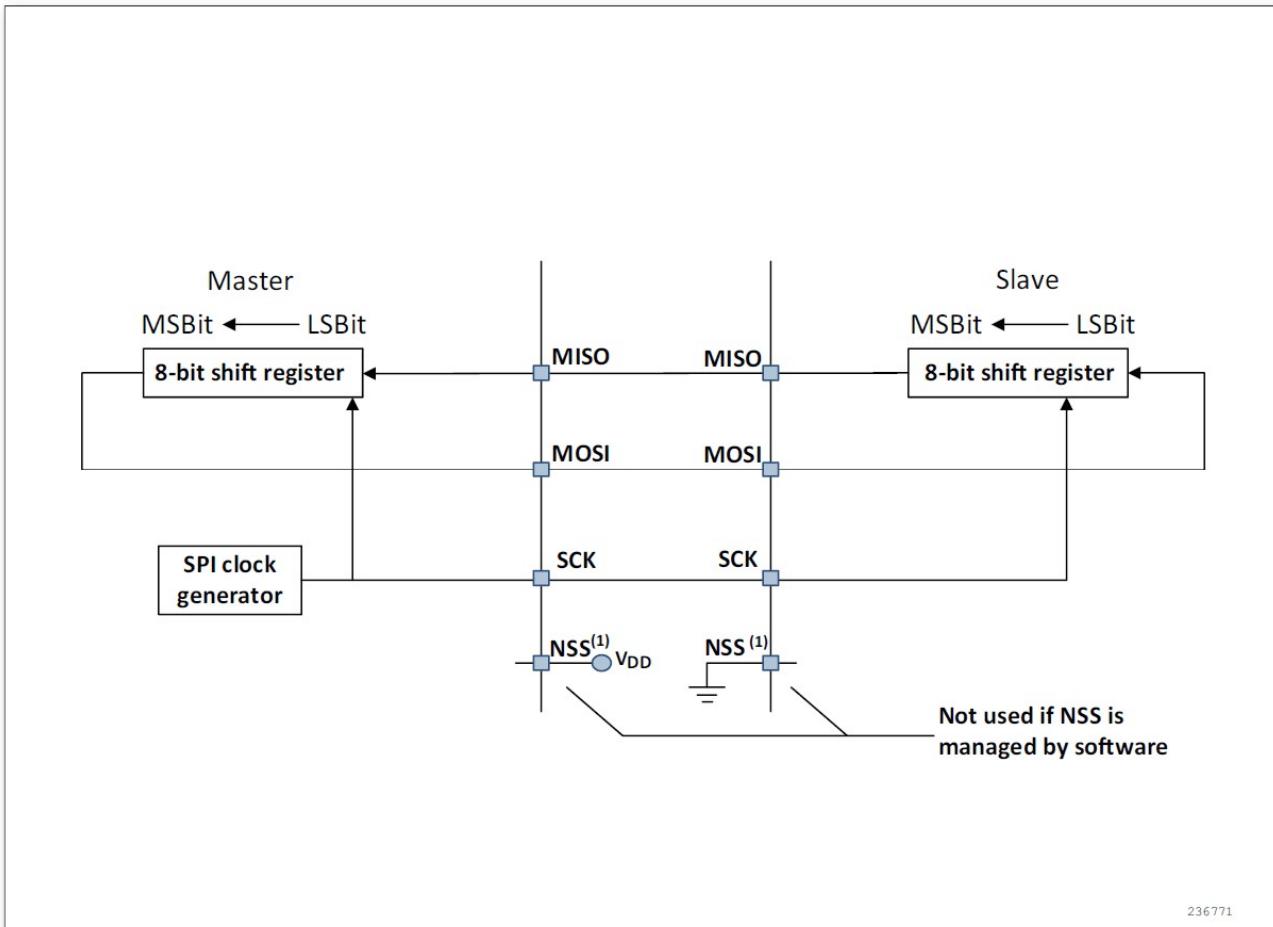


Figure 221. Single master/single slave application

The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

The communication is always initiated by the master. When the master device transmits data to a slave device via the MOSI pin, the slave device responds via the MISO pin. This implies full-duplex communication with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

### Clock signal phase and polarity

The CPOL and CPHA bits in the SPI\_CCTL register can generate four possible combinations of timing relationships. The CPOL (clock polarity) bit controls the idle state level of the SCK clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state or a low level between two transmissions. If CPOL is set, the SCK pin has a high-level idle state or a high level between two transmissions.

If the CPHA (clock phase) bit is set, the first data bit is latched on the occurrence of the first clock edge (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset) of the SCK clock. Concurrently, the first received data bit is captured. SPI changes the serial data at the first transition of SCK clock in the transmission (at this point the clock shifts in the opposite direction of the idle state) and captures the data on the next edge.

If the CPHA (clock phase) bit is reset, the first data bit is latched on the occurrence of the second clock edge (rising edge if the CPOL bit is set, falling edge if the CPOL bit is reset) of the SCK clock. Concurrently, the first received data bit is captured. SPI captures the serial data at the first transition of SCK clock in the transmission (at this point the clock shifts in the opposite direction of the idle state) and changes the data on the next edge.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge. The diagram 222 shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

### High-speed transmission

With regard to board-level time delay sensitivity in the high-speed transmission mode, the TXEDGE and RXEDGE control bits in the SPI\_CCTL register conduct the time adjustment for transmission phase and reception sampling.

- In the Slave mode: Tx data is sent to the data bus immediately when TXEDGE = 1 (suitable for the high-speed mode, SPBRG = 4); Tx data is sent to the data bus after one effective clock edge when TXEDGE = 0 (suitable for the low-speed mode, SPBRG > 4).
- In the Master mode: Data is captured on the end edge of clock in the Tx data bits when RXEDGE = 1 (suitable for the high-speed mode); data is captured in the middle of the Tx data bits when RXEDGE = 0.
  1. Prior to changing the CPOL/CPHA bits, the SPI must be disabled by resetting the SPIEN bit.
  2. Master and slave must be programmed with the same timing mode.
  3. The idle state of SCK must correspond to the polarity selected in the SPI\_CCTL register (by pulling up SCK in the idle state if CPOL = 1 or pulling down SCK in the idle state if CPOL = 0).

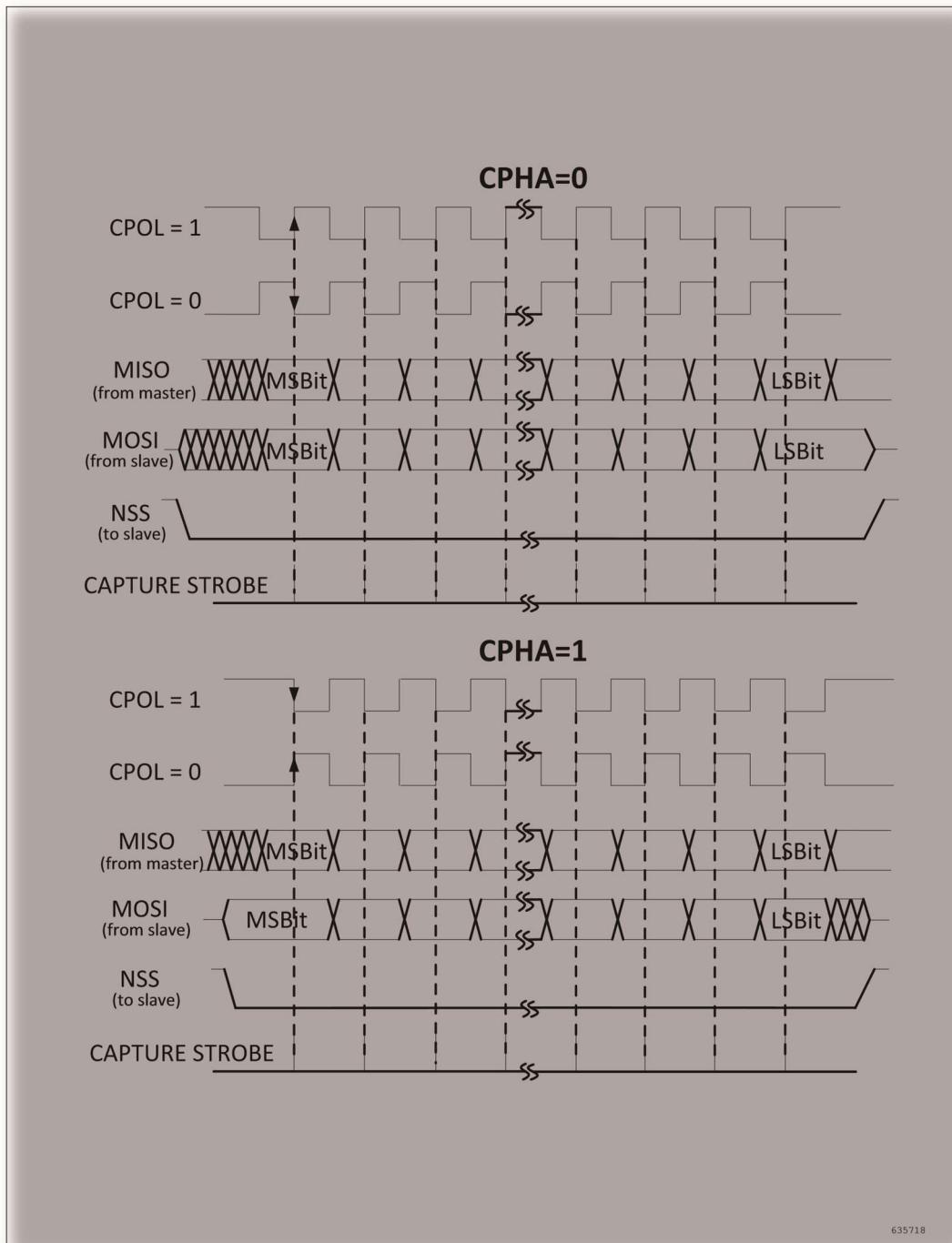


Figure 222. Data clock timing diagram

### Data frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LSBFE bit in the SPI\_CCTL register.

Each data frame is 7 or 8 bits long depending on the SPILEN bit in the SPI\_CCTL register. The selected data frame format is applicable for transmission and/or reception.

Moreover, 1 to 32 bits of data frame should be configured in the SPI\_EXTCTL register. This configuration should be coupled with the following settings: DW8\_32 bit set to 0 in the SPI\_GCTL register, LSBFE bit as

well as SPILEN bit set to 1 in the SPI\_CCTL register. The DMA data size should be 8 bits to match with the DMA data transfer.

### 22.3.2 SPI in slave mode

In the slave configuration, the serial clock is received on the SCK pin from the master device. The value set in the SPI\_SPBRG register does not affect the data transfer rate.

#### Procedure

1. Set the SPILEN bit to define 7- or 8-bit data frame format.
2. Select the CPOL and CPHA bits to define phase relationships between the data transfer and the serial clock. For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and the master device.
3. The frame format (MSB-first or LSB-first depending on the value of the LSBFE bit in the SPI\_CCTL register) must be the same as the master device.
4. Clear the MDOE bit and set the SPIEN bit to assign corresponding pins to work in the SPI mode. In this configuration, the MOSI pin is a data input and the MISO pin is a data output.

#### Transmit sequence

The data byte is parallel-loaded into the Tx buffer during a write cycle.

The transmit sequence begins when the slave device receives the clock signal and the first data bit is received on its MOSI pin. The first bit is then sent. The remaining bits are loaded into the shift-register. The TX\_INTF flag in the SPI\_INTSTAT register is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXIEN bit in the SPI\_INTEN register is set.

#### Receive sequence

For the receiver, when data transfer is completed:

- The data in shift register is transferred to Rx Buffer and the RX\_INTF flag is set in the SPI\_INTSTAT register.
- An Interrupt is generated if the RXIEN bit is set in the SPI\_INTEN register.

After the last sampling clock edge, the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI\_RXREG register is read, the SPI peripheral returns this buffered value.

### 22.3.3 SPI in master mode

In the master configuration, the serial clock is generated on the SCK pin.

#### Procedure

1. Use the SPI\_SPBRG register to define the serial clock baud rate.
2. Select the CPOL and CPHA bits to define phase relationships between the data transfer and the serial clock.
3. Set the SPILEN bit to define 7- or 8-bit data frame format.
4. Configure the LSBFE bit in the SPI\_CCTL register to define the frame format.
5. If the data will be only received and not transmitted, the SPI\_RNDNR register should be configured to define the number of bytes to be received.
6. The MDOE and SPIEN bits must be set.

In this configuration, the MOSI pin is a data output and the MISO pin is a data input. The NSS is the Slave select signal output.

## Transmit sequence

The transmit sequence begins when a byte is written in the Tx Buffer. The data byte is parallel-loaded into the shift register (from the internal bus) during the first bit transmission and then shifted out serially to the MOSI pin. MSB first or LSB first depends on the LSBFE bit in the SPI\_CCTL register. The TX\_INTF flag is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXIEN bit in the SPI\_INTEN register is set.

## Receive sequence

For the receiver, when data transfer is completed:

- The data in shift register is transferred to Rx Buffer and the RX\_INTF flag is set in the SPI\_INTSTAT register.
- An Interrupt is generated if the RXIEN bit is set in the SPI\_INTEN register.

After the last sampling clock edge, the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI\_RXREG register is read, the SPI peripheral returns this buffered value.

If the data will be only received and not transmitted, the RXMATCH\_INTF bit should be set on the transfer completion of bytes defined by RXDNR, which implies that all the data is received and the clock signal is no longer sent in the Master mode.

### 22.3.4 Status flags

To ensure the convenient operation of the software, four current status flags and seven interrupt status flags are provided for the application to monitor the state of the SPI bus. The current status flags are read-only, automatically set and reset by hardware. Interrupt status flags are set when an event occurs. A CPU interrupt is generated when the flags are enabled. They are cleared by software.

SPI has one 8-byte Tx buffer and one 8-byte Rx buffer. According to the DW8\_32 bit in the SPI\_GCTL, the CPU can read/write one or four bytes at one time. According to the DW8\_32 bit configuration, the Tx buffer and Rx buffer have one status flag of one byte or effective entry, respectively.

Table 63. SPI status

Category	Status flags	Buffer and signal status
Interrupt Status	TX_INTF	According to the DW8_32 bit configuration, the minimum space of one effective entry is provided for completing one write operation on the Tx data register.
	RX_INTF	According to the DW8_32 bit configuration, the minimum space of one effective entry is provided for completing one read operation on the Rx data register.
	UNDERRUN_INTF	Tx buffer is empty and transmission is repeated.
	RXOERR_INTF	Rx buffer is not empty and overridden.
	RXMATCH_INTF	Not empty. Last entry is transferred to the Rx buffer.
	RXFULL_INTF	Rx buffer is full and not able to receive new data.

	TXEPT_INTF	Tx buffer is empty and not able to transmit data.
Current status	RXAVL_4BYTE	Rx buffer contains effective data beyond 4 bytes.
	TXFULL	Tx buffer is full.
	TXEPT	Tx buffer is empty.
	RXAVL	Rx buffer is not empty and able to receive at least one byte.
		When the TXTLF = 00 in the SPI_GCTL register, the TX_INTF is set if the Tx buffer has the space not less than one idle entry; when the TXTLF = 01, the TX_INTF is set if the Tx buffer has over 50% idle space.

When the RXTLF = 00 in the SPI\_GCTL register, the RX\_INTF is set if the Rx buffer has not less than one effective entry;

when the RXTLF = 01, the RX\_INTF is set if the Rx buffer contains effective data accounting for over 50%.

### 22.3.5 Baud rate configuration

The baud rate is the generated frequency of SCLK, generally the frequency division of PCLK. BRG is a 16-bit baud rate generator.

The SPBREG register controls the counting cycle of the 16-bit counter.

If expected baud rate and  $f_{\text{pclk}}$  (frequency of APB module) are provided, the formula shown in the following table can be used to calculate an approximate value to assign the SPBRG register, where X represents the value for SPBRG register (2 ~ 65535).

Table 64. Baud rate formula

Mode	Formula
SPI mode	Baud rate = $f_{\text{pclk}}/X$

### 22.3.6 SPI communication using DMA

To operate at its maximum speed, the SPI Tx buffer needs to be fed with data and the data received on the Rx buffer should be read to avoid overrun. To ensure the convenient high-speed data transfer, the SPI features a DMA capability implementing a simple request/acknowledge protocol.

When the DMAEN bit in the SPI\_GCTL register is set, the SPI module sends the request of DMA data transfer. DMA requests of both Tx and Rx buffers are enabled through DMAEN.

- In transmission, when the TXTLF = 00 in the SPI\_GCTL register, a DMA request is issued if the Tx buffer has the space not less than one idle entry; when the TXTLF = 01, a DMA request is issued if the Tx buffer has over 50% idle space. Only one DMA transfer is allowed for each request. The number of DMA transfer and the data size of Tx buffer each time depend on DW8\_32.
- In reception, when the RXTLF = 00 in the SPI\_GCTL register, a DMA request is issued if the Rx buffer has the space not less than one valid entry; when the RXTLF = 01, a DMA request is issued if the Rx buffer has over 50% valid data. Only one DMA transfer is allowed for each request. The number of DMA transfer and the data size of Rx buffer each time depend on DW8\_32.

## 22.4 Register file and memory mapping description

Table 65. Overview of SPI registers

Offset	Acronym	Register Name	Reset	Section
0x00	SPI_TXREG	Tx data register	0x00000000	Section 22.4.1
0x04	SPI_RXREG	Rx data register	0x00000000	Section 22.4.2
0x08	SPI_CSTAT	Current status register	0x00000001	Section 22.4.3
0x0C	SPI_INTSTAT	Interrupt status register	0x00000000	Section 22.4.4
0x10	SPI_INTEN	Interrupt enable register	0x00000000	Section 22.4.5
0x14	SPI_INTCLR	Interrupt clear register	0x00000000	Section 22.4.6
0x18	SPI_GCTL	Global control register	0x00000004	Section 22.4.7
0x1C	SPI_CCTL	Common control register	0x00000008	Section 22.4.8
0x20	SPI_SPBRG	Baud rate generator	0x00000002	Section 22.4.9
0x24	SPI_RXDNR	Rx data number register	0x00000001	Section 22.4.10
0x28	SPI_NSSR	Slave select register	0x000000FF	Section 22.4.11
0x2C	SPI_EXTCTL	Data control register	0x00000008	Section 22.4.12

**22.4.1 Transmit data register (SPI\_TXREG)**

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:0	TXREG	rw	0x0000 0000	Transmit data register The effective number of bits is controlled by the DW8_32 bit. 0: only low 8 bits are effective 1: TXREG[31 : 0] bits are all effective

**22.4.2 Receive data register (SPI\_RXREG)**

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:0	RXREG	r	0x0000 0000	Receive data register The effective number of bits is controlled by the DW8_32 bit. 0: only low 8 bits are effective 1: RXREG[31 : 0] bits are all effective. This register is read-only and not writable.

**22.4.3 Current status register (SPI\_CSTAT)**

Address offset: 0x08

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RXFADDR				TXFADDR				RXAVL_4BYTE	TXFULL	RXAVL	TXEPT
r r r r r r r r r r r r r r r r															
Bit	Field	Type	Reset	Description											
31:12	Reserved			always read as 0.											
11:8	RXFADDR	r	0x00	number of effective data in the current Rx buffer											
7:4	TXFADDR	r	0x00	number of effective data in the current Tx buffer											
3	RXAVL_4BYTE	r	0x00	Receive available 4 byte data message 1: Rx buffer contains data beyond 4 bytes. 0: Rx buffer contains data less than 4 bytes.											
2	TXFULL	r	0x00	Transmitter FIFO full status bit 1: Tx buffer is full. 0: Tx buffer is not full.											
1	RXAVL	r	0x00	Receive available byte data message This bit is set when the Rx buffer receives a complete byte. 1: Rx buffer has received an effective byte. 0: Rx buffer is empty. This bit is read-only, automatically set and reset by hardware.											
0	TXEPT	r	0x01	Transmitter empty bit 1: Tx buffer and Tx shift register are both empty. 0: Transmitter is not empty. This bit is read-only, automatically set and reset by hardware.											

#### 22.4.4 Interrupt status register (SPI\_INTSTAT)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXEPT_INTF	RXFULL_INTF	RX_MATCH_INFT	RXO_ERR_INFT	UNDE_RRUN_INFT	RX_INTF	TX_INTF	r

Bit	Field	Type	Reset	Description
31:7	Reserved			always read as 0.
6	TXEPT_INTF	r	0x00	<p>Transmitter empty interrupt flag bit</p> <p>This bit is set automatically by hardware. The TXEPT_ICLR bit is cleared by writing to the INTCLR register.</p> <p>1: Tx buffer and Tx shift register are both empty. 0: Transmitter is not empty.</p> <p>Note: This bit is an interrupt status signal whereas TXEPT is a status signal.</p>
5	RXFULL_INTF	r	0x00	<p>RX FIFO full interrupt flag bit</p> <p>This bit is set automatically by hardware. The RXFULL_ICLR bit is cleared by writing to the INTCLR register.</p> <p>1: RX buffer is full. 0: RX buffer is not full.</p>
4	RXMATCH_INFT	r	0x00	<p>Receive data match the RXDNR number, the receive process will be completed and generate the interrupt</p> <p>This bit is set automatically by hardware. The RXMATCH_ICLR bit is cleared by writing to the INTCLR register.</p> <p>1: A number of bytes specified by the RXDNR register are received. 0: A number of bytes specified by the RXDNR register are not received.</p>
3	RXOERR_INTF	r	0x00	<p>Receive overrun error interrupt flag bit</p> <p>This bit is set automatically by hardware. The RXOERR_ICLR bit is cleared by writing to the INTCLR register.</p> <p>1: overrun error 0: No overrun error</p>

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

2	UNDERRUN_INTF	r	0x00	SPI underrun interrupt flag bit in the Slave mode This bit is set automatically by hardware. The UNDERRUN_ICLR bit is cleared by writing to the INTCLR register. 1: underrun error 0: no underrun error
1	RX_INTF	r	0x00	Receive data available interrupt flag bit This bit is set automatically by hardware. The RX_ICLR bit is cleared by writing to the INTCLR register. This bit is set when the Rx buffer receives a complete byte. 1: Rx buffer has received data of effective byte 0: Rx buffer is empty.
0	TX_INTF	r	0x00	Transmit FIFO available interrupt flag bit (one byte is sent) This bit is automatically set by hardware and reset when the Tx buffer is not empty. 1: Tx buffer is valid. 0: Tx buffer is invalid.

#### 22.4.5 Interrupt enable register (SPI\_INTEN)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								TXEPT_IEN	RXFULL_IEN	RX_MATCH_IEN	RXO_ERR_IEN	UNDE_RRUN_IEN	RX_IEN	TX_IEN		
rw								rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Field	Type	Reset	Description
31:7	Reserved			always read as 0.
6	TXEPT_IEN	rw	0x00	Transmit empty interrupt enable bit 1: interrupt enabled 0: interrupt disabled
5	RXFULL_IEN	rw	0x00	Receive FIFO full interrupt enable bit 1: interrupt enabled 0: interrupt disabled
4	RXMATCH_IEN	rw	0x00	Receive data complete interrupt enable bit 1: interrupt enabled 0: interrupt disabled

Serial Peripheral Interface (SPI)					UM_MM32F013x_Ver1.00
3	RXOERR_ IEN	rw	0x00	Receive overrun error interrupt enable bit	
				1: interrupt enabled 0: interrupt disabled	
2	UNDERRUN_ IEN	rw	0x00	Transmitter underrun interrupt enable bit (SPI slave mode only) 1: interrupt enabled 0: interrupt disabled	
1	RX_IEN	rw	0x00	Receive FIFO interrupt enable bit 1: interrupt enabled 0: interrupt disabled	
0	TX_IEN	rw	0x00	Transmit FIFO empty interrupt enable bit 1: interrupt enabled 0: interrupt disabled	

#### 22.4.6 Interrupt clear register (SPI\_INTCLR)

Address offset: 0x14  
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								TXEPT_ ICLR	RXFULL_ ICLR	RX MATCH_ ICLR	RXO ERR_ ICLR	UNDE RRUN_ ICLR	RX_ ICLR	TX_ ICLR		r

Bit	Field	Type	Reset	Description
31:7	Reserved			always read as 0.
6	TXEPT_ICLR	w	0x00	Transmitter empty interrupt clear bit 1: interrupt cleared 0: interrupt uncleared
5	RXFULL_ICLR	w	0x00	Receiver buffer full interrupt clear bit 1: interrupt cleared 0: interrupt uncleared
Bit	Field	Type	Reset	Description
4	RXMATCH_ ICLR	w	0x00	Receive completed interrupt clear bit 1: interrupt cleared 0: interrupt uncleared
3	RXOERR_	w	0x00	Receive overrun error interrupt clear bit

## ICLR

1: interrupt cleared

0: interrupt uncleared

2	UNDERRUN_	w	0x00	Transmitter underrun interrupt clear bit (SPI slave mode only)
ICLR				1: interrupt cleared 0: interrupt uncleared
1	RX_ICLR	w	0x00	Receive interrupt clear bit 1: interrupt cleared 0: interrupt uncleared
0	TX_ICLR	r	0x00	Transmitter FIFO empty interrupt clear bit 1: interrupt cleared 0: interrupt uncleared

**22.4.7 Global control register (SPI\_GCTL)**

Address offset: 0x18

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	NSS TOG	DW8_32	NSS	DMAEN	TXTLF	RXTLF	RXEN	TXEN	MODE	INTEN	SPIEN				

Bit	Field	Type	Reset	Description
31:13	Reserved			always read as 0.
12	NSSTOG	rw	0x00	Slave select automatic toggle 1: NSS signal toggles automatically on the completion of data transfer 0: NSS signal does not toggle Note: This bit is only valid in the Master mode.
Bit	Field	Type	Reset	Reset
11	DW8_32	rw	0x00	Valid byte or doubleword data select signal 0: only low 8 bits are effective 1: 32 bits are all effective Note: The access is allowed in the specified data format.

## Serial Peripheral Interface (SPI)

UM MM32F013x Ver1.00

10	NSS	rw	0x00	<p>NSS select signal that from software or hardware 0: controlled by the NSSR register 1: controlled by hardware automatically during data transfer</p>
9	DMAEN	rw	0x00	<p>DMA access mode enable 0: DMA mode disabled 1: DMA mode enabled</p>
8:7	TXTLF	rw	0x00	<p>TX FIFO trigger level bit 00: a DMA or interrupt request is issued if the Tx buffer has the space not less than one idle entry 01: a DMA or interrupt request is issued if the Tx buffer has over 50% idle space 1x: Reserved Note: when DW8_32 = 0, one data entry represents one byte; when DW8_32 = 1, one data entry represents four bytes;</p>
6:5	RXTLF	rw	0x00	<p>RX FIFO trigger level bit 00: a DMA or interrupt request is issued if the Rx buffer has the space not less than one idle entry 01: a DMA or interrupt request is issued if the Rx buffer has over 50% valid data 1x: Reserved Note: when DW8_32 = 0, one valid data entry represents one byte; when DW8_32 = 1, one valid data entry represents four bytes;</p>
4	RXEN	rw	0x00	<p>Receive enable bit 1: Reception enabled. 0: Reception disabled. Meanwhile, Rx buffer can be cleared. Note: When SPI is working only in the Master receive mode, txen must be set to 0.</p>
3	TXEN	rw	0x00	<p>Transmit enable bit 1: Transmission enabled 0: Transmission disabled. Meanwhile, Tx buffer can be cleared. Note: Transmission and reception are simultaneous in the Master mode.</p>
Bit	Field	Type	Reset	Description
2	MODE	rw	0x01	<p>Master mode bit 1: Master mode (the internal BRG generates the serial clock) 0: Slave mode (the external host generates the serial clock)</p>

1	INTEN	rw	0x00	SPI interrupt enable bit 1: SPI interrupt enabled 0: SPI interrupt disabled
0	SPIEN	rw	0x00	SPI select bit 0 : SPI disabled (reset status) 1 : SPI enabled

### 22.4.8 Common control register (SPI\_CCTL)

Address offset: 0x1C

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								CPH ASEL	TX EDGE	RX EDGE	SPI LEN	LSB FE	CPOL	CPHA		
								rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Field	Type	Reset	Description
31:7	Reserved			always read as 0.
6	CPHASEL	rw	0x00	CPHA polarity invert select 1: CPHA is configured as inverted. When CPHA =1, the first data bit capture starts on the second clock edge. When CPHA =0, the first data bit capture starts on the first clock edge. 0: CPHA configuration is not changed.
5	TXEDGE	rw	0x00	Transmit data edge select (Slave) 1: Tx data is sent to the data bus immediately (suitable for the high-speed mode, SPBRG = 4). 0: Tx data is sent to the data bus after one effective clock edge (suitable for the low-speed mode, SPBRG > 4).
4	RXEDGE	rw	0x00	Receive data edge select (Master) 1: Data is captured on the end edge of clock in the Tx data bits (suitable for the high-speed mode). 0: Data is captured in the middle of the Tx data bits.

Bit	Field	Type	Reset	Description
3	SPILEN	rw	0x01	SPI character length bit This bit is enabled after the DW8_32 is set (DW8_32 = 0). 1: 8 characters (default) 0: 7 characters

2	LSBFE	rw	0x00	LSBFE: LSB first enable bit 1: least significant bit first in the data Tx or Rx 0: most significant bit first in the data Tx or Rx
1	CPOL	rw	0x00	Clock polarity select bit 1: clock has a high level when idle (between two transmissions). 0: clock has a low level when idle (between two transmissions).
0	CPHA	rw	0x00	Clock phase select bit 1: the first data bit capture starts on the first clock edge. 0: the first data bit capture starts on the second clock edge.

#### 22.4.9 Baud rate generator (SPI\_SPBRG)

Address offset: 0x20

Reset value: 0x0000 0002

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

Reserved

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

SPBRG

rw rw

Bit	Field	Type	Reset	Description
31:16	Reserved			always read as 0.
15:0	SPBRG	rw	0x0002	SPI baud rate control register for baud rate Baud rate formula: $f_{\text{pclk}} = f_{\text{pclk}} / \text{SPBRG}$ ( $f_{\text{pclk}}$ is the APB clock frequency) Note: Never write '0' or '1' to this register.

#### 22.4.10 Receive data number register (SPI\_RXDNR)

Address offset: 0x24

Reset value: 0x0000 0001

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

Reserved

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

RXDNR

Serial Peripheral Interface (SPI)										UM_MM32F013x_Ver1.00							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset		Description												
31:16	Reserved		always read as 0.														
15:0	RXDNR	rw	0x0001		The register is used to hold a count of to be received bytes in next receive process. The value of this register is valid when the SPI is in the Master receive mode. The default value is 1. It can be written by MCU. Note: Never write '0' to this register.												

#### 22.4.11 Slave chip select register (SPI\_NSSR)

Address offset: 0x28

Reset value: 0x0000 00FF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved								NSS									
								rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description													
15:8	Reserved																
7:0	NSS	rw	0xFF	Chip select output signal in Master mode. Valid in the lower bit, invalid in the Slave mode 0: slave is selected 1: slave is not selected													

#### 22.4.12 Data control register (SPI\_EXTCTL)

Address offset: 0x2C

Reset value: 0x0000 0008

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

Bit	Field	Type	Reset	Description													
31:5	Reserved		always read as 0.														

4:0	EXTLEN	rw	0x08	Control the SPI character length 0 0000: 32 bits 0 0001: 1 bits 0 0010: 2 bits 0 0011: 3 bits ... 1 1100: 28 bits 1 1101: 29 bits 1 1110: 30 bits 1 1111: 31 bits  Note: This bit is valid only when the DW8_32 bit is set to '0' in the SPI_GCTL register, LSBFE bit as well as SPILEN bit set to '1' in the SPI_CCTL register.
-----	--------	----	------	---

# 23 | Inter-Integrated Circuit (I2C) Interface

## Inter-Integrated Circuit (I2C) Interface

### 23.1 I2C introduction

I2C (inter-integrated circuit) bus interface serves as an interface between the microcontroller and the serial I2C bus. It provides multimaster capability, and controls all I2C bus-specific sequencing, protocol, arbitration and timing.

The I2C bus is a two-wire serial interface, where the serial data line (SDA) and the serial clock line (SCL) pass information between the devices connected to the bus. Each device is addressable by a unique address and can operate as a transmitter or receiver. Besides a transmitter or receiver, the device can also be considered as a master or slave when it conducts data transfer. A master is a device that initializes the data transfer on the bus and generates transportable clock signals. At this point, any addressable device is considered as a slave.

I2C can operate in the Standard mode (data transfer rate between 0 to 100Kbps) and Fast mode (data transfer rate up to 400Kbps).

### 23.2 I2C main features

- Parallel-bus I2C bus protocol converter
- Half-duplex synchronous operation
- Master and Slave modes supported
- 7-bit/10-bit address supported
- Standard mode (100Kbps) and Fast mode (400Kbps) supported
- Start, Stop, and Restart generation, alongside Acknowledge signal detection
- A sole master supported in the Master mode
- Respective 2-byte Tx and Rx FIFOs
- Spike-free circuit added in the SCLI and SDAL
- Supports DMA operation
- Interrupt and query operations supported
- I2C slave multi-addressing capability supported (see the register description for details)

### 23.3 I2C protocol

### 23.3.1 Start and Stop conditions

When the bus is in the idle state, SCL and SDA are pulled up by an external pull-up resistor simultaneously. The data transfer initialized in the Master begins with a Start condition. A high to low transition on the SDA line while SCL is high defines a Start condition. The data transfer in the Master is terminated by the Master sending a Stop condition. A low to high transition on the SDA line while SCL is high defines a Stop condition. A timing diagram of Start and Stop conditions is shown as below. SDA must maintain steady during the data transfer when  $SCL = 1$ .

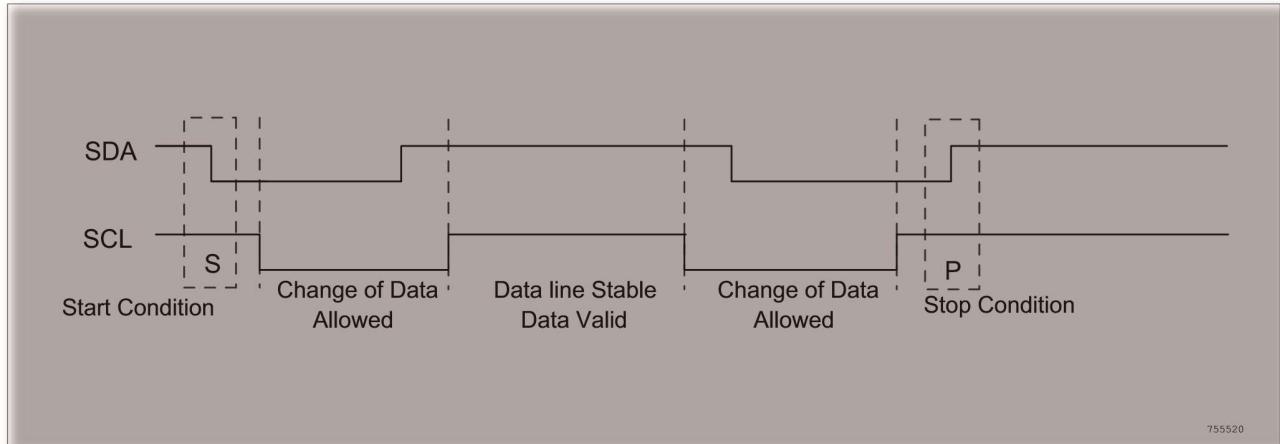


Figure 223. Start and Stop conditions

### 23.3.2 Slave addressing protocol

I2C has two address formats: 7-bit address format and 10-bit address format.

#### 7-bit address format

The following figure shows the slave address is formed from the first 7 bits (bit 7:1) of one byte sent following a Start condition (S). The least significant bit (bit 0) determines the direction of the message. If bit 0 = 0, then the Master writes data to the Slave; if bit 0 = 1, then the Master reads data from the Slave.

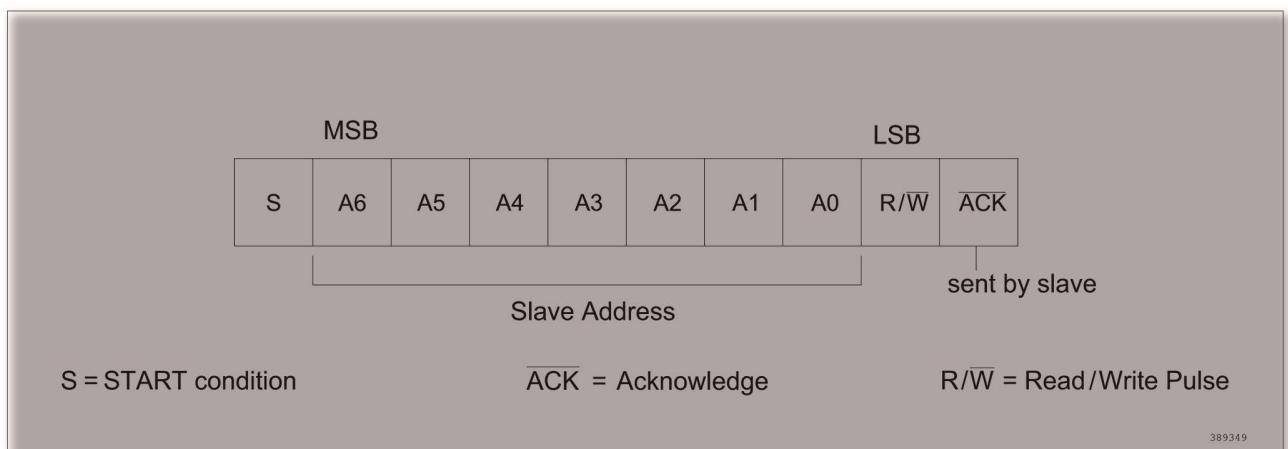


Figure 224. 7-bit address format

## 10-bit address format

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves address bits 9:8, and the LSB bit (bit 0) determines the direction of the message (R/W). The second byte transferred sets lower eight bits of the 10-bit address.

For details, see the diagram below:

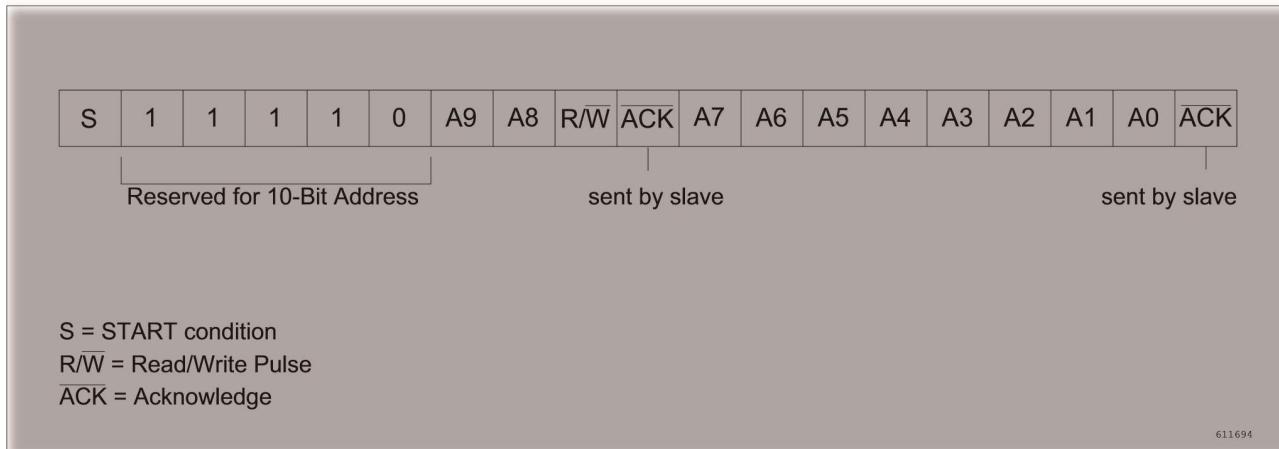


Figure 225. 10-bit address format

The following table defines the special purposes and reserved addresses of first bytes for I2C:

Table 66. First bytes for I2C

Slave Address	R/W Bit	Description
0000 000	0	General call address. I2C places the data in the receive buffer and issues a general call interrupt.
0000 000	1	Start byte
0000 001	X	CBUS address. I2C interface ignores these accesses.
0000 010	X	Reserved
0000 011	X	Reserved
0000 1xx	X	Reserved
1111 1xx	X	Reserved
1111 0xx	X	10-bit slave addressing

### 23.3.3 Transmitting and receiving protocol

The master can initiate data transmission and reception to or from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests from the master to either transmit data or receive data, acting as either a slave-transmitter or slave-receiver.

#### Master transmit and Slave receive

All data is transmitted in byte format, with no limit on the number of bytes per data transfer. After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a Stop condition. If a slave

is not able to respond, the slave must leave the SDA line high so that the master can abort the transfer.

If the master-transmitter is transmitting data as shown in the following figure, then the slave-receiver responds to the master-transmitter with an ACK pulse after every byte of data is received.

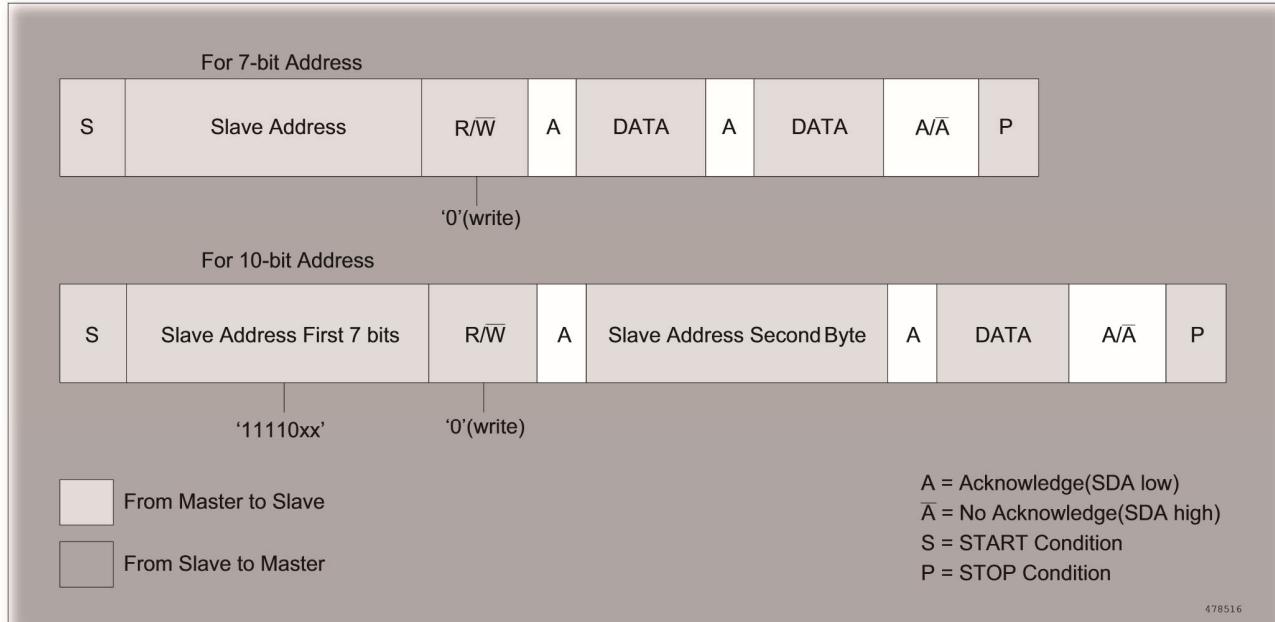


Figure 226. Master transmit protocol

### Master receive and Slave transmit

If the master is receiving data as shown in the following figure, then the master responds to the slave-transmitter after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) bit so that the master can issue a Stop condition.

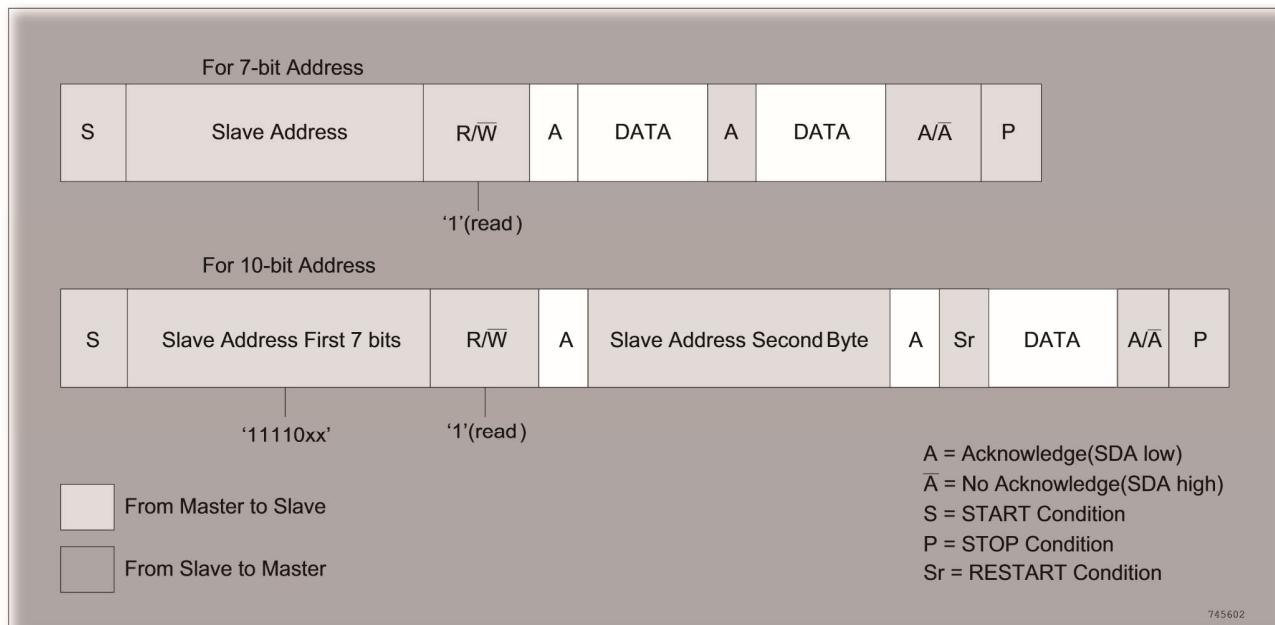


Figure 227. Master receive protocol

When a master does not want to relinquish the bus with a Stop condition, the master can issue a Restart condition. This is identical to a Start condition except it occurs after the ACK. Operating in Master mode, the I2C interface can then communicate with the same slave using a transfer of a different direction.

### Start Byte transfer protocol

The Start Byte transfer protocol is set up for systems that do not have a dedicated I2C hardware module. When I2C module is set as a master, it supports the generation of Start Byte transfers at the beginning of every transfer in case a slave device requires it.

This protocol consists of seven zeros followed by a 1, as illustrated in the following figure. This allows the processor that is polling the bus to under-sample the address phase until the processor detects a 0. Once the processor detects a 0, it switches from the under sampling rate to the correct rate of the master.

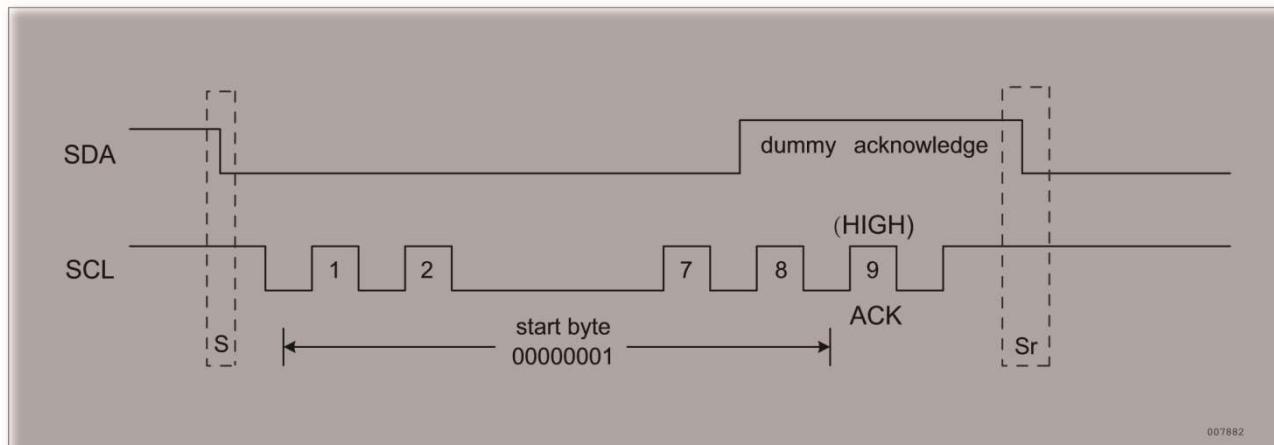


Figure 228. Start Byte transfer

The Start Byte has the following procedure:

1. Master generates a Start condition
2. Master transmits the Start byte (0000 0001)
3. Master transmits the ACK clock pulse (ACK)
4. No slave responds with an ACK pulse
5. Master generates a Restart condition (RESTART)

A hardware I2C receiver does not respond to the Start Byte because it is a reserved address and resets after the Restart condition is generated.

#### 23.3.4 Tx buffer management and Start, Stop, and Restart generation

Operating in Master mode, I2C module generates a Stop condition once the Tx buffer is empty. If the RESTART generation function is enabled (RESTART = 1), then a Restart condition is generated when the transfer direction switches from read/write to write/read. If the Restart condition is disabled, then a Start condition is generated following a Stop condition.

The figure below shows bits of a DR register.

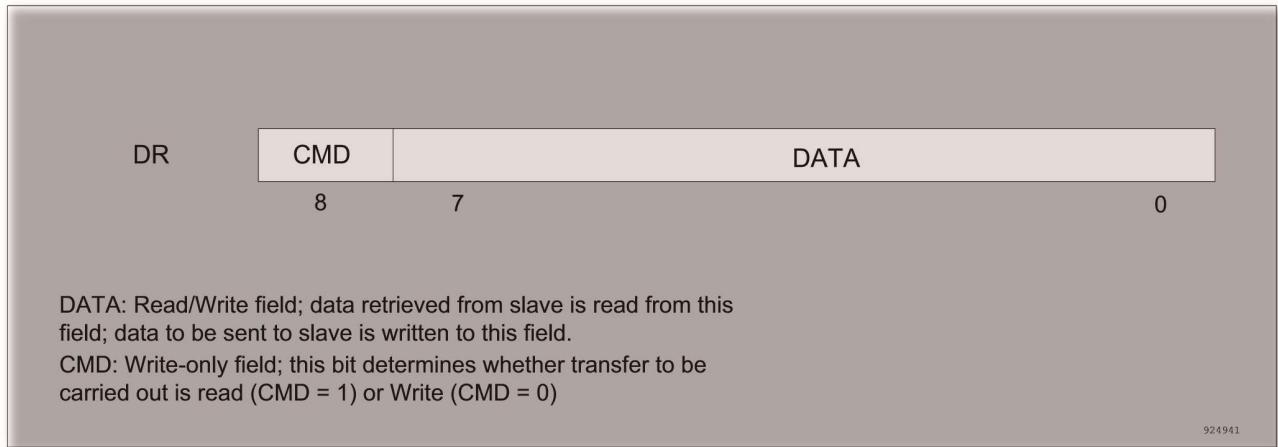


Figure 229. DR register

The following timing diagram illustrates the behavior of I2C module operating in the master transmit mode when Tx FIFO becomes empty.

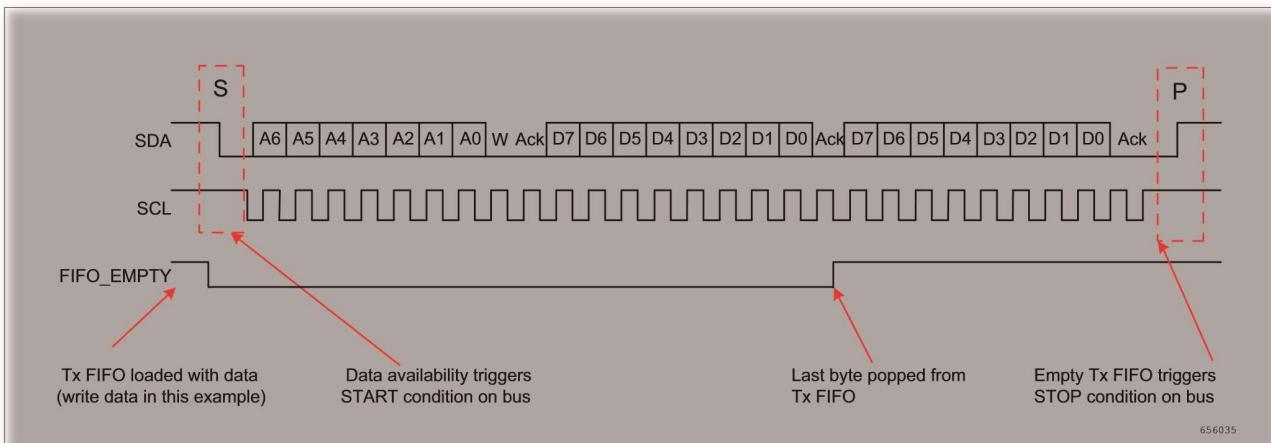


Figure 230. Master transmit - empty Tx FIFO

The following timing diagram illustrates the behavior of I2C module operating in the master receive mode when Tx FIFO becomes empty.

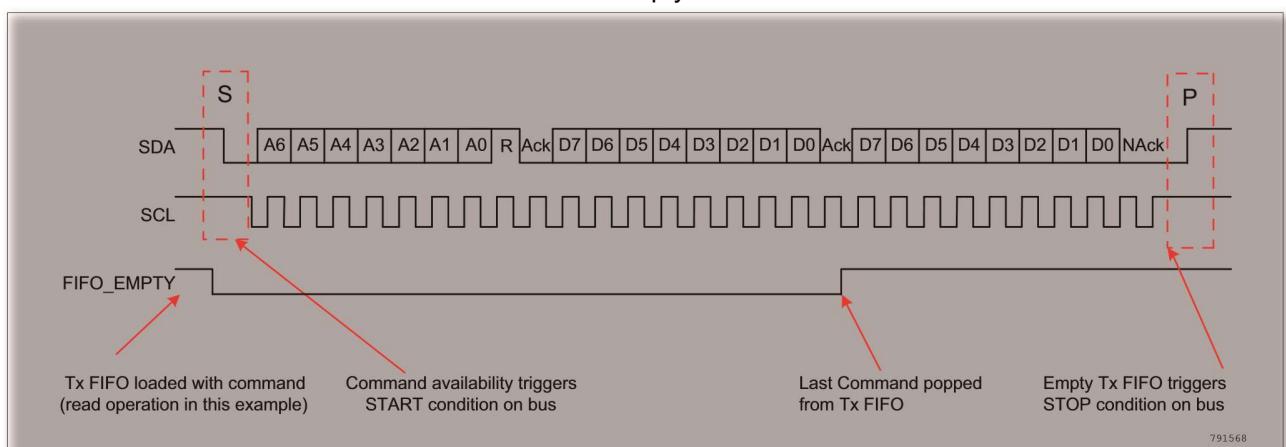


Figure 231. Master receive - empty Tx FIFO

### 23.3.5 Multiple master arbitration

The I2C bus allows multiple masters to reside on the same bus. If there are several masters on the same I2C-bus, there is an arbitration procedure if they try to take control of the bus at

the same time. Through the arbitration, only one master can take control of the bus with intact messages. Once a master has control of the bus, no other master can take control until the first master sends a Stop condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is in the high level. If there are two or more masters trying to send messages to the bus, the master, which transmits a 1 while the other master transmits 0, loses arbitration. The master that lost arbitration can continue to generate clock pulses until the end of the byte transfer. If all masters are addressing the same device, the arbitration could go into the data phase.

Upon detecting that it has lost arbitration, the I2C interface stops generating SCL signals.

The following figure illustrates the timing of two masters arbitrating on the bus.

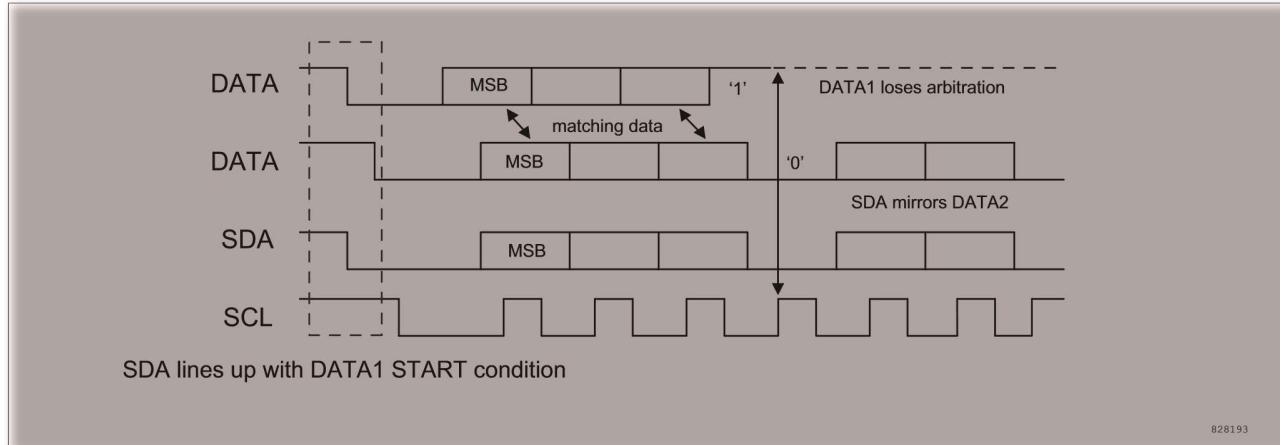


Figure 232. Multiple master arbitration

### 23.3.6 Clock synchronization

When two or more masters try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All masters generate their own clocks to transfer messages. Data is only valid during the high period of clock. Clock synchronization is performed using the wired-AND connection to the SCL signal. When the master transitions the SCL clock to 0, the master starts counting the low time of the SCL clock and transitions the SCL clock to 1 at the beginning of the next clock period. However, if another master is holding the SCL line to 0, then the master goes into a wait state until the SCL clock transitions to 1.

All masters then count off their high time, and the master with the shortest high time transitions the SCL to 0. The masters then counts out their low times and the one with the longest low time forces the other masters into a wait state. Therefore, a synchronized SCL clock is generated, which is illustrated in the following figure.

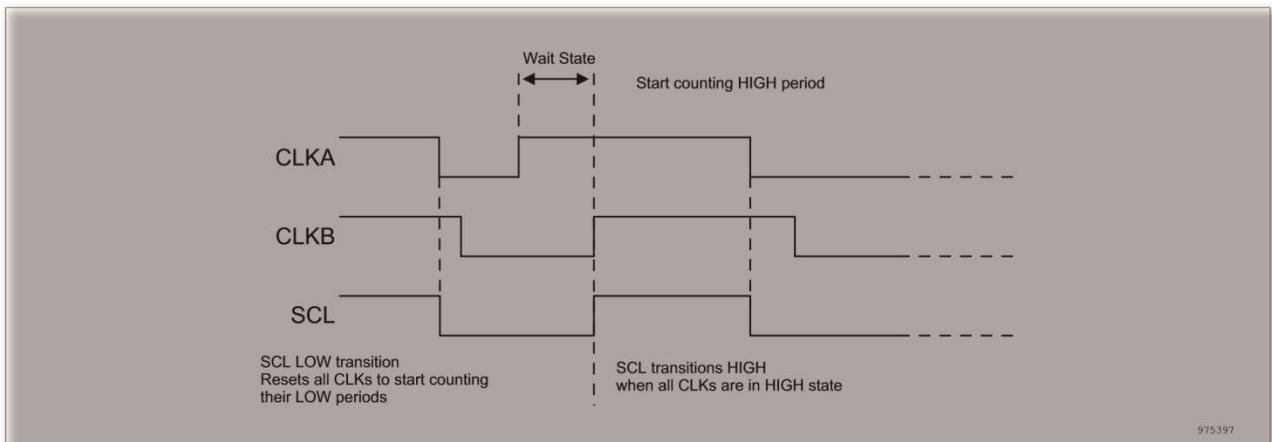


Figure 233. Multiple master clock synchronization

### 23.3.7 SCL configuration

For I2C level configuration, refer to the following:

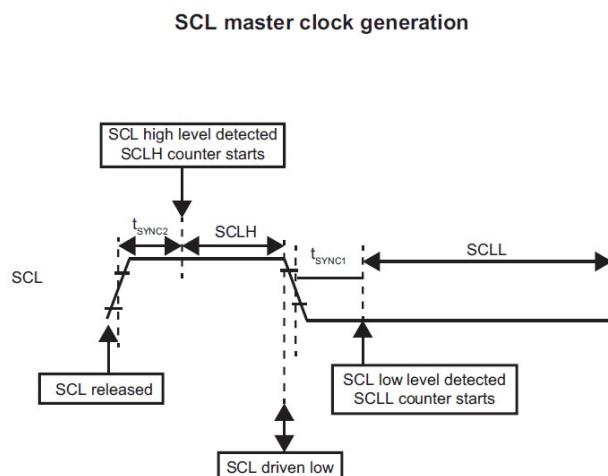


Figure 234. SCL master clock generation

$$SCLH = (SSHR + 12) \times I2CCLK + t_{syn1}$$

$$SCLL = (SSLR + 1) \times I2CCLK + t_{syn2}$$

Note: t<sub>syn1</sub> is equal to 0 ~ 1 I2CCLK and t<sub>syn2</sub> is equal to 0 ~ 1 I2CCLK

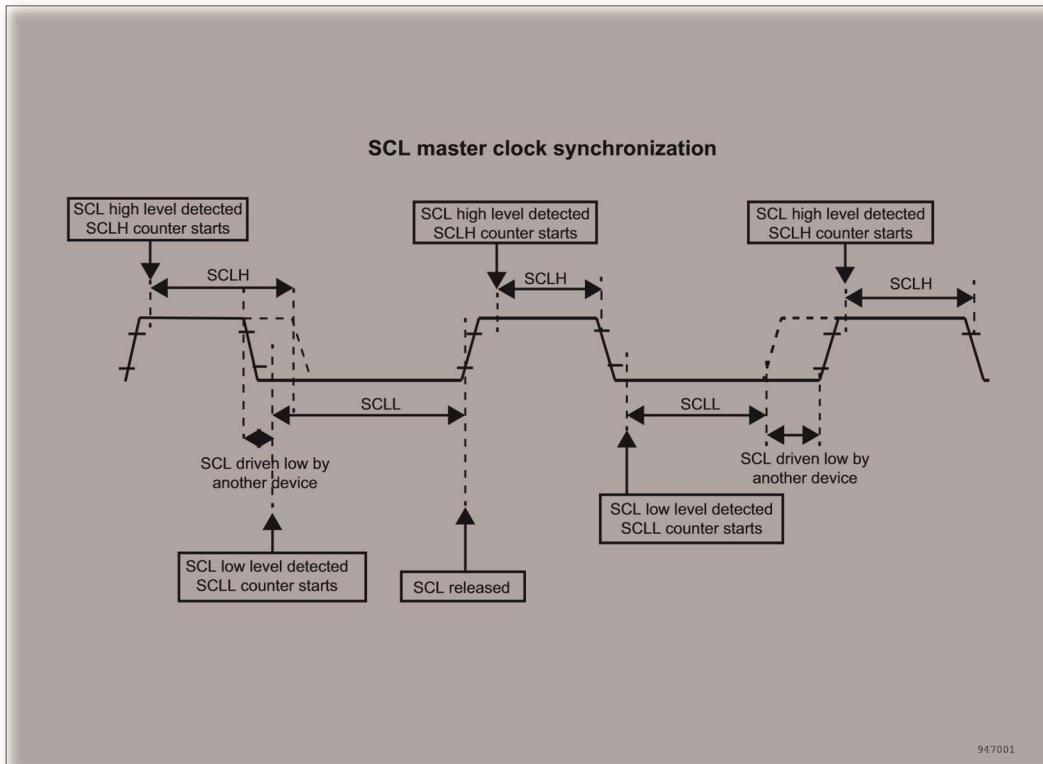


Figure 235. SCL master clock synchronization

$$SCLH = (FSHR + 12) \times I2CCLK + tsyn1$$

$$SCLL = (FSLR + 1) \times I2CCLK + tsyn2$$

Note:  $tsyn1$  is equal to  $2 \sim 3$  I2CCLK and  $tsyn2$  is equal to  $2 \sim 3$  I2CCLK.

## 23.4 I2C operation mode

I2C interface can operate in one of the four following modes:

- Slave transmitter mode
- Slave receiver mode
- Master transmitter mode
- Master receiver mode

Note: I2C interface module can only operate in the Master or Slave mode but it cannot operate in both modes at the same time. Therefore, make sure that the bit 6 (DISSLAVE) and bit 0 (MASTER) in the CR register are not set to 0 and 1 (or 1 and 0 respectively).

The I2C functional block diagram is shown as below:

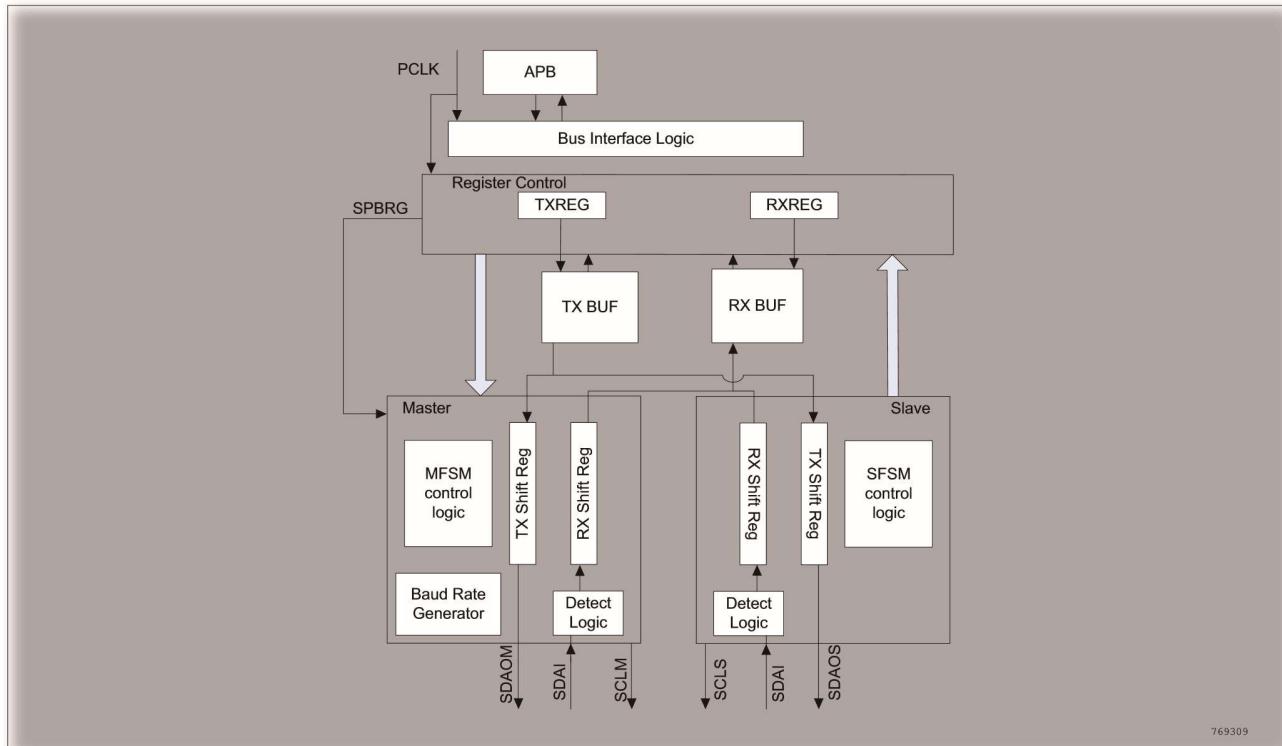


Figure 236. I2C function block diagram

### 23.4.1 Slave mode

The program flow chart of the Slave mode is described as follows:

#### Initial configuration

1. Disable I2C by writing a 0 to bit 0 of the ENR register.
2. Initialize the SAR register to set the slave address. This is the address to which the I2C interface responds.
3. Set the CR register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Write a 0 into bit 6 (DISSLAVE) and a 0 to bit 0 (MASTER) in the CR register.
4. Enable the I2C interface module by writing a 1 to bit 0 of the ENR register.

#### Slave-transmitter operation for a single byte

When another I2C master device on the bus addresses the I2C interface and requests data, the I2C interface acts as a slave-transmitter and the following steps occur:

1. Another I2C master device initiates an I2C transfer with an sending address that matches the slave address in the SAR register.
2. The I2C interface acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.
3. The I2C interface asserts the RD\_REQ interrupt (bit 5 of the RAWISR register) and pulls down the SCL line. The bus stays in the wait state until the software responses.

If the RD\_REQ interrupt has been masked (register IMR[5] = 0), then it is recommended that CPU does periodic query of the RAWISR register.

1. Setting bit 5 of the RAWISR register to 1 can be treated as the equivalent of the RD\_REQ interrupt being asserted.
2. Software must then act to satisfy the I2C transfer requirements.
3. The timing intervals are generally 10 times the SCL clock period. For example, for 400Kbps, the timing interval is 25us.

4. If there is any data remaining in the Tx FIFO before receiving the read request, the I2C interface asserts a TX\_ABRT interrupt (RAWISR[6]) to flush the old data from the Tx FIFO.
5. Software writes data to the DR register and writes a 0 in bit 8.
6. Software must clear the RD\_REQ and TX\_ABRT interrupts (bit 5 and bit 6) of the RAWISR register.
7. The I2C interface relinquishes SCL and transmits the byte.
8. The master controls the I2C bus by issuing a Restart condition or releases the bus by issuing a Stop condition.

### Slave-receiver operation for a single byte

When another master device addresses the I2C interface and sends data, the I2C interface acts as a slave-receiver and the following steps occur:

1. Another I2C master device initiates an I2C transfer with a sending address that matches the slave address in the SAR register.
2. The I2C interface acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-receiver.
3. I2C interface receives the transmitted data from master and places it in the receive buffer.
4. I2C interface asserts the RX\_FULL interrupt (RAWISR[2]). If the RX\_FULL interrupt has been masked (IMR[2] = 0), then it is recommended that the CPU does periodic query of the SR register. Reads of the SR register, with bit 3 (RFNE) set at 1, must then be treated as the equivalent of the RX\_FULL interrupt being asserted.
5. Software may read the byte from the DR register (bits 7:0) to get the data that was received.
6. The master controls the I2C bus by issuing a Restart condition or releases the bus by issuing a Stop condition.

### Slave-transfer operation for bulk transfers

In the standard I2C protocol, all transactions are single byte transactions and the program responds to a master read request by writing one byte into the slave's Tx FIFO. When a slave (slave-transmitter) receives a read request (RD\_REQ) from the master (master-receiver), there should be at least one entry placed into the slave-transmitter's Tx FIFO. The I2C interface module is designed to handle more data in the Tx FIFO so that subsequent read requests can receive that data without raising an interrupt. Ultimately, this eliminates latencies significantly being incurred by the interrupt for data each time.

This mode only occurs when I2C interface is acting as a slave-transmitter. If the master acknowledges the data sent by the slave-transmitter and there is no data in the slave's TX FIFO, the I2C interface will pull down the SCL line of I2C bus until the read request interrupt (RD\_REQ) is generated; moreover, it will not relinquish the SCL line unless the data in the TX FIFO is prepared.

If the RX\_REQ interrupt has been masked (ISR[5] = 0), then it is recommended that the software does periodic query/reads of the RAWISR register. Reads of RAWISR[5] that return 1 must be treated as the equivalent of the RX\_REQ interrupt.

The RD\_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the TX FIFO. During the transmission of these bytes to the master, if the master acknowledges the last byte, then the slave must raise the RD\_REQ interrupt request again because the master is requesting for more data.

If the master is to receive n bytes from the I2C interface but the program wrote a number of bytes larger than n to the Tx FIFO, then when the slave finishes sending the requested n bytes, it clears the Tx FIFO and ignores any excess bytes.

### 23.4.2 Master mode

#### Initial configuration

1. Disable the I2C interface by setting ENR[0] = 0.
2. Set the I2C operation speed mode (Stand mode or Fast mode) by configuring bit 2:1 in the CR register. Meanwhile, make sure bit 6(DISSLAVE) = 1 and bit 0(MASTER) = 1.
3. Write to the TAR register the address of the I2C device. Set this register so that a general call address or a Start Byte command can be configured.
4. Enable the I2C interface by writing a 1 in ENR[0].
5. Now write the transfer direction and data to be sent to the DR register. If the DR register is written before the I2C interface is enabled, the data and commands are lost as the buffers are kept cleared when the I2C interface is not enabled. Through these steps, I2C interface will generate a Start condition and send the address byte data to the I2C bus.

#### Master transmit and Master receive

The I2C interface supports switching back and forth between reading and writing dynamically. When transmitting data, write the data to be written to the lower byte of the I2C Rx/Tx data buffer and Command register (DR). The CMD bit should be written to 0 for write operations. Subsequently, a read command does not require the configuration of lower byte of the DR register, and a 1 should be written to the CMD bit. If the sent FIFO is empty, the I2C module pulls down SCL until the next command is written into the sent FIFO.

#### Program flow chart

The following flow chart shows a programming model when I2C interface is set as a master:

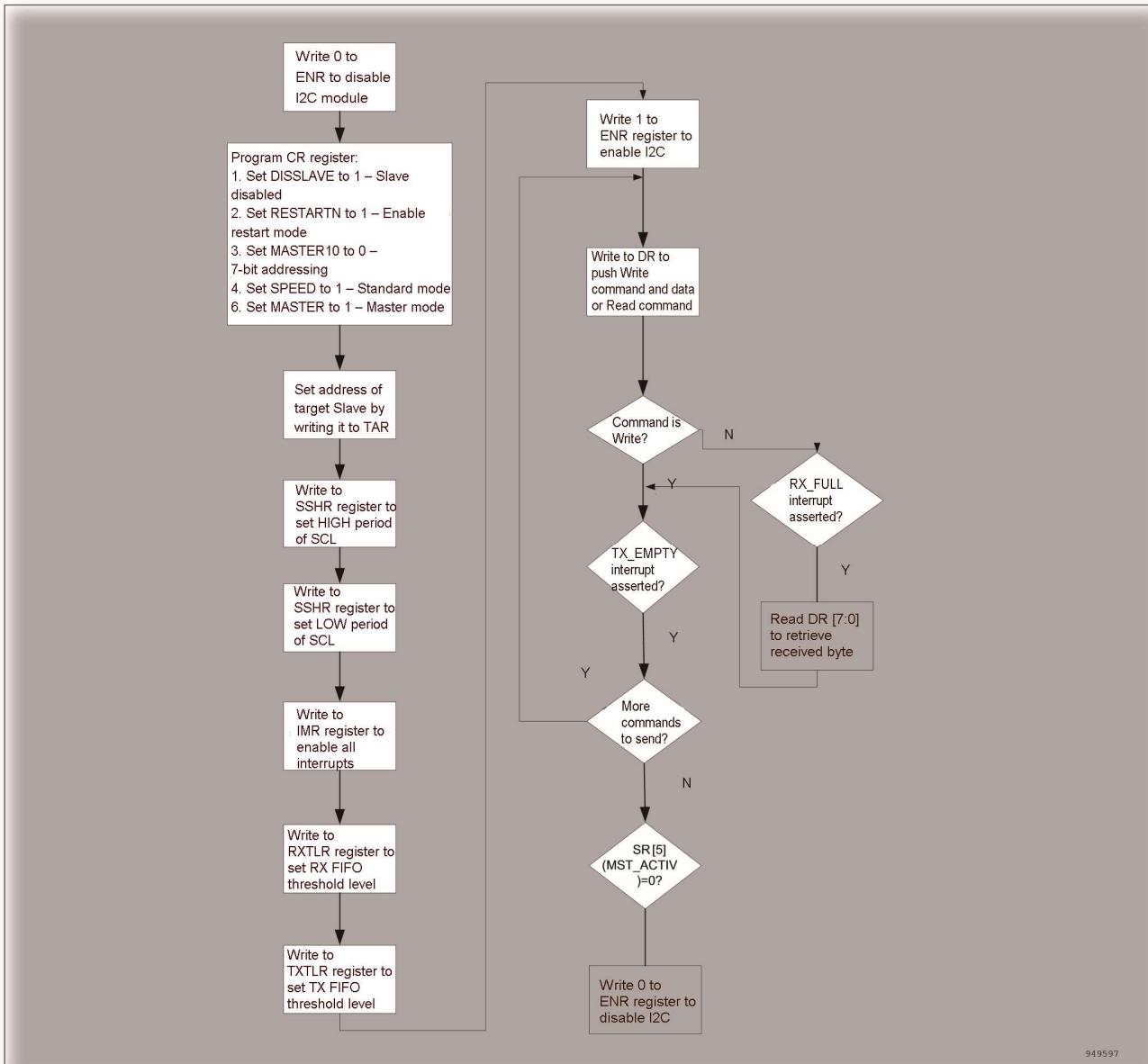


Figure 237. I2C interface master flow chart

### 23.4.3 I2C abort transfer

The ABRT control bit of the ENR register allows the software to relinquish the I2C bus before completing the issued transfer commands from the Tx FIFO.

In response to an ABORT request, the I2C module issues the Stop condition over the I2C bus, alongside Tx FIFO flush. Aborting the transfer is allowed only in master mode of operation.

#### Procedure

1. Stop filling the Tx FIFO (DR) with new commands.
2. When operating in DMA mode, disable the transmit DMA by setting TDMAE to 0.
3. Set ABRT bit of the ENR register to 1.
4. Wait for the TX\_ABRT interrupt.

## 23.5 Communication using DMA

I2C interface supports DMA requests for data transmission and reception. DMA transmission or reception can be enabled independently by setting a corresponding bit in the DMA register. DMA requests are generated by Data register becoming empty in transmission and Data register becoming full in reception. The DMA request must be served before the end of the current byte transfer.

### Transmission using DMA

DMA mode can be enabled for transmission by setting the TXEN bit in the DMA register. Data will be loaded from a Memory area configured using the DMA controller to the DR register whenever a data byte is sent after the DMA channel is distributed for I2C.

### Reception using DMA

DMA mode can be enabled for reception by setting the RDMAE bit in the DMA register. Data will be loaded from the DR register to a Memory area configured using the DMA controller whenever a data byte is received after the DMA channel is distributed for I2C.

## 23.6 I2C interrupts

The table below gives the list of I2C interrupt bits, as well as the way to set and clear them. Some bits are set by hardware and cleared by software, others are set and cleared by hardware.

Table 67. Setting and clearing the interrupt bits

Interrupt bit	Set by hardware/Cleared by software	Set and cleared by hardware
GEN_CALL	√	✗
START_DET	√	✗
STOP_DET	√	✗
ACTIVITY	√	✗
RX_DONE	√	✗
TX_ABRT	√	✗
RD_REQ	√	✗
TX_EMPTY	✗	√
TX_OVER	√	✗
RX_FULL	✗	√
RX_OVER	√	✗
RX_UNDER	√	✗

The figure below describes the interrupt bit is set by hardware and cleared by software in the interrupt register.

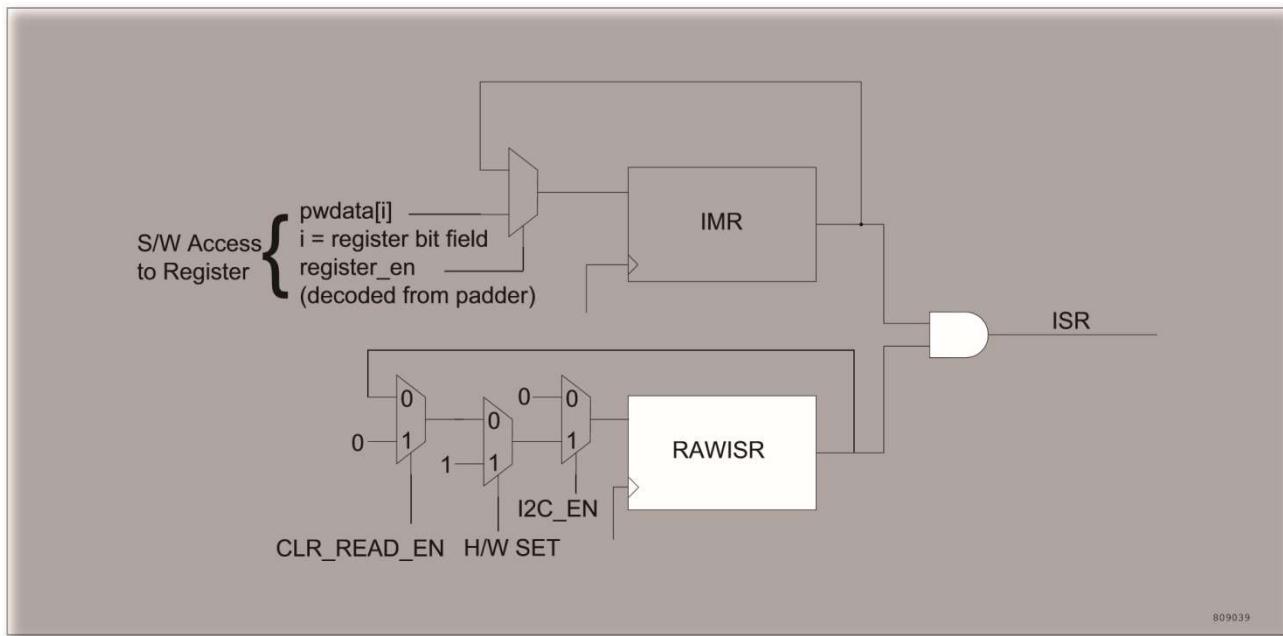


Figure 238. Interrupt mechanism

## 23.7 I2C register description

Table 68. Overview of I2C register description

Offset	Acronym	Register Name	Reset	Section
0x00	I2C_CR	I2C control register	0x0000007F	Section 23.7.1
0x04	I2C_TAR	I2C target address register	0x00000055	Section 23.7.2
0x08	I2C_SAR	I2C slave address register	0x00000055	Section 23.7.3
0x10	I2C_DR	I2C data command register	0x00000001	Section 23.7.4
0x14	I2C_SSHR	Standard mode I2C clock SCL high count register	0x00000190	Section 23.7.5
0x18	I2C_SSLR	Standard mode I2C clock SCL low count register	0x000001D6	Section 23.7.6
0x1C	I2C_FSHR	Fast mode I2C clock SCL high count register	0x0000003C	Section 23.7.7
0x20	I2C_FSLR	Fast mode I2C clock SCL low count register	0x00000082	Section 23.7.8
0x2C	I2C_ISR	I2C interrupt status register	0x00000000	Section 23.7.9
0x30	I2C_IMR	I2C interrupt mask register	0x000008FF	Section 23.7.10
0x34	I2C_RAWISR	I2C RAW interrupt status register	0x00000000	Section 23.7.11
0x38	I2C_RXTLR	I2C receive threshold register	0x00000000	Section 23.7.12
0x3C	I2C_TXTLR	I2C transmit threshold register	0x00000000	Section 23.7.13

Offset	Acronym	Register Name	Reset	Section
0x40	I2C_ICR	I2C combined and independent interrupt clear register	0x00000000	Section 23.7.14
0x44	I2C_RX_UNDER	I2C RX_UNDER interrupt clear register	0x00000000	Section 23.7.15
0x48	I2C_RX_OVER	I2C RX_OVER interrupt clear register	0x00000000	Section 23.7.16
0x4C	I2C_TX_OVER	I2C TX_OVER interrupt clear register	0x00000000	Section 23.7.17
0x50	I2C_RD_REQ	I2C RD_REQ interrupt clear register	0x00000000	Section 23.7.18
0x54	I2C_TX_ABRT	I2C TX_ABRT interrupt clear register	0x00000000	Section 23.7.19
0x58	I2C_RX_DONE	I2C RX_DONE interrupt clear register	0x00000000	Section 23.7.20
0x5C	I2C_ACTIV	I2C ACTIVITY interrupt clear register	0x00000000	Section 23.7.21
0x60	I2C_STOP	I2C STOP_DET interrupt clear register	0x00000000	Section 23.7.22
0x64	I2C_START	I2C START_DET interrupt clear register	0x00000000	Section 23.7.23
0x68	I2C_GC	I2C GEN_CALL interrupt clear register	0x00000000	Section 23.7.24
0x6C	I2C_ENR	I2C enable register	0x00000000	Section 23.7.25
0x70	I2C_SR	I2C status register	0x00000006	Section 23.7.26
0x74	I2C_TXFLR	I2C transmit FIFO level register	0x00000000	Section 23.7.27
0x78	I2C_RXFLR	I2C receive FIFO level register	0x00000000	Section 23.7.28
0x7C	I2C_HOLD	I2C SDA hold time register	0x00000001	Section 23.7.29
0x88	I2C_DMA	I2C DMA control register	0x00000000	Section 23.7.30
0x94	I2C_SETUP	I2C SDA setup time register	0x00000064	Section 23.7.31
0x98	I2C_GCR	I2C general call ACK register	0x00000001	Section 23.7.32
0xB0	I2C_SLVMASK	I2C slave address mask register	0x000003FF	Section 23.7.33
0xB4	I2C_SLVRCVADDR	I2C slave receive address register	0x00000000	Section 23.7.34

### 23.7.1 I2C control register (I2C\_CR)

Address offset: 0x00

Reset value: 0x007F

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SLV_TX_ABRT_DIS	RESTART	STOPEMPINT		STOPINT	DISSLAVEREPE		ASTER10	SLAVE10	SPEED	MASTER				

Bit	F ield	T ype	Reset	Description
15:12	Reserved			always read as 0.
11	SLV_TX_ ABRT_ DIS	rw	0x00	<p>When I2C is read as Slave</p> <p>1: Clearing TXFIFO 0 after the reception of RD_REQ signal is disabled</p> <p>0: Clearing TXFIFO 0 after the reception of RD_REQ signal is enabled</p>
10	RESTART	rw	0x00	<p>This bit controls whether a RESTART is issued before transmission or reception.</p> <p>It is valid only when the IC_EMPTYFIFO_HOLD_MASTER_EN is set to '1'.</p> <p>1: If RESTART is '1', a RESTART signal is issued before the data is sent/received (according to the value of CMD), regardless of whether or not the transfer direction is changing from the previous command. If RESTART is '0', a START is issued instead, followed by a STOP signal.</p> <p>0: If RESTART is 1, a RESTART is issued only if the transfer direction is changing from the previous command. If RESTART is '0', a START is issued instead, followed by a STOP signal.</p>
9	STOP	rw	0x00	<p>STOP: This bit controls whether a STOP is issued after transmission or reception.</p> <p>It is valid only when the IC_EMPTYFIFO_HOLD_MASTER_EN is set to '1'.</p> <p>1: STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus.</p> <p>0: STOP is not issued after this byte, regardless of whether or not the Tx FIFO is empty. The master continues the current transfer by sending/receiving data bytes according to the value of the CMD bit. If the Tx FIFO is empty, the master holds the SCL low and stalls the bus until Tx FIFO receives new data.</p>
8	EMPINT	rw	0x00	This bit controls whether a TX_EMPTY interrupt is generated. For details, refer to the RAWISR register.
7	STOPINT	rw	0x00	<p>This bit controls whether a STOP interrupt is generated in the Slave mode.</p> <p>1: STOP interrupt is generated only when the address is matched.</p> <p>0: STOP interrupt is generated regardless of whether or not the address is matched. This bit is only applicable in the Slave mode.</p> <p>Note: During general call addressing, if this bit is set, the slave doesn't generate a STOP interrupt.</p> <p>The STOP interrupt is generated only when the sent address is matched with the slave address.</p>

Bit	Field	Type	Reset	Description
6	DISSLAVE	rw	0x01	This bit controls whether I2C has its slave disabled. 0: slave enabled 1: slave disabled
5	REPEN	rw	0x01	This bit determines whether RESTART conditions may be sent when acting as a master. 0: disabled 1: enabled  When RESTART is disabled, I2C interface acting as a master is prohibited from performing the following functions: Send a Start Byte Change transfer direction in the combined format Read operation with a 10-bit address Replace Restart condition with a Stop and a subsequent Start condition. If the above operations are performed, it will result in setting 6(TX_ABRT) bit of RAWISR register to 1.
4	MASTER10	rw	0x01	Address mode when I2C acts as a master 0: 7-bit address mode 1: 10-bit address mode
3	SLAVE10	rw	0x01	When acting as a slave, this bit controls whether the I2C responds to 7- or 10-bit addresses 0: 7-bit addressing. The I2C interface ignores transactions that involve 10-bit addressing; for 7-bit addressing, only the lower 7 bits of the SAR register are compared. 1: 10-bit addressing. The I2C responds to only 10-bit addressing transfers that match the full 10 bits of the SAR register.
2:1	SPEED	rw	0x03	These bits control at which speed the I2C operates. The setting is valid only when the I2C operates in the Master mode. 1: Standard mode (0 ~ 100Kbps) 2: Fast mode (400Kbps)
0	MASTER	rw	0x01	This bit controls whether the I2C master is enabled. 0: master disabled 1: master enabled

DISSLAVE(bit 6) and MASTER(bit 0) are set according to the table below:

Table 69. DISSLAVE(bit 6) and MASTER(bit 0) settings

DISSLAVE CR[6]	MASTER CR[0]	Status
0	0	Slave device
0	1	Wrong configuration
1	0	Wrong configuration
1	1	Master device

### 23.7.2 I2C target address register (I2C\_TAR)

Address offset: 0x04

Reset value: 0x0055

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved			SPECIAL	GC	ADDR											
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:12	Reserved			always read as 0.
11	SPECIAL	rw	0x00	This bit indicates whether software performs a special command (General Call or Start Byte command). 0: Ignore bit 10(GC) and use ADDR bit normally. 1: Perform special I2C command as specified in GC bit.
10	GC	rw	0x00	If bit 11(SPECIAL) is set to 1, then this bit indicates whether a General Call or Start byte command is to be performed by the I2C. 0: General Call address After issuing a General Call, only writes may be performed. The I2C interface remains working in General Call mode until the SPECIAL(bit 11) is cleared. 1: START BYTE command
9:0	ADDR	rw	0x55	This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE command, the CPU needs to write only once into these bits.

### 23.7.3 I2C Slave address register (I2C\_SAR)

Address offset: 0x08

Reset value: 0x0055

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved					ADDR											
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description

15:10	Reserved			always read as 0
9:0	ADDR	rw	0x55	The bit holds the slave address when the I2C interface is operating as a slave. For 7-bit addressing, only ADDR [6:0] is valid.

### 23.7.4I2C Data command register (I2C\_DR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								CMD	DAT							
w	rw	rw	rw	rw	rw	rw	rw	rw								

Bit	Field	Type	Reset	Description
15:9	Reserved			always read as 0.
8	CMD	w	0x00	<p>This bit controls whether a read or a write is performed in the Master mode.</p> <p>1: Read</p> <p>0: Write</p> <p>When a command is entered in the TX FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, write operation to this bit is ignored. In slave-transmitter mode, a "0" indicates that data of DR register is to be transmitted.</p>
7:0	DAT	rw	0x00	This register contains the data to be transmitted or received on the I2C bus.

### 23.7.5Standard mode I2C clock SCL high count register (I2C\_SSHR)

Address offset: 0x14

Reset value: 0x0190

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

Bit	Field	Type	Reset	Description
15:0	CNT	rw	0x0190	<p>This register sets the SCL clock high-period for I2C interface in the Standard mode.</p> <p>Note: This register is programmed to a value between 6 and 65525, because I2C interface uses a 16-bit counter to flag an I2C bus idle condition when this counter reaches a value of SSHR + 10.</p>

### 23.7.6Standard mode I2C clock SCL low count register (I2C\_SSLR)

Address offset: 0x18

Reset value: 0x01D6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

CNT																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description												
15:0	CNT	rw	0x01D6	This register sets the minimum SCL clock low-period to 8 for I2C interface in the Standard mode.												

### 23.7.7Fast mode I2C clock SCL high count register (I2C\_FSHR)

Address offset: 0x1C

Reset value: 0x003C

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

CNT																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description												
15:0	CNT	rw	0x003C	This register sets the SCL clock high-period for I2C interface in the Fast mode. When I2C operates in the Standard mode, this register is read-only and returns 0. The minimum value is 6.												

### 23.7.8Fast mode I2C clock SCL low count register (I2C\_FSLR)

Address offset: 0x20

Reset value: 0x0082

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

CNT																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description												
15:0	CNT	rw	0x0082	This register sets the SCL clock low-period for I2C interface in the Fast mode. When I2C operates in the Standard mode, this register is read-only and returns 0. The minimum value is 8.												

### 23.7.9I2C interrupt status register (I2C\_ISR)

Address offset: 0x2C

Reset value: 0x0000

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

Reserved	GC	START	STOP	ACTIV	RX_DONE	TX_ABR	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER				
	r	r	r	r	r	r	r	r	r	r	r	r				r

Bit	Field	Type	Reset	Description
15:12	Reserved			always read as 0.
11:0	ISR	r	0x0000	Refer to RAWISR register for detailed descriptions of these bits.

### 23.7.10 I2C interrupt mask register (I2C\_IMR)

Address offset: 0x30

Reset value: 0x08FF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
					GC	START	STOP	ACTIV	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:12	Reserved			always read as 0
11:0	IMR	rw	0x08FF	These bits mask their corresponding ISR bits.

### 23.7.11 I2C RAW interrupt status register (I2C\_RAWISR)

Address offset: 0x34

Reset value: 0x0000

Unlike the ISR register, RAWISR is not masked.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
					GC	START	STOP	ACTIV	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
					r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15:12	Reserved			always read as 0
11	GC	r	0x00	General call Set when a General Call address is received.  It is cleared either by disabling I2C interface or when the CPU reads the GC register. I2C stores the received data in the Rx buffer.
10	START	r	0x00	Start condition detection  This bit is set to 1 once it detects that a START or RESTART condition has occurred on the I2C interface regardless of whether I2C interface is operating in Slave or Master mode.

Bit	Field	Type	Reset	Description

9	STOP	r	0x00	<p>Stop condition detection</p> <p>The status of this bit differs based on the STOPINT status in the CR register.</p> <p>When STOPINT = 0</p> <p>This bit is set to 1 once it detects that a STOP condition has occurred on the I2C interface regardless of whether I2C interface is operating in Slave or Master mode. In Slave mode, a STOP interrupt is generated irrespective of whether the address is matched or not.</p> <p>When STOPINT = 1</p> <p>In Master mode (MASTER = 1), this bit indicates whether a STOP condition has occurred on the I2C interface.</p> <p>In Slave mode (MASTER = 0), a STOP interrupt is generated only if the slave address is matched successfully.</p>
8	ACTIV	r	0x00	<p>I2C interface activity</p> <p>This bit captures I2C module activity and stays set until it is cleared by four ways:</p> <ul style="list-style-type: none"> <li>Disabling the I2C interface</li> <li>Reading the ACTIV register</li> <li>Reading the ICR register</li> <li>System reset</li> </ul> <p>Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the I2C is idle, this bit remains set until cleared.</p>
7	RX_DONE	r	0x00	<p>Slave transmit done</p> <p>When the I2C is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte.</p> <p>This occurs on the last byte of the transmission, indicating that the transmission is done.</p>
6	TX_ABRT	r	0x00	<p>Transmit abort</p> <p>This bit is set to 1 when I2C interface, as an I2C transmitter, is unable to complete the data transfer in the FIFO.</p> <p>Note: The I2C flushes the Tx FIFO and Rx FIFO when there is a transmit abort. The Tx FIFO remains in this flushed state until the TX_ABRT register is read. Once this read is performed, the Tx FIFO is then ready to accept new data from the APB bus.</p>

Bit	Field	Type	Reset	Description

5	RD_REQ	r	0x00	<p>Read request</p> <p>This bit is set to 1 when I2C is acting as a slave and another master is attempting to read data from the I2C interface.</p> <p>The I2C interface holds the bus in a wait state (SCL = 0) until this interrupt is serviced, which means that the I2C interface, as a slave, has been addressed by another master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the DR register. This bit is set to 0 just after the processor reads the RD_REQ register.</p>
4	TX_EMPTY	r	0x00	<p>Transmit buffer empty</p> <p>The status of this bit differs based on the EMPINT status in the CR register.</p> <p>If EMPINT = 0, this bit is set to 1 when the transmit buffer is at or below the threshold value</p> <p>If EMPINT = 1, this bit is set to 1 when the transmit buffer is at or below the threshold value and the most recent data transmission from the internal shift register is completed.</p> <p>It is automatically cleared by hardware when the transmit buffer is not empty.</p>
3	TX_OVER	r	0x00	<p>Transmit buffer over</p> <p>When the transmit buffer is full, the processor writes new data and causes overflow, which sets the bit to 1.</p>
2	RX_FULL	r	0x00	<p>Receive buffer not empty</p> <p>Set when the receive buffer is not empty.</p> <p>It is automatically cleared by hardware when the receive buffer is empty.</p>
1	RX_OVER	r	0x00	<p>Receive buffer over</p> <p>Set if the receive buffer is full and an additional byte is received. The I2C interface acknowledges this, but new data will be lost.</p>
0	RX_UNDER	r	0x00	<p>Receive buffer under</p> <p>Set to 1 if the processor attempts to read the DR register when RX FIFO is empty.</p>

### 23.7.12 I2C receive threshold register (I2C\_RXTLR)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TL							
		rw		rw		rw		rw		rw		rw		rw	

Bit	Field	Type	Reset	Description

15:8	Reserved			always read as 0
7:0	TL	rw	0x00	Receive FIFO threshold level This bit controls whether the RX_FULL interrupt is triggered.

### 23.7.13 I2C transmit threshold register (I2C\_TXTLR)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TL							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	TL	rw	0x00	Transmit FIFO threshold level This bit controls whether the TX_EMPTY interrupt is triggered.

### 23.7.14 I2C combined and independent interrupt clear register (I2C\_ICR)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ICR	
rc_r															

Bit	Field	Type	Reset	Description
15:1	Reserved			always read as 0.
0	ICR	rc_r	0x00	Reading this register clears all combined and independent interrupts. This bit only clears software clearable interrupts, instead of hardware clearable interrupts.

### 23.7.15 I2C RX\_UNDER interrupt clear register (I2C\_RX\_UNDER)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0	
Reserved														RX_UNDER	
rc_r															

Bit	Field	Type	Reset	Description
15:0				

15:1	Reserved			always read as 0
0	RX_UNDER	rc_r	0x00	Reading this register clears the RX_UNDER interrupt (RAWISR[0]).

### 23.7.16 I2C\_RX\_OVER interrupt clear register (I2C\_RX\_OVER)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0
Reserved													RX_OVER	rc_r

Bit	Field	Type	Reset	Description
15:1	Reserved			always read as 0
0	RX_OVER	rc_r	0x00	Reading this register clears the RX_OVER interrupt (RAWISR[1]).

### 23.7.17 I2C\_TX\_OVER interrupt clear register (I2C\_TX\_OVER)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0
Reserved													TX_OVER	rc_r

Bit	Field	Type	Reset	Description
15:1	Reserved			always read as 0
0	TX_OVER	rc_r	0x00	Reading this register clears the TX_OVER interrupt (RAW_ISR[3]).

### 23.7.18 I2C\_RD\_REQ interrupt clear register (I2C\_RD\_REQ)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0
Reserved													RD_REQ	rc_r

Bit	Field	Type	Reset	Description
15:1	Reserved			always read as 0
0	RD_REQ	rc_r	0x00	Reading this register clears the RD_REQ interrupt (RAW_ISR[5]).

### 23.7.19 I2C\_TX\_ABRT interrupt clear register (I2C\_TX\_ABRT)

Address offset: 0x54

Reset value: 0x0000

### Inter-integrated circuit (I2C) interface

15	14	13	12	11	10	9	8	7	6	5	4	3	2	UM_MM32F013x_Ver1.00	0
Reserved														TX_ABRT	
														rc_r	

Bit	Field	Type	Reset	Description
15:1	Reserved			always read as 0.
0	TX_ABRT	rc_r	0x00	Reading this register clears the TX_ABRT interrupt (RAWISR6]. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO.

### 23.7.20 I2C\_RX\_DONE interrupt clear register (I2C\_RX\_DONE)

Address offset: 0x58

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0	
Reserved														RX_DONE	
														rc_r	

Bit	Field	Type	Reset	Description
15:1	Reserved			always read as 0
0	RX_DONE	rc_r	0x00	Reading this register clears the RX_DONE interrupt (RAWISR[7]).

### 23.7.21 I2C ACTIVITY interrupt clear register (I2C\_ACTIV)

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved														ACTIV		
														rc_r		

Bit	Field	Type	Reset	Description
15:1	Reserved			always read as 0.
0	ACTIV	rc_r	0x00	Reading this register clears the ACTIV interrupt (RAWISR[8]) if the I2C bus is not active anymore.  If the I2C is still active, the ACTIV interrupt bit continues to be set. It is automatically cleared by hardware if the I2C module is disabled and if there is no further activity on the I2C bus. By reading this register, you get status of the ACTIV(bit 8) in the RAWISR register.

### 23.7.22 I2C\_STOP\_DET interrupt clear register (I2C\_STOP)

Address offset: 0x60

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---	--

Reserved											STOP
											rc_r
Bit	Field	Type	Reset	Description							
15:1	Reserved			always read as 0							

### 23.7.23 I2C START\_DET interrupt clear register (I2C\_START)

Address offset: 0x64

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														START	
														rc_r	

Bit	Field	Type	Reset	Description							
15:1	Reserved			always read as 0							
0	START	rc_r	0x00	Reading this register clears the START interrupt (RAWISR[10]).							

### 23.7.24 I2C GEN\_CALL interrupt clear register (I2C\_GC)

Address offset: 0x68

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														GC	
														rc_r	

Bit	Field	Type	Reset	Description							
15:1	Reserved			always read as 0							
0	GC	rc_r	0x00	Reading this register clears the GC interrupt (RAWISR[11]).							

### 23.7.25 I2C enable register (I2C\_ENR)

Address offset: 0x6C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	1	0	
Reserved														ABORT	ENABLE
														rw	rw

Bit	Field	Type	Reset	Description							
15:2	Reserved			always read as 0.							

1	ABORT	rw	0x00	I2C transfer abort 0: ABORT not initiated or ABORT done 1: ABORT operation in progress The software can abort the I2C transfer by setting this bit when the I2C module acts as a master. The software cannot clear the ABORT bit once set. In response to an ABORT, the I2C controller issues a STOP and flushes the Tx FIFO after completing the current transfer, then sets the TX_ABRT interrupt after the abort operation. The ABORT bit is cleared automatically after the abort operation.
0	ENABLE	rw	0x00	I2C mode enable 0: I2C module disabled (TX and RX FIFOs are held in an erased state) 1: I2C module enabled

### 23.7.26 I2C status register (I2C\_SR)

Address offset: 0x70

Reset value: 0x0006

This is a read-only register used to indicate the current transfer status and FIFO status. None of the bits in this register request an interrupt.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
									Reserved	SLV_ACTIV	MST_ACTIV	RFF	RFNE	TFE	TFNE	ACTIV

Bit	Field	Type	Reset	Description
15:7	Reserved			always read as 0.
6	SLV_ACTIV	r	0x00	Slave FSM activity status 0: Slave FSM is in IDLE state so the Slave part of I2C is not active 1: Slave FSM is not in IDLE state so the Slave part of I2C is active
5	MST_ACTIV	r	0x00	Master FSM activity status 0: Master FSM is in IDLE state so the Master part of I2C is not active 1: Master FSM is not in IDLE state so the Master part of I2C is active
4	RFF	r	0x00	Receive FIFO completely full 0: Receive FIFO is not full 1: Receive FIFO is full

3	RFNE	r	0x00	Receive FIFO not empty 0: Receive FIFO is empty 1: Receive FIFO is not empty
2	TFE	r	0x01	Transmit FIFO completely empty 0: Transmit FIFO is not empty 1: Transmit FIFO is empty
1	TFNF	r	0x01	Transmit FIFO not full 0: Transmit FIFO is full 1: Transmit FIFO is not full
0	ACTIV	r	0x00	I2C activity status This bit is an ORed result of MST_ACTIV bit and SLV_ACTIV bit.

### 23.7.27 I2C transmit FIFO level register (I2C\_TXFLR)

Address offset: 0x74

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CNT	

Bit	Field	Type	Reset	Description
15:2	Reserved			always read as 0
1:0	CNT	r	0x00	Contains the number of valid data entries in the transmit FIFO (0~2)

r r

### 23.7.28 I2C receive FIFO level register (I2C\_RXFLR)

Address offset: 0x78

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CNT	

r r

Bit	Field	Type	Reset	Description
15:12	Reserved			Inter-integrated circuit (I2C) interface always read as 0 UM_MM32F013x_Ver1.00
1:0	CNT	r	0x00	Contains the number of valid data entries in the receive FIFO (0~2)

### 23.7.29 I2C SDA hold time register (I2C\_HOLD)

Address offset: 0x7C

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RX_HOLD							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX_HOLD															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description											
31:24	Reserved			always read as 0.											
23:16	RX_HOLD	rw	0x00	Sets the SDA hold time in units of APB1 system clock period, when I2C device acts as a receiver.											
15:0	TX_HOLD	rw	0x01	Sets the SDA hold time in units of APB1 system clock period, when I2C device acts as a transmitter.											

### 23.7.30 I2C DMA controller (I2C\_DMA)

Address offset: 0x88

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TXEN	RXEN	rw	rw
15:2	Reserved														
1	TXEN	rw	0x00	Transmit DMA enable 0: Transmit DMA disabled 1: Transmit DMA enabled											
0	RXEN	rw	0x00	Receive DMA enable 0: Receive DMA disabled 1: Receive DMA enabled											

### 23.7.31 I2C SDA setup time register (I2C\_SETUP)

Address offset: 0x94

Reset value: 0x0064

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CNT							

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0.
7:0	CNT	rw	0x64	SDA setup time It is recommended that if the required delay is 1000ns, then for an APB1 clock frequency of 10MHz, this register should be programmed to a value of 11. The minimum value of this register is 2.

**23.7.32 I2C general call ACK register (I2C\_GCR)**

Address offset: 0x98

Reset value: 0x0001

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															GC

rw

Bit	Field	Type	Reset	Description
15:1	Reserved			always read as 0.
0	GC	rw	0x01	ACK general call 1: I2C responds with an ACK when it receives a General Call. 0: I2C neither responds with an ACK nor generates an interrupt, when it receives a General Call.

**23.7.33 I2C slave address mask register (I2C\_SLVMASK)**

Address offset: 0xB0

Reset value: 0x03FF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MASK							

rw

Bit	Field	Type	Reset	Description
15:10	Reserved			always read as 0.
9:0	MASK	rw	0x03FF	Slave address mask 1: a corresponding bit in the SAR register requires comparison 0: a corresponding bit in the SAR register is masked, not requiring comparison

**23.7.34 I2C slave receive address register (I2C\_SLVRCVADDR)**

Address offset: 0xB4

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	ADDR
----------	------

r

Bit	Field	Type	Reset	Description
15:10	Reserved			always read as 0
9:0	ADDR	r	0x0000	Address actually received by the slave

## 24

# Universal Asynchronous Receiver Transmitter (UART)

Universal Asynchronous Receiver Transmitter (UART)

## 24.1 UART introduction

The universal asynchronous receiver transmitter (UART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The UART offers a very wide range of baud rates using a fractional baud rate generator. It supports synchronous one-way communication and half-duplex single wire communication. It also supports the modem operations (CTS/RTS).

High speed data communication is possible by using the DMA for multibuffer configuration.

## 24.2 UART main features

- Support asynchronous RS-232S protocol and comply with industrial standard 16550
- Supports DMA request
- Full duplex, asynchronous communications
- Fractional baud rate generator systems
- Common programmable transmit and receive baud rates
- Separate transmit and receive FIFO registers
- Embedded 1-byte Tx and Rx FIFOs
- Data transmission and reception in the way of least significant bit first
- A start bit followed by data bits, with length of 5, 6, 7, or 8 bits, plus stop bits in the end In addition, an optional parity bit (with odd or even select) can be put between the data bits and stop bits.
- The bit 9 can be configured for synchronization frame.
- Generation and detection of hardware odd or even verification
- Generation and detection of line breaks
- Generation and detection of idle lines
- BRK transmission and reception based on LIN protocol
- Support exchange of Tx and Rx signals and inversion of Tx and Rx
- Support baud rate adaption function
- Support hardware automatic flow control
- Support following interrupt sources:
  - Tx BUFFER empty
  - Rx data valid
  - Rx buffer overrun
  - Framing error
  - Parity error
  - Rx break frame
  - Tx shift register complete

- Tx break frame complete
- Rx synchronization frame
- Idle frame complete
- Automatic end of baud rate
- Automatic baud rate error

## 24.3 UART functional description

Any UART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX).

RX: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating between incoming data and noise.

TX: Transmit Data Output. When the transmitter is disabled, the output pin returns to its IO port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level.

- The bus is in the idle state prior to transmission or reception.
- A start bit
- A data word (5, 6, 7 or 8 bits), least significant bit first
- 0.5, 1, 1.5, 2 stop bits indicating that the data frame is completed
- This interface uses a fractional baud rate generator - with a 16-bit integer and 4-bit fraction representation.
- By enabling the SWAP bit in the UART\_GCR register, the interface swaps signals from Tx and Rx ends.
- By enabling the RX\_TOG/TX\_TOG bit in the UART\_GCR register, the interface inverts signals from Tx/Rx ends.

The following pins are required in hardware flow control mode:

- nCTS: Clear To Send blocks the data transmission at the end of the current transfer when high.
- nRTS: Request to send indicates that the UART is ready to receive data when low.

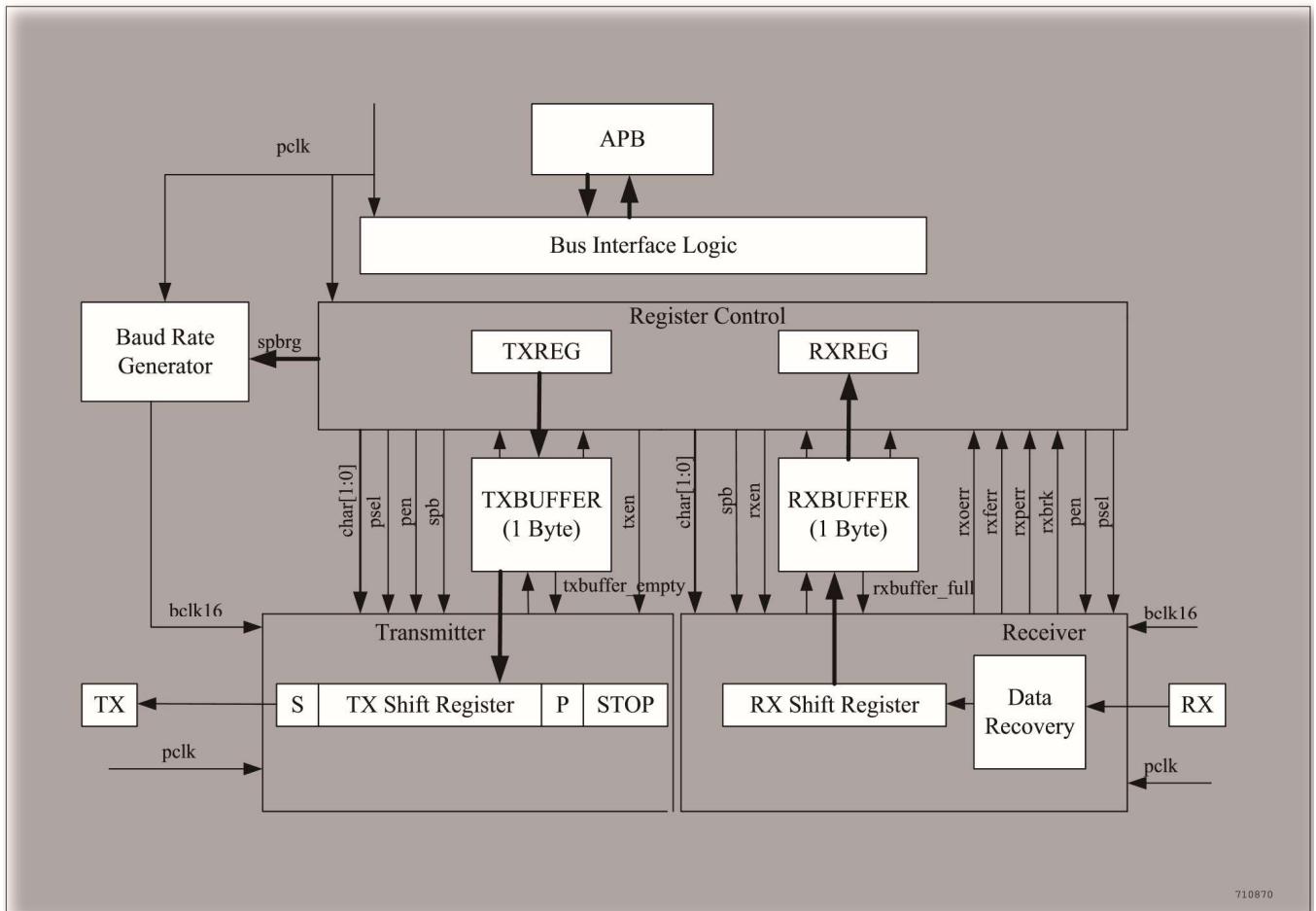


Figure 239. UART block diagram

### 24.3.1 UART character description

Word length may be selected as being 5 to 8 bits by programming the CHAR bit in the UART\_CCR register. The TX pin is in low state during the start bit. It is in high state during the stop bit.

An Idle character is interpreted as an entire data frame of "1"s followed by the start bit of the next frame which contains data (The number of "1" bits will include the number of stop bits).

A Break character is interpreted on receiving "0"s for a frame period (including the period of stop bits which are also "0"s). At the end of the break frame, the transmitter inserts 1 stop bit ("1") again to acknowledge the start bit.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

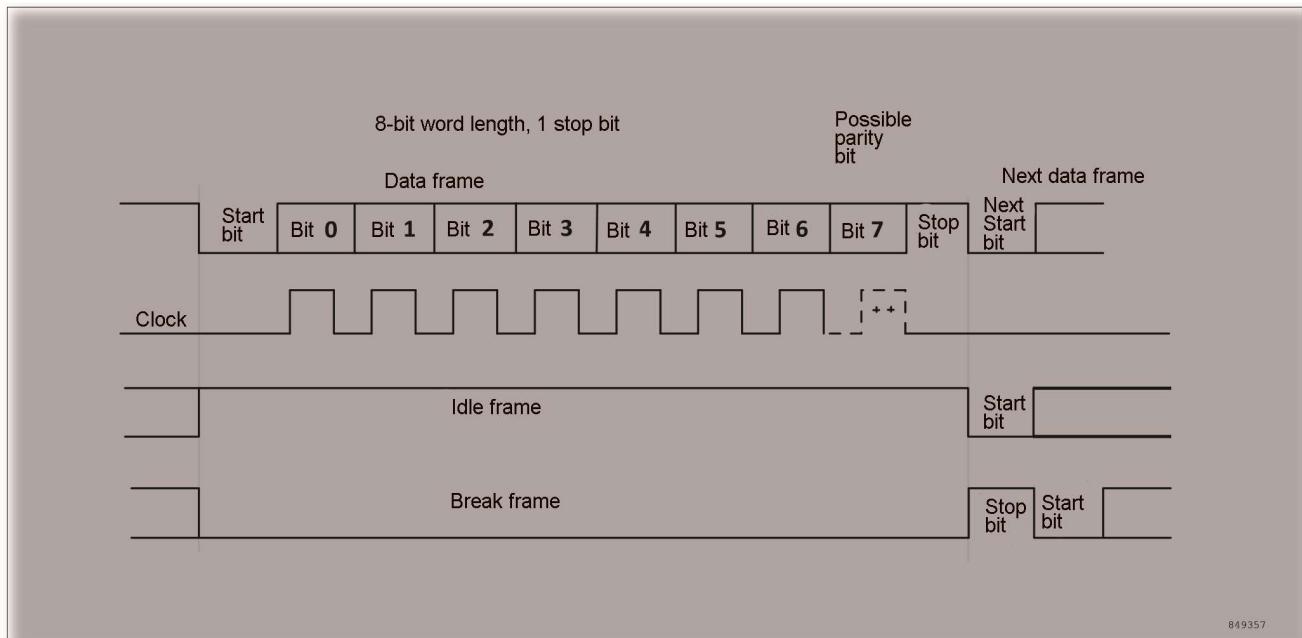


Figure 240. UART timing

### 24.3.2 Transmitter

The transmitter can send 5- to 8-bit data depending on the CHAR bit status. When the transmit enable bit (TXEN) is set, the data in the transmit shift register is output on the TX pin.

#### Character transmission

During a UART transmission, data shifts out least significant bit first on the TX pin. In this mode, the UART\_TDR register consists of a buffer between the internal bus and the transmit shift register.

Every character is preceded by a start bit at low level. The character is followed by a configurable number of stop bits.

The TXEN bit should not be reset during transmission of data. Resetting this bit during the transmission will corrupt the data on the TX pin as the baud rate counters will get frozen. The current data being transmitted will be lost.

#### Configurable stop bits

The number of stop bits to be transmitted with every character can be programmed in bit SPB.

Break frame is comprised of start bit, data bits, optional parity bit and stop bits. Its length is not fixed, depending on the various settings of data bits, parity bit and stop bits. Once the break frame is received, the RXBRK\_INTF bit in the interrupt status register is set to 1.

#### Procedure

1. Enable the UART by writing the UARTEN bit in UART\_GCR to 1.
2. Program the CHAR bit in UART\_CCR to define the word length.
3. Program the number of stop bits in SPB of UART\_CCR.
4. Set the TXEN bit in UART\_GCR.
5. Select the desired baud rate using UART\_BRR and UART\_FRA registers.

6. Write the data to send in the UART\_TDR register (this clears the TX\_INTF bit). Repeat this for each data to be transmitted in case of single buffer.

### Single byte communication

The TX\_INTF bit is always cleared by a write to the data register. The TX\_INTF bit is set by hardware and it indicates:

- The data has been moved from TDR to the shift register and the data transmission has started.
- The TDR register is emptied.
- The next data can be written in the UART\_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXIEN bit is set. When UART transmission is taking place, a write instruction to the UART\_TDR register stores the data in the TDR register and copies it in the shift register at the end of the current transmission.

When no UART transmission is taking place, a write instruction to the UART\_TDR register places the data directly in the shift register, the data transmission starts, and the TX\_INTF bit is immediately set. Meanwhile, the TXBUF\_EMPTY bit in the UART\_CSR is also set. If a frame is transmitted (after the stop bit) and no more data is written to UART\_TDR (TDR register is empty), then TXC is set, indicating all transmissions are complete.

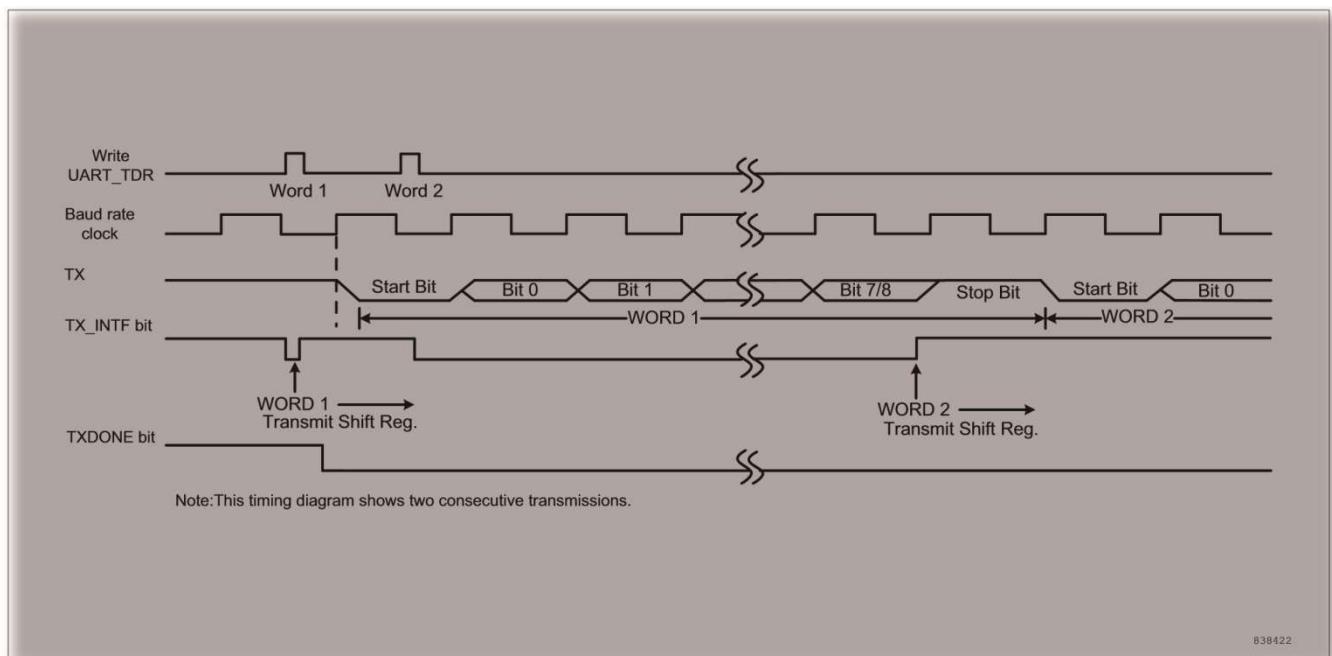


Figure 241. Status bit change during transmission

### Break characters

Setting the BRK bit transmits a break character. If the BRK bit is set to '1', a break character is sent on the TX line after completing the current character transmission. This bit is reset by hardware when the break character is completed (during the stop bit of the break character).

The UART inserts a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

### Idle characters

After TXEN is enabled, setting the UART\_IDLR bit to 1 drives the UART to send an idle frame

before the subsequent data transmission. The length of the idle frame is written to the UART\_IDL register.

### 24.3.3 Receiver

The receiver FIFO depth is one byte.

#### Character reception

During a UART reception, data shifts in least significant bit first through the RX pin. In this mode, the UART\_RDR register consists of a buffer between the internal bus and the receive shift register.

Procedure:

1. Enable the UART by writing the UARTEN bit in UART\_GCR register to 1.
2. Program the CHAR bit in UART\_CCR to define the word length.
3. Program the number of stop bits in SPB of UART\_CCR.
4. Select the desired baud rate using UART\_BRR and UART\_FRA registers.
5. Set the RXEN bit in UART\_GCR. This enables the receiver which begins searching for a start bit.

When a character is received,

- The RX\_INTF bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- An interrupt is generated if the RXIEN bit is set.
- The error flags can be set if a frame error or an overrun error has been detected during reception.
- The software reads the UART\_RDR register. The RX\_INTF bit must be cleared before the end of the reception of the next character.

The RXEN bit should not be reset while receiving data. If the RXEN bit is cleared during reception, the reception of the current byte will be lost.

#### Break character

When a break frame is received, the UART will set an RXBRK\_INTF interrupt.

#### Idle character

If the UART\_IDLR is not cleared, UART will set the RXIDL\_INTF interrupt when an idle frame is received.

#### Overrun error

An overrun error occurs when a character is received when UART\_RDR has not been read.

When an overrun error occurs:

- The RXOERR\_INTF bit is set.
- The RDR content will not be lost. The previous data is available when a read to UART\_RDR is performed.
- The shift register will be overwritten. Subsequently, any data received is lost.
- An interrupt is generated if the RXOERREN bit is set.

### Framing error

A framing error is detected when the stop bit is not recognized or received at the expected time. When the framing error is detected:

- The RXFERR\_INTF bit is set by hardware.
- The invalid data is not transferred from the Shift register to the UART\_RDR register.
- An interrupt is generated if the RXFERREN bit is set.

### 24.3.4 9-bit data communication

If the B8EN control bit in the UART\_CCR register, then the UART is able to transmit and receive 9-bit data. Note: Parity enable bit (PEN) is invalid after enabling B8E.

During data transmission, the B8TXD should be set before the data is written to the transmit register UART\_TDR. B8TXD, acting as MSB of data to be sent, is delivered together with the UART\_TDR value. If the B8TOG is set and the B8TXD is identical to the B8POL, then the data is used as the address frame or synchronization frame. The B8TXD will toggle automatically at the end of transmission.

It is not required to set the B8TXD as an invalid level in the subsequent data transfer.

During data reception, the most significant bit of the received data can be obtained by reading the B8RXD bit in the register. If the received B8RXD is identical to the B8POL, the RXB8\_INTF bit in the interrupt status register UART\_ISR will be set.

### 24.3.5 Multiprocessor communication

There is a possibility of performing multiprocessor communication with the UART (several UARTs connected in a network). For instance, one of the UARTs can be the master, its TX output is connected to the RX input of the other UART slaves.

Respective TX outputs of UART slaves are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient should actively receive the subsequent message contents, thus reducing redundant UART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in UART\_CCR register is set to 1. RWU can be controlled automatically by hardware or written by the software under certain conditions.

The UART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the UART\_CCR register:

- Idle bus detection if the WAKE bit is reset;
- Address Mark detection if the WAKE bit is set.

#### Idle bus detection (WAKE = 0)

The UART enters mute mode when the RWU bit is written to 1. It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the interrupt status flag (RX\_INTF) is not set. RWU can also be written to 0 by software.

### Address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is ‘B8POL’ else they are considered as data. In an address byte, the address of the targeted receiver is compared by the receiver with its own address. The address and mask bit of the receiver are programmed in the UART\_RXADDR and UART\_RXMASK registers.

The UART enters mute mode when a byte is received which does not match its programmed address. In this case, the RWU bit is set by hardware. Neither the interrupt flag (RX\_INTF) is set nor an interrupt/DMA request is issued for this reception, as the UART would have entered mute mode.

It exits from mute mode when a byte is received which matches the programmed address in the receiver. Then the RWU bit is cleared and subsequent bytes are received normally. The interrupt status flag (RX\_INTF) is set for the matched address character since the RWU bit has been cleared.

### 24.3.6 Single-wire half-duplex communication

The single-wire half-duplex mode is selected by setting the HDSEL bit in the UART\_SCR register. In this mode, the SCEN bit in the UART\_SCR register must be kept cleared.

The UART can be configured to follow a single-wire half-duplex protocol. In single-wire half-duplex mode, the TX and RX pins are connected internally. The selection between half and full-duplex communication is made with a control bit ‘HALF DUPLEX SEL’ (HDSEL in UART\_SCR).

As soon as HDSEL is written to 1:

- RX is no longer used
- TX is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured as floating input (or output high open-drain) when not driven by the UART.

Apart from this, the communications are similar to what is done in normal UART mode. The conflicts on the line must be managed by the software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continues to occur as soon as a data entry is written in the data register while the TXEN bit is set.

### 24.3.7 Smartcard

The Smartcard mode is selected by setting the SCEN bit in the UART\_SCR register.

The Smartcard interface is designed to support Smartcard asynchronous protocol as defined in the ISO7816-3 standard. The UART should be configured as:

- 8 bits plus parity: where CHAR = 11 and PEN = 1 in the UART\_CCR register
- 1.5 stop bits when transmitting and receiving: where SPB1 = 1 and SPB0 = 1 in the UART\_CCR register.

The figure below shows examples of what can be seen on the data line with and without parity error.

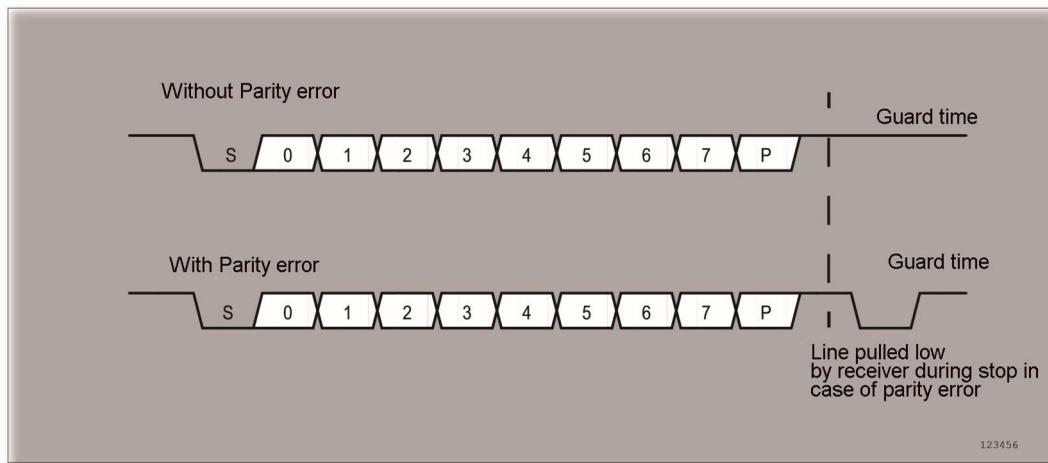


Figure 242. UART block diagram

When connected to a smartcard, the UART TX output drives a bidirectional line of the smartcard. To do this, RX and TX must be connected to the same I/O. During the transmission of start bit and data bytes, the Tx enable bit (TXEN) of the transmitter is set. This bit is released (weak pull-up) during the transmission of stop bits. Therefore, the receiver can hold the data line low in case of parity error. If the TXEN is not used and TX is held high during the transmission of stop bits, then the receiver can drive this line if TX is configured as open drain.

Smartcard is a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is delayed by a minimum of 1/2 baud clock. In normal operation, a full transmit shift register will start shifting on the next baud clock edge. In Smartcard mode, this transmission is further delayed by a guaranteed 1/2 baud clock.
- If a parity error is detected during reception of a data frame programmed with a 0.5 or 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame (i.e. the end of stop bits). This is to indicate to the Smartcard that the data transmitted to UART has not been correctly received. This NACK signal (pulling transmit line low for 1 baud clock) will cause a framing error on the transmitter side (configured with 1.5 stop bits). The application can handle re-sending of data according to the protocol. A parity error is 'NACK'ed by the receiver if the SCAEN control bit is set, otherwise a NACK is not transmitted.
- The assertion of the TXC flag can be delayed by programming the Guard Time register. In normal operation, TXC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode, an empty transmit shift register triggers the guard time counter to count up to the programmed value in the Guard Time register. TXC is forced low during this time. When the guard time counter reaches the value in the Guard Time register, TXC is asserted high.
- The de-assertion of TXC flag is unaffected by Smartcard mode.
- If a framing error is detected by the transmitter (due to a NACK from the receiver), the NACK will not be detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted, the receiver will not detect the NACK as a start bit.

Note: 1. A break character is not significant in Smartcard mode. A 00h data with a framing error will be treated as data and not as a break.

2. No IDLE frame is transmitted when toggling the TXEN bit. The IDLE frame is not defined by the ISO protocol.

The figure below details how the NACK signal is sampled by the UART. In this example, the UART transmits a data and is configured with 1.5 stop bits. The receiver part of the UART is enabled in order to check the integrity of the data and the NACK signal.

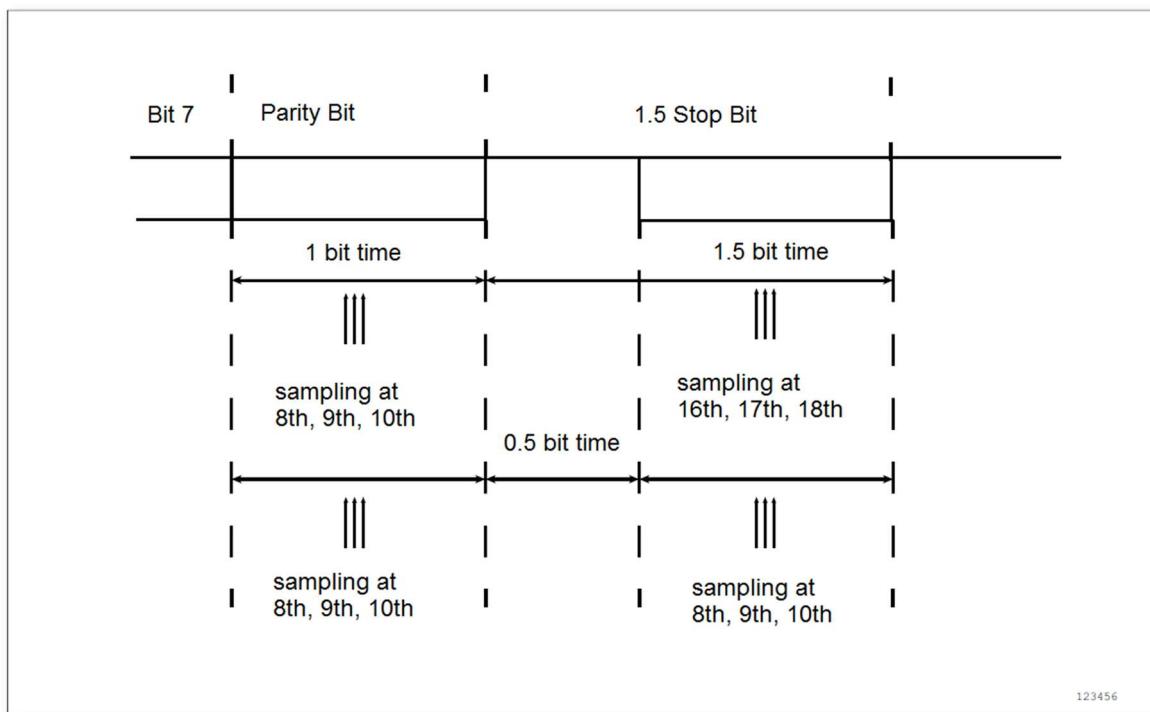


Figure 243. UART block diagram

#### 24.3.8 Fractional baud rate generator

The corresponding baud rate can be set using the BRR and FRA registers. Refer to the formula below:

$$f_{baudrate} = \frac{f_{PCLK}}{16 \times UARTDIV}$$

$$UARTDIV = BRR + \frac{FRA}{16}$$

This leads to:

$$f_{baudrate} = \frac{f_{PCLK}}{16 \times BRR + FRA}$$

The minimum value of BRR register is 1.

#### 24.3.9 Sampling

Since there is no individual clock for asynchronous communications, it is required that the receiver is synchronous. UART has one detection circuit to obtain correct character data from the receive pin 'RX'. UART adopts the x16 data baud rate 'bclk16' clock for data sampling from the RX pin. Each data entry is sampled by 16 clocks, from which the sampling values of the 7, 8 and 9 bits are used.

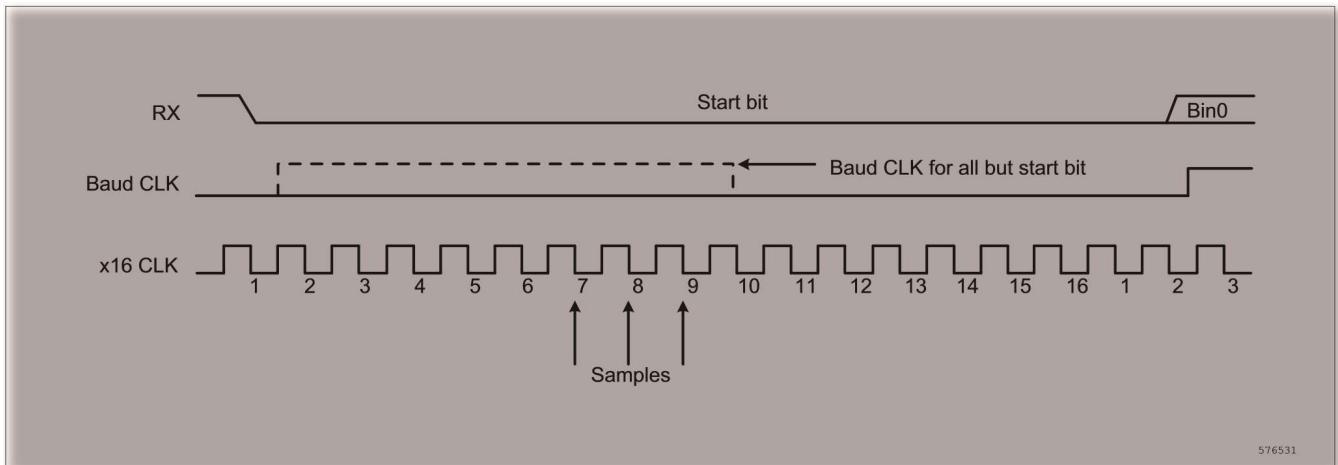


Figure 244. RX pin sampling scheme

#### 24.3.10 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PEN bit in the UART\_CCR register. The invalid data is not transferred from the Shift register to the UART\_RDR register in case of parity error.

Even parity: the parity bit is calculated to obtain an even number of “1s” inside the frame and the parity bit.

Ex: data = 00110101; 4 bits set => parity bit will be 0 if even parity is selected (PSEL bit in UART\_CCR = 1).

Odd parity: the parity bit is calculated to obtain an odd number of “1s” inside the frame and the parity bit.

Ex: data = 00110101; 4 bits set => parity bit will be 1 if odd parity is selected (PSEL bit in UART\_CCR = 0).

Transmission mode: If the PEN bit is set in UART\_CCR, then the parity bit is sent after the data in the data register is transmitted (even number of “1s” if even parity is selected or an odd number of “1s” if odd parity is selected). If the parity check fails, the RXPERR\_INTF flag is set in the UART\_ISR register and an interrupt is generated if RXPERREN is set in advance.

Note: The parity check is invalid if the 9bit function is enabled.

#### 24.3.11 Automatic baud rate detection

UART can detect and set the UART\_BRR register automatically according to the signal it has received.

The baud rate is automatically configured as follows:

1. Set the Former\_edge and Latter\_edge in the ABRCR register to automatically select the former edge and the latter edge of baud rate.
2. Set the Abr\_bitcnt in the ABRCR register to select the automatic detection of baud rate bit length.
3. Set the Abr\_en in the ABRCR register to enable the automatic detection of baud rate.

Enable ABRENDIEN to generate an automatic baud rate completion interrupt by the automatic end of the baud rate.

Enable ABRERRIEN to generate an automatic baud rate error interrupt in case of an automatic baud rate bit error or detection overrun error.

Ex:

1. Set Former\_edge = 1, Latter\_edge=0, Abr\_bitcnt=2, Abr\_en=1

When the data 0xEF has been received, the baud rate will be detected automatically and the UART\_BRR register will be reset. The Rx FIFO will receive the remaining data 0x0E.

2. Set Former\_edge = 0, Latter\_edge=1, Abr\_bitcnt=2, Abr\_en=1

When the data 0xF8 has been received, the baud rate will be detected automatically and the UART\_BRR register will be reset. The Rx FIFO will receive the remaining data 0x1F.

Note: If Former\_edge = 1 and the LSB bit of received data is not 1, then a reception error will occur. No data will be in the Rx FIFO.

#### 24.3.12 Communication using DMA

The UART is capable of continuing communication using the DMA.

##### Transmission using DMA

To transmit using DMA, the user should: write the UART\_TDR register address in the DMA control register to configure it as the destination of the transfer; write the memory address to configure it as the source of the transfer; configure the total number of bytes to be transferred. DMA mode can be enabled for transmission by setting DMAMODE bit in the UART\_GCR register. Data is loaded from a specified SRAM area by DMA to the UART\_TDR register whenever the TXEN bit is set.

##### Reception using DMA

To receive using DMA, the user should: write the UART\_RDR register address in the DMA control register to configure it as the source of the transfer; write the memory address to configure it as the destination of the transfer; configure the total number of bytes to be transferred.

DMA mode can be enabled for reception by setting DMAMODE bit in the UART\_GCR register. If the RXEN bit is set, data is loaded from the UART\_RDR register to a specified SRAM area by DMA whenever a data byte is received.

### 24.4 UART interrupt requests

Table 70. UART interrupt requests

Interrupt Event	Interrupt Status	Enable Bit
Tx buffer empty	TX_INTF	TXIEN
Effective data received	RX_INTF	RXIEN
Tx shift register complete	TXC_INTF	TXC_EN
Rx overrun error	RXOERR_INTF	RXOERREN
Parity error	RXPERR_INTF	RXPERREN
Framing error	RXFERR_INTF	RXFERREN
Rx break frame	RXBRK_INTF	RXBRKEN

Interrupt Event	Interrupt Status	Enable Bit
Tx break frame	TXBRK_INTF	TXBRK_EN
Rx synchronization frame	RXB8_INTF	RXB8_EN
Rx idle frame	RXIDL_INTF	RXIDLEN
Automatic end of baud rate	ABREND_INTF	ABRENDIEN
Automatic baud rate error	ABRERR_INTF	ABRERRIEN

These events generate an interrupt if the corresponding interrupt Enable Control Bit is set.

## 24.5 UART register description

Table 71. Overview of UART registers

Offset	Acronym	Register Name	Reset	Section
0x00	UART_TDR	UART transmit data register	0x00000000	Section 24.5.1
0x04	UART_RDR	UART receive data register	0x00000000	Section 24.5.2
0x08	UART_CSR	UART current status register	0x00000009	Section 24.5.3
0x0C	UART_ISR	UART interrupt status register	0x00000000	Section 24.5.4
0x10	UART_IER	UART interrupt enable register	0x00000000	Section 24.5.5
0x14	UART_ICR	UART interrupt clear register	0x00000000	Section 24.5.6
0x18	UART_GCR	UART global control register	0x00000000	Section 24.5.7
0x1C	UART_CCR	UART common control register	0x00000000	Section 24.5.8
0x20	UART_BRR	UART baud rate register	0x00000001	Section 24.5.9
0x24	UART_FRA	UART fractional baud rate register	0x00000000	Section 24.5.10
0x28	UART_RXADDR	UART receive address register	0x00000000	Section 24.5.11
0x2C	UART_RXMASK	UART receive mask register	0x000000FF	Section 24.5.12
0x30	UART_SCR	UART SCR register	0x00000000	Section 24.5.13
0x34	UART_IDLR	UART IDLE data length register	0x0000000C	Section 24.5.14
0x38	UART_ABRCR	UART automatic baud rate control register	0x00000000	Section 24.5.15

### 24.5.1 UART transmit data register (UART\_TDR)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:8	Reserved			always read as 0
7:0	TXREG	rw	0x00	Transmit data register

### 24.5.2 UART receive data register (UART\_RDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:8	Reserved			always read as 0
7:0	RXREG	r	0x00	Receive data register This is a read-only register.

### 24.5.3 UART current status register (UART\_CSR)

Address offset: 0x08

Reset value: 0x0000 0009

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TXBUF_EMPTY	TXFULL	RXAVL	TXC
r      r      r      r															
Bit	Field	Type	Reset	Description											
31:4	Reserved			always read as 0.											
3	TXBUF_EMPTY	r	0x01	Transmit buffer empty flag bit 1: Transmit buffer is empty 0: Transmit buffer is not empty											
2	TXFULL	r	0x00	Transmit buffer full flag bit 1: Transmit buffer is full 0: Transmit buffer is not full											
1	RXAVL	r	0x00	Receive valid data flag bit This bit is set when the Rx buffer has received a complete byte. 1: Rx buffer has received a complete and effective byte. 0: Rx buffer is empty.											
0	TXC	r	0x01	Transmit complete flag bit 1: Tx buffer and Tx shift register are both empty. 0: Tx buffer is not empty.											

#### 24.5.4UART interrupt status register (UART\_ISR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved      ABRERR_INTF      ABREND_INTF      RXDLE_INTF      RXB8_INTF      TX_BRK_INTF      RX_BRK_INTF      RX_FERR_INTF      RXP_ERR_INTF      RXO_ERR_INTF      TXC_INTF      RX_INTF      TX_INTF															
r      r      r      r      r      r      r      r      r      r      r      r      r      r      r      r															
Bit	Field	Type	Reset	Description											
31:12	Reserved			always read as 0.											
11	ABRERR_INTF	r	0x00	UART auto baud rate error interrupt flag bit 1: Auto baud rate error 0: No auto baud rate error											

Bit	Field	Type	Reset	Description
10	ABREN <sub>D</sub> _INTF	r	0x00	UART auto baud rate end interrupt flag bit 1: Auto baud rate is at the end 0: Automatic baud rate is not at the end
9	RXIDLE_INTF	r	0x00	UART receive frame idle interrupt flag bit After stop bits, RX pin has received several (number of UART_IDLR) high levels. 1: Idle frame detected 0: No idle frame
8	RXB8_INTF	r	0x00	UART synchronization frame interrupt flag bit In 9-bit communication mode, the RXB8_INT is set once the 9 bit of received data is identical to the CCR.B8POL register. This bit can be treated as an interrupt request signal. 1: Synchronization frame received 0: Synchronization frame not received
7	TXBRK_INTF	r	0x00	UART break frame transmission complete interrupt flag bit 1: Transmission of shift register break frame data is completed 0: Shift register is empty or shift transmission is ongoing Note: The break frame cannot be sent continuously.
6	RXBRK_INTF	r	0x00	UART receive frame break interrupt flag bit After abnormal stop bits, RX pin has received no less than 10 low levels. 1: Break frame detected 0: No break frame
5	RXFERR_INTF	r	0x00	Frame error interrupt flag bit A frame error occurs when an abnormal stop bit is detected. 1: A frame error detected 0: no frame error
4	RXPERR_INTF	r	0x00	Parity error interrupt flag bit 1: Parity error detected 0: No parity error
3	RXOERR_INTF	r	0x00	Receive overflow error interrupt flag bit This bit is set only when autoflowen = 0. 1: Overrun error received 0: No overrun error

Bit	Field	Type	Reset	Description
2	TXC_INTF	r	0x00	UART Tx shift register complete interrupt flag bit 1: Transmission of shift register data is completed 0: Shift register is empty or shift transmission is ongoing Note: The flag bit is related with the guard time.
1	RX_INTF	r	0x00	Receive valid data interrupt flag bit This bit is set when the Rx buffer receives a complete byte. 1: Rx buffer has received data of effective byte. 0: Rx buffer is empty.
0	TX_INTF	r	0x00	Transmit buffer empty interrupt flag bit 1: Transmit buffer is empty 0: Transmit buffer is not empty

#### 24.5.5UART interrupt enable register (UART\_IER)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ABRERR_IEN	ABREND_IEN	RXIDL_EN	RXB8_IEN	TXBRK_IEN	RX_BRK_EN	RX_FERR_EN	RXP_ERR_EN	RXO_ERR_EN	TXC_IEN	RXIEN	TXIEN
rw				rw				rw				rw			

Bit	Field	Type	Reset	Description
31:12	Reserved			always read as 0.
11	ABRERRIEN	rw	0x00	Auto baud rate error interrupt enable bit 1: interrupt enabled 0: interrupt disabled
10	ABRENDIEN	rw	0x00	Auto baud rate end interrupt enable bit 1: interrupt enabled 0: interrupt disabled
9	RXIDLLEN	rw	0x00	Receive frame idle interrupt enable bit 1: interrupt enabled 0: interrupt disabled

Bit	Field	Type	Reset	Description
8	RXB8_IEN	rw	0x00	UART synchronization frame interrupt enable control bit 1: Rx synchronization frame interrupt enabled 0: Rx synchronization frame interrupt disabled
7	TXBRK_IEN	rw	0x00	UART break frame transmission complete interrupt enable control bit 1: Tx break frame complete interrupt enabled 0: Tx break frame complete interrupt disabled
6	RXBRKEN	rw	0x00	UART receive frame break interrupt enable bit 1: interrupt enabled 0: interrupt disabled
5	RXFERREN	rw	0x00	Frame error interrupt enable bit 1: interrupt enabled 0: interrupt disabled
4	RXPERRREN	rw	0x00	Parity error interrupt enable bit 1: interrupt enabled 0: interrupt disabled
3	RXOERREN	rw	0x00	Receive overflow error interrupt enable bit 1: interrupt enabled 0: interrupt disabled
2	TXC_IEN	rw	0x00	UART Tx shift register complete interrupt enable control bit 1: Transmission of shift register data is completed 0: Shift register is empty or shift transmission is ongoing
1	RX_IEN	rw	0x00	Receive buffer interrupt enable bit 1: interrupt enabled 0: interrupt disabled
0	TX_IEN	rw	0x00	Transmit buffer empty interrupt enable bit 1: interrupt enabled 0: interrupt disabled

#### 24.5.6UART interrupt clear register (UART\_ICR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ABRERRCLR	ABRENDCLR	RXIDLCLR	RXB8_CLR	TXBRK_CLR	RXBRKCLR	RXFEEERRCLR	RXPERRCLR	RXOERRCLR	TXC_ICLRL	RX_ICLRL	TX_ICLRL
W W W W W W W W W W W W W W W															
Bit	Field	Type	Reset	Description											
31:12	Reserved			always read as 0.											
11	ABRERRCLR	w	0x00	UART auto baud rate error interrupt clear bit 1: interrupt cleared 0: interrupt uncleared											
10	ABRENDCLR	w	0x00	UART auto baud rate end clear bit 1: interrupt cleared 0: interrupt uncleared											
9	RXIDLCLR	w	0x00	UART receive frame idle interrupt clear bit 1: interrupt cleared 0: interrupt uncleared											
8	RXB8_CLR	w	0x00	UART synchronization frame interrupt flag clear control bit 1: Rx synchronization frame interrupt flag cleared 0: No action											
7	TXBRK_CLR	w	0x00	UART break frame transmission complete interrupt flag clear control bit 1: Break frame transmission complete interrupt flag cleared 0: No action											
6	RXBRKCLR	w	0x00	UART receive frame break interrupt clear bit 1: interrupt cleared 0: interrupt uncleared											
5	RXFERRCLR	w	0x00	Frame error interrupt enable bit 1: interrupt cleared 0: interrupt uncleared											
4	RXPERRCLR	w	0x00	Parity error interrupt clear bit 1: interrupt cleared 0: interrupt uncleared											

Bit	Field	Type	Reset	Description
3	RXOERRCLR	w	0x00	Receive overflow error interrupt clear bit 1: interrupt cleared 0: interrupt uncleared
2	TXC_ICLR	w	0x00	Transmit complete interrupt clear bit 1: interrupt cleared 0: interrupt uncleared
1	RX_ICLR	w	0x00	Receive interrupt clear bit 1: interrupt cleared 0: interrupt uncleared
0	TX_ICLR	w	0x00	Transmit buffer empty interrupt clear bit 1: interrupt cleared 0: interrupt uncleared

#### 24.5.7 UART global control register (UART\_GCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TX_TOG	RX_TOG	SWAP	SEL_B8	Reserved	TXEN	RXEN	AUTO_FLOW_EN	DMA_MODE	UARTEN	
rw					rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:11	Reserved			always read as 0.
10	TX_TOG	rw	0x00	Transmit toggle bit 1: Transmit toggle enabled 0: invalid
9	RX_TOG	rw	0x00	Receive toggle bit 1: Receive toggle enabled 0: invalid
8	SWAP	rw	0x00	Swap between input and output 1: Input and output are swapped 0: invalid  Note: Once the SWAP bit is set, the MODE in the GPIOx_CRL register needs to be changed, e.g. the original input mode changes to the output mode.

Bit	Field	Type	Reset	Description
7	SEL_B8	rw	0x00	This bit selects whether B8 data transmission or reception is valid or not. 1: B8 data transfer 0: invalid  When the B8EN bit is enabled in the UART_CCR, 9-bit data is allowed to transmit; the data with a length of 'CHAR bit +1' is allowed to receive.
6:5	Reserved			always read as 0
4	TXEN	rw	0x00	Enable transmit 1: Transmission enabled. 0: Transmission disabled. TX BUFFER can be cleared.
3	RXEN	rw	0x00	Enable receive 1: Reception enabled. 0: Reception disabled. RX BUFFER can be cleared.
2	AUTO FLOWEN	rw	0x00	Automatic flow control enable bit 1: Automatic flow control enabled 0: Automatic flow control disabled
1	DMAMODE	rw	0x00	DMA mode selection bit 1: Select DMA mode 0: Select normal mode
0	UARTEN	rw	0x00	UART mode selection bit 1: UART mode enabled 0: UART mode disabled

#### 24.5.8UART common control register (UART\_CCR)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Reserved

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

Res.	LIN	WAKE	RWU	B8EN	B8TOGB8POL	B8TXD	B8RXD	SPB1	CHAR	BRK	SPB0	PSEL	PEN
	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:15	Reserved			always read as 0.
14	LIN	rw	0x00	UART LIN protocol transfer break frame enable bit 1: LIN protocol valid 0: LIN protocol invalid

Bit	Field	Type	Reset	Description
13	WAKE	rw	0x00	<p>Wakeup method This bit determines the way to wake up UART.</p> <p>1: Address flag wakeup 0: Idle bus wakeup</p> <p>Note: UART should have received one byte before it is placed in the mute mode. Otherwise, it will not be wakened up by idle bus detection.</p>
12	RWU	rw	0x00	<p>Receive wakeup This bit determines whether to place the UART in mute mode. It can be set or reset by software. When a wakeup sequence arrives, the hardware will clear it automatically.</p> <p>1: Receiver in mute mode 0: Receiver in normal mode</p> <p>If the Address flag wakeup is set, it can not be revised by software, if the receive buffer is not empty.</p>
11	B8EN	rw	0x00	<p>UART synchronization frame bit 9 enable control bit</p> <p>The enable bit is checked when this bit is enabled. PEN is then invalid.</p> <p>1: Synchronization frame bit 9 transmission enabled 0: Synchronization frame bit 9 transmission disabled</p>
10	B8TOG	rw	0x00	<p>UART synchronization frame transmission bit 9 auto toggle control bit</p> <p>1: Bit 9 auto toggle enabled 0: Bit 9 auto toggle disabled</p> <p>Note: When B8TXD value is identical to B8POL value and the register configuration is completed, the first transmitted data is the default address bit and the second data begins to toggle.</p>
9	B8POL	rw	0x00	<p>UART synchronization frame bit 9 polarity control bit</p> <p>1: Synchronization frame bit 9 high level enabled 0: Synchronization frame bit 9 low level enabled</p>
8	B8TXD	rw	0x00	<p>UART synchronization frame transmitted data bit 9</p> <p>1: Tx synchronization frame bit 9 is high level 0: Tx synchronization frame bit 9 is low level</p>
7	B8RXD	r	0x00	<p>UART synchronization frame received data bit 9 Read-only.</p> <p>1: Rx synchronization frame bit 9 is high level 0: Rx synchronization frame bit 9 is low level</p>
6	SPB1	rw	0x00	Stop selection bit, which sets the number of stop bits in conjunction with SPB0.
5:4	CHAR	rw	0x00	<p>UART data width bit</p> <p>00: 5 bits 01: 6 bits 10: 7 bits 11: 8 bits</p>

Bit	Field	Type	Reset	Description
3	BRK	rw	0x00	UART transmit frame break 1: serial forced output of logic '0' (frame break) 0: break disabled
2	SPB0	rw	0x00	Stop bit selection Set the number of Tx stop bits. SPB1, SPB0: 00, 1 stop bit SPB1, SPB0: 01, 2 stop bits SPB1, SPB0: 10, 0.5 stop bit SPB1, SPB0: 11, 1.5 stop bits
1	PSEL	rw	0x00	Parity selection bit When enabled, this bit selects whether even parity or odd parity is used. 1: even parity 0: odd parity
0	PEN	rw	0x00	Parity enable bit 1: Tx and Rx parity check enabled. 0: Parity check disabled.

#### 24.5.9 UART baud rate register (UART\_BRR)

Address offset: 0x20

Reset value: 0x0000 0001

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

Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

DIV\_Mantissa

rw															
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
31:16	Reserved			always read as 0.
15:0	DIV_Mantissa	rw	0x0001	Integer part of UARTDIV These 16 bits define the integer part of the UART frequency divider division factor (UARTDIV). The minimum value of DIV_Mantissa is 1.

#### 24.5.10 UART fractional baud rate register (UART\_FRA)

Address offset: 0x24

Reset value: 0x0000 0000

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

Reserved
----------

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	DIV_Fraction
----------	--------------

rw	rw	rw	rw
----	----	----	----

Bit	Field	Type	Reset	Description
31:4	Reserved			always read as 0.
3:0	DIV_Fraction	rw	0x00	Fraction part of UARTDIV These 4 bits define the fraction part of the UART frequency divider division factor (UARTDIV).

#### 24.5.11 UART receive address register (UART\_RXADDR)

Address offset: 0x28

Reset value: 0x0000 0000

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

Reserved
----------

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	RXADDR
----------	--------

rw															
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
31:8	Reserved			always read as 0.
7:0	RXADDR	rw	0x00	Local address match with the UART synchronization frame data. When RXMASK = 0xFF, RXB8_INTF is generated when the received synchronization frame data is identical to the local address match. Address 0 is the general call address. A response will be generated when this address is received.

#### 24.5.12 UART receive mask register (UART\_RXMASK)

Address offset: 0x2C

Reset value: 0x0000 00FF

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

Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved

RXMASK

rw															
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
31:8	Reserved			always read as 0.
7:0	RXMASK	rw	0xFF	If all data bits are '0', a synchronization frame break request is generated when any data is received. If a data bit is '1', a synchronization frame break request is generated when the RDR matches with the corresponding bit in the RXADDR.

### 24.5.13 UART SCR register (UART\_SCR)

Address offset: 0x30

Reset value: 0x0000 0000

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

Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	HDSEL	SCFCNT	Res.	NACK	SCAE	SCEN
	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:13	Reserved			Reserved, always read as 0.
12	HDSEL	rw	0x00	Single-wire half-duplex mode selection. 1: Enable the half-duplex mode 0: Disable the half-duplex mode
11:4	SCFCNT	rw	0x00	ISO7816 guarding counter. If the Tx data is at low level in the period of guarding counter, it is not allowed to be the start bit of next data. 0 = 16 baud rate count time 15..1 = 15..1 time
3	Reserved			Reserved and always read as 0.
2	NACK	r	0x00	Master receive frame ACK bit

Bit	Field	Type	Reset	Description
1	SCAEN	rw	0x00	ISO7816 check auto ACK bit 1: Auto ACK enabled 0: Auto ACK disabled
0	SCEN	rw	0x00	ISO7816 enable control bit 1: ISO7816 function enabled 0: ISO7816 function disabled

#### 24.5.14 UART IDLE data length register (UART\_IDLR)

Address offset: 0x34

Reset value: 0x0000 000C

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

Bit	Field	Type	Reset	Description
31:16	Reserved			Reserved and always read as 0.
15:0	IDLR	rw	0x000C	UART idle data length register The data length is not 0.

#### 24.5.15 UART automatic baud rate control register (UART\_ABRCR)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
											Latter_edge	Former_edge	Abr_bitcnt	Abren	
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:5	Reserved			Reserved, always read as 0.
4	LATTER_EDGE	rw	0x00	Select the latter edge of baud rate automatically 1: Rising edge 0: Falling edge

Bit	Field	Type	Reset	Description
3	FORMER_EDGE	rw	0x00	Select the former edge of baud rate automatically 1: Rising edge 0: Falling edge
2:1	ABR_BITCNT	rw	0x00	Automatic baud rate length detection. This bit detects the length of bits between the former edge and the latter edge. 11:1 bit 10: 2 bits 01: 4 bits 00: 8 bits
0	ABREN	rw	0x00	Auto baud rate enable. This bit enables the auto baud rate only when the UART is idle. After it is enabled, it detects the edge of received signal. After the auto baud rate detection, the hardware set the UART_BRR and UART_FRA registers automatically. 1: Auto baud rate enabled 0: Auto baud rate disabled

## Controller Area Network (CAN)

### 25.1 CAN introduction

CAN has been designed to manage a high number of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for transmit messages. The priority can be configured by software.

For safety-critical applications, the CAN controller provides all hardware functions for supporting the Time Triggered Communication option.

### 25.2 CAN main features

- Supports CAN protocol version 2.0 A and 2.0 B
- Extended receive buffers (64 bytes FIFO)
- Supports 11-bit identifier as well as 29-bit identifier
- Bit rates up to 1 Mbit/s
- PeliCAN mode extensions:
  - Error counters with read/write access
  - Programmable error warning limit
  - Last error code register
  - Error interrupt for each CAN-bus error
  - Arbitration lost interrupt with detailed bit position
  - Single-shot transmission (no re-transmission)
  - Listen only mode (no acknowledge, no active error flags)
  - Software bit rate detection
  - Acceptance filter extension (4-byte code, 4-byte mask)
  - Reception of 'own' messages (self reception request)

### 25.3 CAN controller general description

In today's CAN applications, the number of nodes in a network is increasing and often several networks are linked together via gateways. Typically the number of messages in the system (and thus to be handled by each node) has significantly increased. In addition to the application messages, network management and diagnostic messages have been introduced.

An enhanced filtering mechanism is required to handle each type of message.

Furthermore, application tasks require more CPU time, therefore real-time constraints caused by message reception have to be reduced.

A receive FIFO scheme allows the CPU to be dedicated to application tasks for a long time period without losing messages.

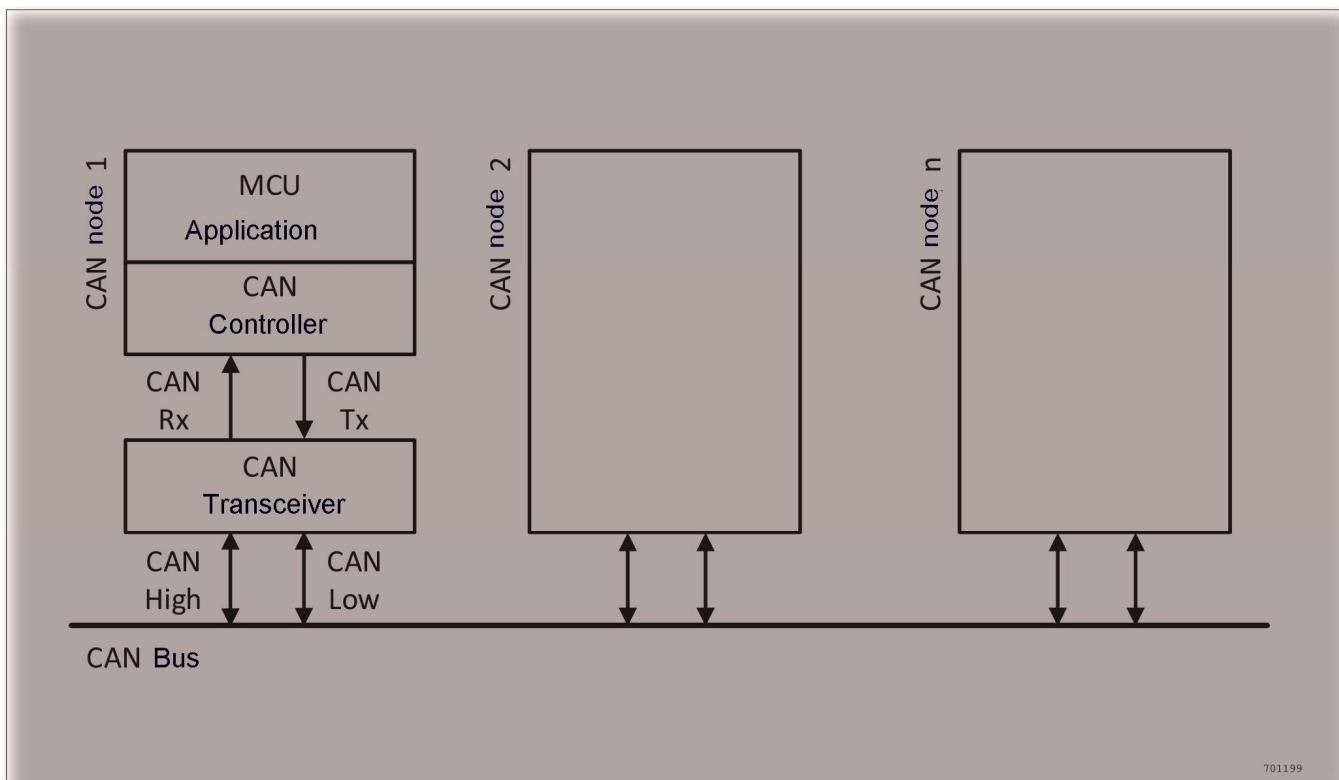


Figure 245. CAN network topology

### 25.3.1 CAN 2.0B active core

The CAN module handles the transmission and the reception of CAN messages fully autonomously. Standard identifiers (11-bit) and extended identifiers (29-bit) are fully supported.

### 25.3.2 CAN block diagram

The CAN structure block diagram is shown below:

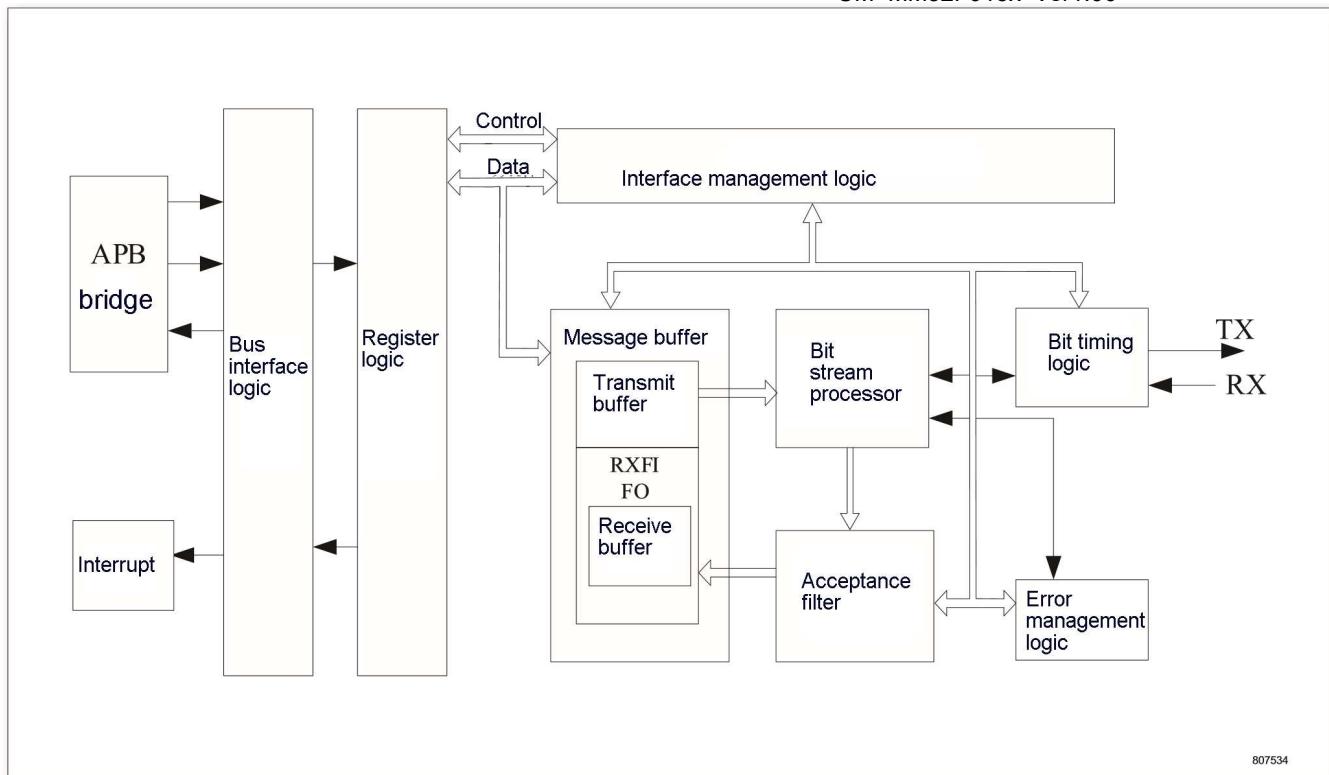


Figure 246. CAN structure block diagram

### 25.3.3 Interface Management Logic (IML)

The interface management logic interprets commands from the CPU, controls addressing of the CAN registers and provides interrupts and status information to the host microcontroller.

### 25.3.4 Transmit Buffer (TXB)

The transmit buffer is an interface between the CPU and the Bit Stream Processor (BSP) that is able to store a complete message for transmission over the CAN network. The buffer is 13 bytes long, written to by the CPU and read out by the BSP.

### 25.3.5 Receive Buffer (RXB, RXFIFO)

The receive buffer is an interface between the acceptance filter and the CPU that stores the received and accepted messages from the CAN-bus line.

The Receive Buffer (RXB) represents a CPU-accessible 13-byte window of the Receive FIFO (RXFIFO), which has a total length of 64 bytes.

With the help of this FIFO, the CPU is able to process one message while other messages are being received.

### 25.3.6 Acceptance Filter (ACF)

The acceptance filter compares the received identifier with the acceptance filter register contents and decides whether this message should be accepted or not. In the event of a positive acceptance test, the complete message is stored in the RXFIFO.

### 25.3.7 Bit Stream Processor (BSP)

The bit stream processor is a sequencer which controls the data stream between the transmit buffer, RXFIFO and the CAN-bus. It also performs the error detection, arbitration, stuffing and error handling on the CAN-bus.

### 25.3.8 Bit Timing Logic (BTL)

The bit timing logic monitors the serial CAN-bus line and handles the bus line-related bit timing. It is synchronized to the bit stream on the CAN-bus on a 'recessive-to-dominant' bus line transition at the beginning of a message (hard synchronization) and re-synchronized on further transitions during the reception of a message (soft synchronization). The BTL also provides programmable time segments to compensate for the propagation delay times and phase shifts and to define the sample point and the number of samples to be taken within a bit time.

### 25.3.9 Error Management Logic (EML)

The EML is responsible for the error confinement of the transfer-layer modules. It receives error announcements from the BSP and then informs the BSP and IML about error statistics.

## 25.4 CAN operating modes

CAN has two main operating modes:

- BasicCAN mode
- PeliCAN mode

Default mode upon reset is the BasicCAN mode.

The new PeliCAN operating mode is designed to support the full frame types of CAN 2.0B protocol specification. It also provides some enhanced features in a broader area.

The CAN\_CDR.7 register defines the CAN mode. If CDR.7 is 0, the CAN controller operates in the BasicCAN mode; else, it operates in the PeliCAN mode.

### 25.4.1 Differences between Basic CAN and PeliCAN mode

In the PeliCAN mode, the CAN controller appears with a re-organized register mapping with a lot of new features. In the PeliCAN mode, the complete CAN 2.0B functionality is supported (29-bit identifier). Main new features of the PeliCAN mode are:

- Reception and transmission of standard and extended frame
- Receive FIFO (64-byte)
- Single/dual acceptance filter with mask and code register for standard and extended frame
- Error counters with read/write access
- Programmable error warning limit
- Last error code register
- Error interrupt for each CAN-bus error
- Arbitration lost interrupt with detailed bit position
- Single-shot transmission (no re-transmission on error or arbitration lost)

- Listen only mode (monitoring of the CAN-bus, no acknowledge, no error flags)

## 25.5 CAN functional description

### 25.5.1 Basic CAN mode

#### Reset mode

The reset mode refers to the initialization mode. The Reset Request bit (CAN\_CR.0) is set (currently) after a hardware reset or when the bus state is set to '1'. If the software accesses these bits, their values will change and impact the next rising edge of the internal clock. Changes of the reset request bit are synchronized with the internal divided clock. Reading the Reset Request bit will reflect the synch state. The reset mode is mainly used for CAN communication parameter configuration. The core has different access rights of CAN registers in different modes.

After the Reset Request bit is set to '0', the CAN controller will wait for:

- One occurrence of bus-free signal (11 recessive bits) if the preceding reset request has been caused by a hardware reset or a CPU-initiated reset.
- 128 occurrences of bus-free, if the preceding reset request has been caused by a CAN controller initiated bus-off, before re-entering the bus-on mode; it should be noted that several registers are modified if the reset request bit was set.

#### Operating mode

Once the initialization is complete, the software must request the hardware to enter Normal mode to be able to start reception and transmission normally. In the Reset mode, if the '1 - 0' falling edge is transferred to the Reset bit, the CAN controller will return to the operating mode to handle the transmission and the reception of messages.

#### Sleep mode

The CAN controller enters the Sleep mode when the Sleep bit is set. There will be no bus activity and no interrupt is pending. A wakeup interrupt will be generated in the Sleep mode when at least one of the previously mentioned situations is disrupted. One situation is the Sleep bit is set to Low (wakeup), and then the bus goes to the activity state; the other situation is the interrupt is enabled. After wakeup, the clock is initiated and generates a wakeup interrupt. If the wakeup is caused by bus activity, this message can be received by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle state). Note that the Sleep mode cannot be set in the Reset mode. After the Reset mode is cleared, the Sleep mode comes into force on detection of bus idle state.

Table 72. Basic CAN register permission allocation

Offset	Segment	Operating Mode		ResetMode	
		Read	Write	Read	Write
00	Control	Control	Control	Control	Control
04		(FFH)	Command	(FFH)	Command
08		Status	-	Status	-
0C		(FFH)	-	Interrupt	-

Offset	Segment	Operating Mode		ResetMode	
		Read	Write	Read	Write
10	Transmit buffer	(FFH)	–	Acceptance code	Acceptance code
14		(FFH)	–	Acceptance mask	Acceptance mask
18		(FFH)	–	Bus timing 0	Bus timing 0
1C		(FFH)	–	Bus timing 1	Bus timing 1
20		(FFH)	–	–	–
24		Test	Test	Test	Test
28		Identifier (10 ~ 3)	Identifier (10 ~ 3)	(0xFF)	–
2C		Identifier (2 ~ 0) RTR and DLC	Identifier (2 ~ 0) RTR and DLC	(0xFF)	–
30		DATA1	DATA1	(0xFF)	–
34		DATA2	DATA2	(0xFF)	–
38		DATA3	DATA3	(0xFF)	–
3C		DATA4	DATA4	(0xFF)	–
40		DATA5	DATA5	(0xFF)	–
44	Receive buffer	DATA6	DATA6	(0xFF)	–
48		DATA7	DATA7	(0xFF)	–
4C		DATA8	DATA8	(0xFF)	–
50	Receive buffer	Identifier (10 ~ 3)			
54		Identifier (2 ~ 0) RTR and DLC			
58		DATA1	DATA1	DATA1	DATA1
5C		DATA2	DATA2	DATA2	DATA2
60		DATA3	DATA3	DATA3	DATA3
64		DATA4	DATA4	DATA4	DATA4
68		DATA5	DATA5	DATA5	DATA5
6C		DATA6	DATA6	DATA6	DATA6
70		DATA7	DATA7	DATA7	DATA7
74		DATA8	DATA8	DATA8	DATA8
78		(FFH)	–	(FFH)	–

Offset	Segment	Operating Mode		ResetMode	
		Read	Write	Read	Write
		Clock divider	Clock divider	Clock divider	Clock divider

Note: '(FFH)' represents all readings are 1; '—' represents no write access; the other represent 'writable'. The address offset is 0x7C. The 'clock divider' is used to select the BasicCAN or PeliCAN mode.

## 25.5.2 Peli CAN mode

### Reset mode

The reset mode refers to the initialization mode. The Reset Mode bit is set (currently) after a hardware reset or when the bus state is set to '1' (bus-off).

If this bit is accessed by software, a value change will become visible and takes effect first with the next rising edge of the internal clock which operates with 1/2 of the external oscillator frequency. Changes of the reset request bit are synchronized with the internal divided clock. Reading the Reset Request bit will reflect the synch state. After the Reset Mode bit is set to '0', the CAN controller will wait for:

1. One occurrence of bus-free signal (11 recessive bits), if the preceding reset request has been caused by a hardware reset or a CPU-initiated reset.
2. 128 occurrences of bus-free, if the preceding reset request has been caused by a CAN controller initiated bus-off, before re-entering the bus-on mode.

### Operating mode

Once the initialization is completed, the software must request the hardware to enter Normal mode to be able to start reception and transmission normally. In the Reset mode, if the '1 - 0' falling edge is generated on detection of the RM bit in the CAN\_MOD register, the CAN controller will return to the operating mode to handle the transmission and the reception of messages.

### Sleep mode

The CAN controller enters the Sleep mode when the Sleep Mode bit (CAN\_MOD.4) is set. There will be no bus activity and no interrupt is pending. A wakeup interrupt will be generated in the Sleep mode when at least one of the previously mentioned situations is disrupted. One situation is the Sleep bit is set to Low (wakeup), and then the bus goes to the activity state; the other situation is the interrupt is enabled. After wakeup, the clock is initiated and generates a wakeup interrupt.

If the wakeup is caused by bus activity, this message can be received by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle state). Note that the Sleep mode cannot be set in the Reset mode. After the Reset mode is cleared, the Sleep mode comes into force on detection of bus idle state.

### Self-Test mode

This mode is mainly used for test. Set the Self-Test mode bit (CAN\_MOD.2) to enter the Self-Test mode. This mode detects all nodes and applies the self reception request to nodes with no activity. The CAN controller will send the request successfully even if no acknowledge is received.

### Listen Only mode

This mode is mainly used for test. Set the Listen Only mode bit (CAN\_MOD.1) to enter the

Listen Only mode. In this mode, the CAN controller enters an error passive status. The message transmission is impossible. This mode can be applied to the software-driven bit rate detection. All the other functions can be used as in the Normal mode.

In Listen Only mode the CAN controller is not able to write dominant bits onto the CAN bus. Neither active error flags nor overload flags are written nor a positive acknowledge is given upon successful reception.

Note, before entering the Listen Only mode the Reset mode has to be entered.

Table 73. Peli CAN register permission allocation

Offset	Operating Mode			Reset Mode		
	Read	Write		Read	Write	
00	Mode	Mode	—	Mode	Mode	
04	(00H)	Command	—	(00H)	Command	
08	Status	—	—	Status	—	
0C	Interrupt	—	—	Interrupt	—	
10	Interrupt enable	—	—	Interrupt enable	Interrupt enable	
14	(00H)	—	—	(00H)	—	
18	Bus timing 0	—	—	Bus timing 0	Bus timing 0	
1C	Bus timing 1	—	—	Bus timing 1	Bus timing 1	
20	Reserved	—	—	—	—	
24	Detection	Detection	—	Detection	Detection	
28	Reserved	—	—	Reserved	—	
2C	Arbitration lost capture	—	—	Arbitration lost capture	—	
30	Error code capture	—	—	Error code capture	—	
34	Error warning limit	—	—	Error warning limit	Error warning limit	
38	RX error counter	—	—	RX error counter	RX error counter	
3C	TX error counter	—	—	TX error counter	TX error counter	
40	RX frame information SFF	RX framing error EFF	TX framing error SFF	TX framing error EFF	Acceptance code 0	Acceptance code 0
44	RX identifier 1	RX identifier 1	TX identifier 1	TX identifier 1	Acceptance code 1	Acceptance code 1
48	RX identifier 2	RX identifier 2	TX identifier 2	TX identifier 2	Acceptance code 2	Acceptance code 2

Offset	Operating Mode				Reset Mode	
	Read		Write		Read	Write
4C	RX data 1	RX identifier 3	TX data 1	TX identifier 3	Acceptance code 3	Acceptance code 3
50	RX data 2	RX identifier 4	TX data 2	TX identifier 4	Acceptance mask 0	Acceptance mask 0
54	RX data 3	RX data 1	TX data 3	TX data 1	Acceptance mask 1	Acceptance mask 1
58	RX data 4	RX data 2	TX data 4	TX data 2	Acceptance mask 2	Acceptance mask 2
5C	RX data 5	RX data 3	TX data 5	TX data 3	Acceptance mask 3	Acceptance mask 3
60	RX data 6	RX data 4	TX data 6	TX data 4	Reserved	–
64	RX data 7	RX data 5	TX data 7	TX data 5	Reserved	–
68	RX data 8	RX data 6	TX data 8	TX data 6	Reserved	–
6C	(FIFO RAM)	RX data 7	–	TX data 7	Reserved	–
70	(FIFO RAM)	RX data 8	–	TX data 8	Reserved	–
74	RX message counter		–	RX message counter		–
78	RX buffer start address		–	RX buffer start address		RX buffer start address
7C	Clock divider		Clock divider	Clock divider		Clock divider
80	Internal RAM address 0 (FIFO)		–	Internal RAM address 0	Internal RAM address 0	
84	Internal RAM address 1 (FIFO)		–	Internal RAM address 1	Internal RAM address 1	
...	...		...	...	...	
17C	Internal RAM address 63 (FIFO)		–	Internal RAM address 63	Internal RAM address 63	
180	Internal RAM address 64 (TX buffer)		–	Internal RAM address 64	Internal RAM address 64	
...	...		...	...	...	
1B0	Internal RAM address 76 (TX buffer)		–	Internal RAM address 76	Internal RAM address 76	
1B4	Internal RAM address 77 (idle)		–	Internal RAM address 77	Internal RAM address 77	

Offset	Operating Mode		Reset Mode	
	Read	Write	Read	Write
1B8	Internal RAM address 78 (idle)	–	Internal RAM address 78	Internal RAM address 78
1BC	Internal RAM address 79 (idle)	–	Internal RAM address 79	Internal RAM address 79
1C0	(00H)	–	(00H)	–
...	...	...	...	...
1FC	(00H)	–	(00H)	–

### 25.5.3 Transmission handling

A transmission of a message is done autonomously by the CAN controller according to the CAN protocol specification. The microcontroller configures the identifier, data length and the message to be transmitted; then it sets the flag “Transmit Request” in the command register. As long as the CAN controller is transmitting a message, the Transmit Buffer is locked for writing. Thus the microcontroller has to check the “Transmit Buffer Status” flag (TBS) of the Status Register, if a new message can be placed into the Transmit Buffer.

A single message transfer is initiated by setting the command bits CMR.0 and CMR.1 simultaneously. If the first attempt fails, due to an arbitration loss or an error, the hardware will not restart the message transmission (Single Shot Transmission). If only the CMR.0 is set, the message will be re-transmitted if it is not transmitted successfully. In the Self Test mode, a self reception message is immediately transmitted by setting the command bits CMR.4 and CMR.1.

#### Abort

The transmission of a message, which was requested, may be aborted using the ‘Abort Transmission’ command by setting the corresponding bit in the Command Register.

Setting the AT bit in the CAN\_CMR register will abort the transmission request.

The abort transmission bit is used when the CPU requires the suspension of the previously requested transmission, e.g. to transmit a more urgent message before. A transmission already in progress is not stopped. In order to see if the original message had been either transmitted successfully or aborted, the transmission complete status bit should be checked. This should be done after the transmit buffer status bit has been set to 1 or a transmit interrupt has been generated.

It should be noted that a transmit interrupt is still generated even the message is aborted when the transmit buffer status bit turns to ‘Released’.

If the transmission request was set to 1 in a previous command, it cannot be canceled by setting the transmission request bit to 0. Instead it can be canceled by setting the Abort Transmission bit to 0.

### 25.5.4 Reception management

The reception of messages is done autonomously by the CAN controller. Received messages are placed into the Receive Buffer. A message, ready to be transferred to the microcontroller, is signalled by the Receive Buffer Status flag 'RBS' of the Status Register and by a Receive Interrupt flag 'RI'.

#### Polling Controlled Reception

The microcontroller reads the Status Register of the CAN controller, checking if the Receive Buffer Status flag (RBS) indicates, that one message has been received. The RBS flag indicates "1", i.e., one or more messages have been received. The microcontroller gets the message from CAN and sends a Release Receive Buffer command afterwards by setting the corresponding flag 'RRB' in the Command Register.

#### Interrupt Controlled Reception

The interrupt enable flags are located in the CAN Control Register (for the BasicCAN mode) or in the Interrupt Enable Register (for the PeliCAN mode). If the CAN controller has received a message, which has passed the acceptance filter and has been placed into the Receive FIFO, a receive interrupt is generated. Entering the ISR, the microcontroller collects the message, and sends a Release Receive Buffer command afterwards by setting the corresponding flag 'RRB' in the Command Register.

#### Overrun

In case the Receive FIFO is full but another message is being received, a Data Overrun is signaled to the microcontroller by setting the Data Overrun Status in the Status Register (if enabled). The overrun state can be cleared by setting the CMR.3 bit to 1.

The ERB bit in the CAN\_CMR register is used to clear FIFO: If ERB is enabled, the hardware will automatically clear the Receive FIFO if an overrun happens to it.

#### Valid message

A received message is considered as valid when it has been received correctly according to the CAN protocol (no error until the last but one bit of the EOF field) and it passed through the identifier filtering successfully.

The RRB bit in the CAN\_CMR register is used to clear the data overrun condition indicated by the Data Overrun Status bit.

### 25.5.5 Identifier filtering

In the CAN protocol, the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. On message reception, a receiver node decides - depending on the identifier value - whether the software needs the message or not. If the message is needed, it is copied into the internal BUFFER. If not, the message must be discarded without intervention by the software.

The stand-alone CAN controller is equipped with a versatile acceptance filter, which allows an automatic check of the identifier and data bytes. Using these effective filtering methods, messages or a group of messages not valid for a certain node can be prevented from being stored in the Receive Buffer. Thus it is possible to reduce the processing load of the microcontroller.

The filter is controlled by the acceptance code and mask registers according to given algorithms. The received data is compared bitwise with the value contained in the Acceptance Code register.

The Acceptance Mask Register defines the bit positions, which are relevant for the comparison (0 = relevant, 1 = not relevant). For accepting a message, all relevant received bits have to match the respective bits in the Acceptance Code Register.

### Acceptance filter in BasicCAN mode

The filter is controlled by two registers – Acceptance Code Register (ACR) and Acceptance Mask Register (AMR). The 8 most significant bits of the identifier of the CAN message are compared to the values contained in these registers. Thus several groups of identifiers can be defined to be accepted for any node.

Example: The Acceptance Code register (ACR) contains:

The Acceptance Code register (ACR) contains:  
0 | 1 | 1 | 1 | 0 | 0 | 1 | 0

The Acceptance Mask register (AMR) contains:  
0 | 0 | 1 | 1 | 1 | 0 | 0 | 0

Messages with the following 11-bit identifiers are accepted  
0 | 1 | X | X | X | 0 | 1 | 0 | X | X | X

X=not relevant  
ID.10

423550

Figure 247. Example of CAN identifier acceptance

At the bit positions containing a “1” in the Acceptance Mask register, any value is allowed in the composition of the identifier. The same is valid for the three least significant bits. Thus 64 different identifiers are accepted in this example. The other bit positions must be equal to the values in the Acceptance Code register

### Acceptance filter in PeliCAN mode

With the help of the acceptance filter the CAN controller is able to allow passing of received messages to the RXFIFO only when the identifier bits of the received message are equal to the predefined ones within the acceptance filter registers. In PeliCAN mode, the acceptance filter is defined by Acceptance Code registers (ACR<sub>n</sub>) and Acceptance Mask registers (AMR<sub>n</sub>). The bit pattern of the message to be received is defined within the Acceptance Code registers. The corresponding Acceptance Mask register allows certain bits to be defined as "not affected", which can be Any value. Two different filtering modes can be selected through bit 3 in the mode register:

- Single filter mode<sup>(1)</sup>
- Dual filter mode<sup>(0)</sup>

### Single filter configuration

In this filter configuration one long filter (4-bytes) could be defined. The bit correspondences between the filter bytes and the message bytes depend on the currently received frame format.

**Standard Frame:** If the received message is in the standard frame format, only the first two data bytes are used to store the complete identifier including the RTR bit in the acceptance filtering. Messages may also be accepted if there are no data bytes existing due to a set RTR bit or if there is none or only one data byte because of the corresponding data length code. For a successful reception of a message, all single bit comparisons have to signal acceptance.

Note that the 4 least significant bits of ACR1 and AMR1 are not used. In order to be compatible with future products, these bits should be programmed to be 'not affected' by setting AMR1.3, AMR1.2, AMR1.1 and AMR1.0 to 1.

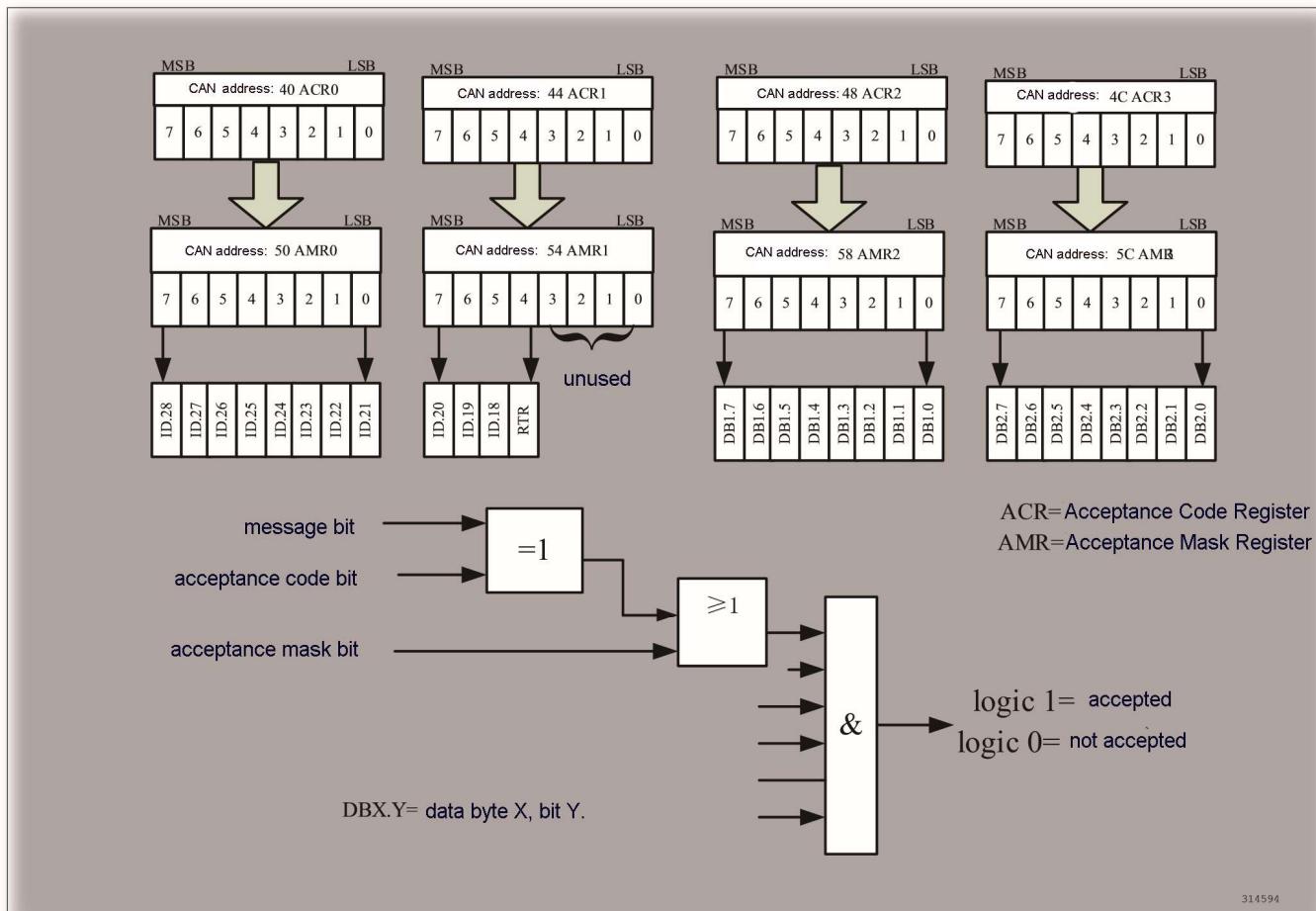


Figure 248. Single filter configuration, receiving standard frame messages

Extended frame: if an extended frame format message is received, the complete identifier including the RTR bit is used for acceptance filtering.

For a successful reception of a message, all single bit comparisons have to signal acceptance.

It should be noted that the 2 least significant bits of AMR3 and ACR3 are not used. These bits should be programmed to be 'not affected' by setting AMR3.1 and AMR3.0 to logic 1.

314594

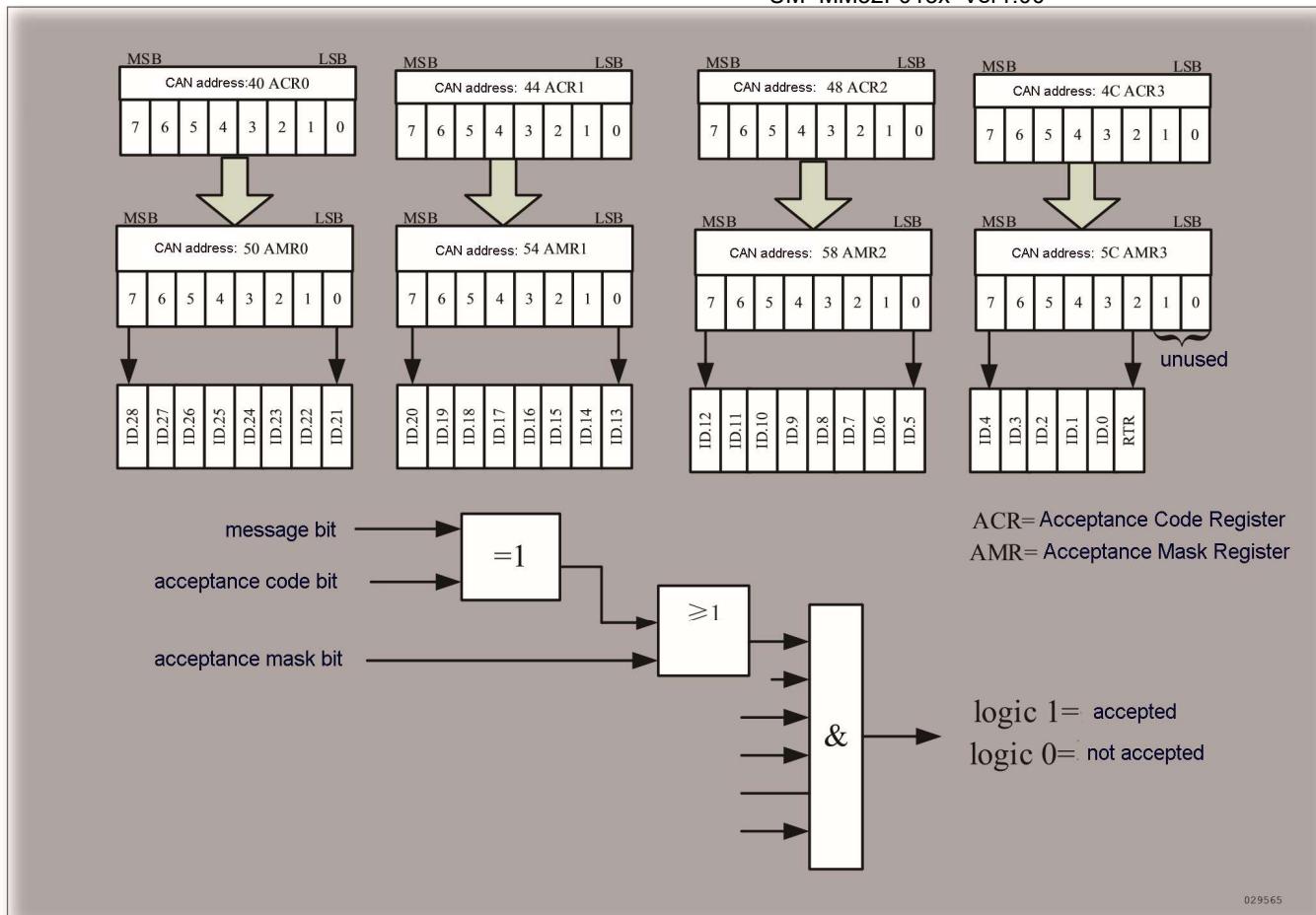


Figure 249. Single filter configuration, receiving extended frame messages

### Dual filter configuration

In this filter configuration two short filters could be defined. A received message is compared with both filters to decide, whether this message should be copied into the receive buffer or not. If at least one of the filters signals an acceptance, the received message becomes valid. The bit correspondences between the filter bytes and the message bytes depends on the currently received frame format.

**Standard frame:** if a standard frame message is received, the two defined filters are looking different. The first filter compares the complete standard identifier including the RTR bit and the first data byte of the message. The second filter just compares the complete standard identifier including the RTR bit.

For a successful reception of a message, all single bit comparisons of at least one complete filter have to signal acceptance. In case of a set RTR bit or a data length code of logic 0, no data byte is existing. Nevertheless a message may pass filter 1, if the first part up to the RTR bit signals acceptance.

If no data byte filtering is required for filter, the four least significant bits of AMR1 and AMR3 have to be set to '1' (not affected).

Then both filters are working identically using the standard identifier range including the RTR bit.

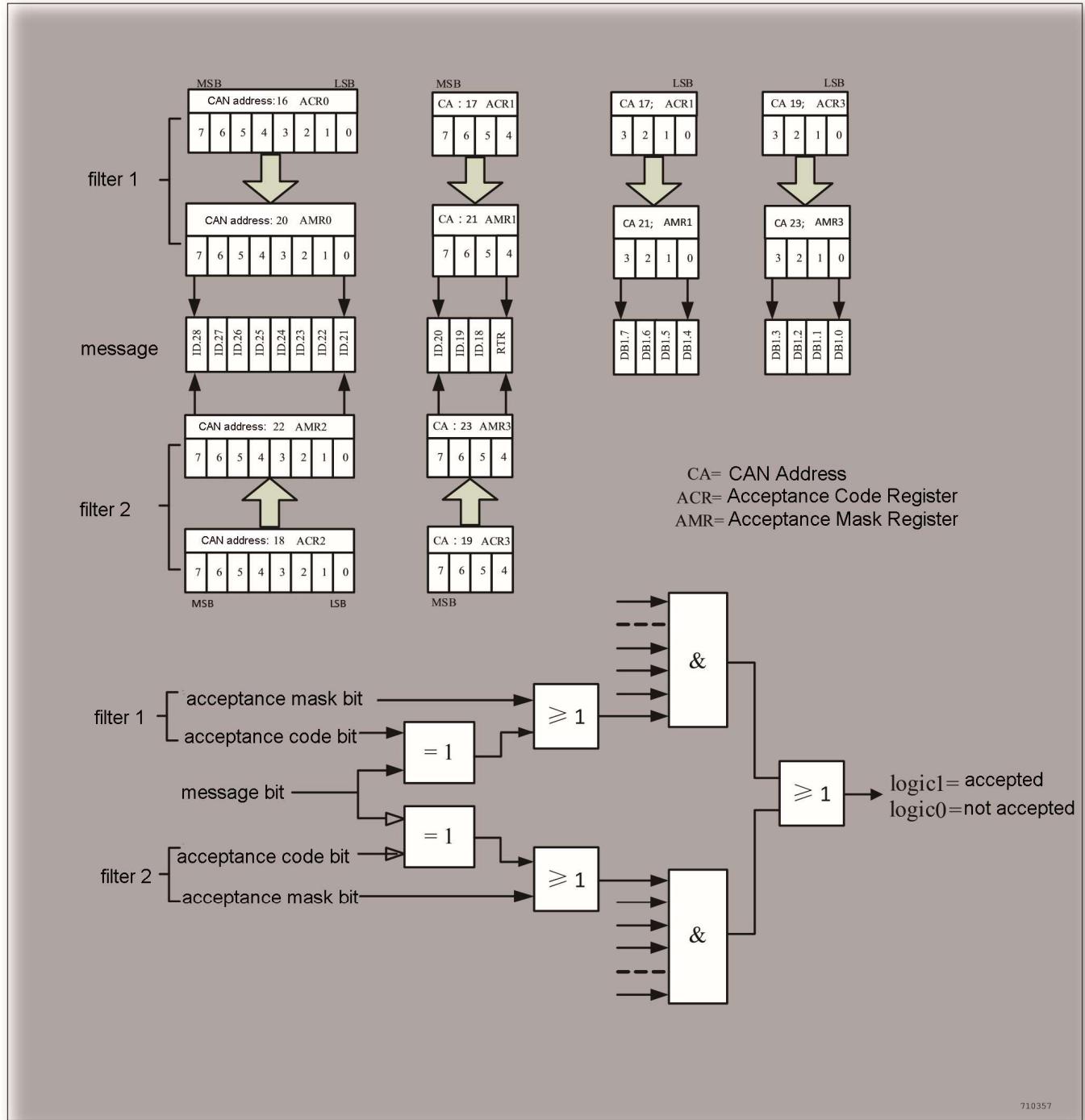


Figure 250. Dual filter configuration, receiving standard frame messages

Extended frame: if an extended frame message is received, the two defined filters are looking identically. Both filters are comparing the first two bytes of the extended identifier range only.

For a successful reception of a message, all single bit comparisons of at least one complete filter have to indicate acceptance.

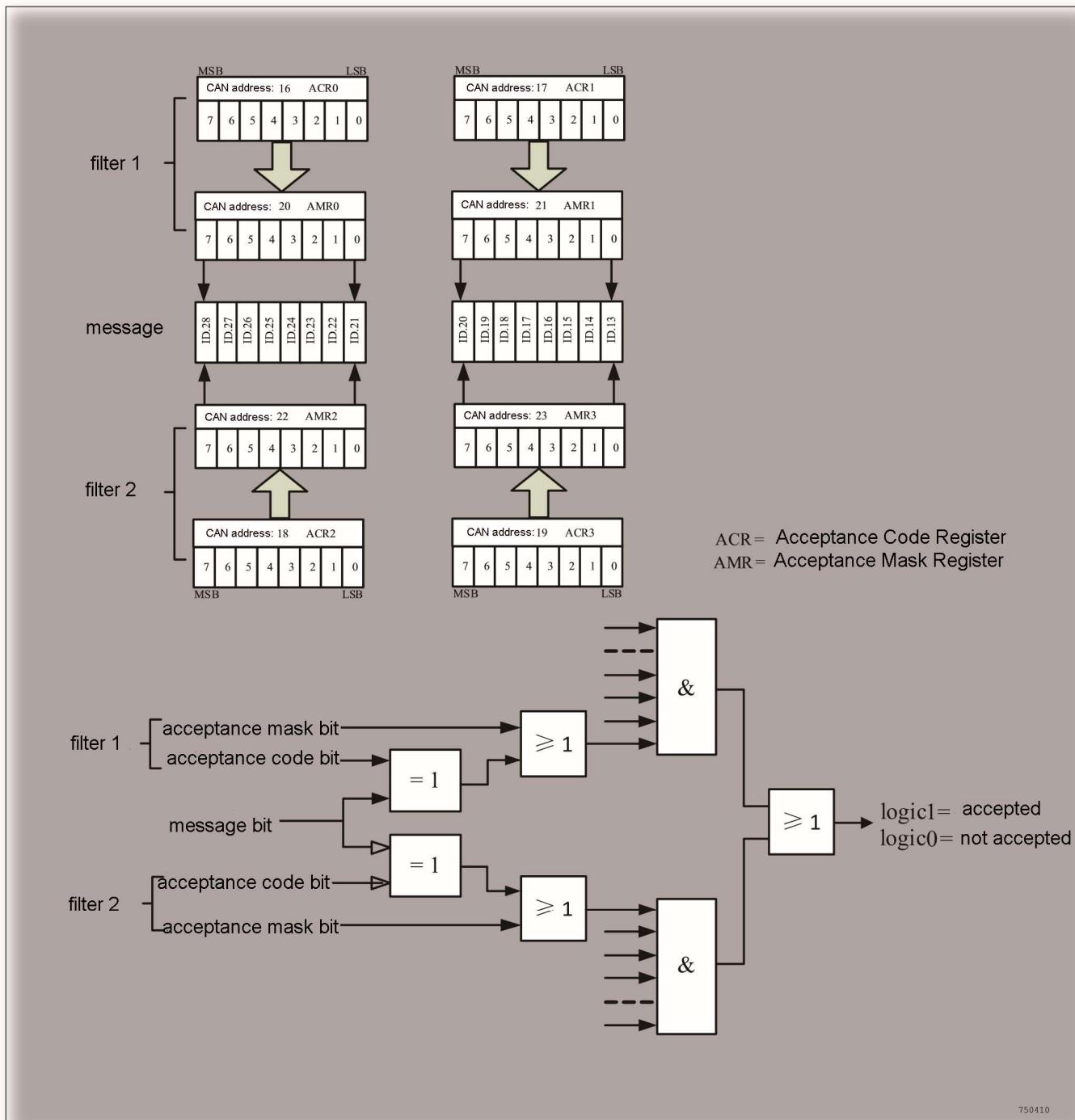


Figure 251. Dual filter configuration, receiving extended frame messages

Example 1: Suppose a standard frame format message is to be filtered in PeliCAN mode; this can be done through a long filter (Single filter mode).

Acceptance Code registers (ACRn) and Acceptance Mask registers (AMRn) contain:

n	0	1 (upper 4 bits)	2	3
ACRn	01XX X010	XXXX	XXXX XXXX	XXXX XXXX
AMRn	0011 1000	1111	1111 1111	1111 1111
Received message (ID28 ~ ID.18, RTR)	01xx x010 xxxx			

'X' = irrelevant, 'x' = any value, only the upper 4 bits of ACR1 and AMR1 are used

Example 2: Suppose the following 2 messages with a standard frame identifier have to be accepted without any further decoding of the identifier bits. Data and remote frames have to be received correctly. Data bytes are not involved in the acceptance filtering.

message 1: (ID.28)1011 1100 101(ID.18)  
message 2: (ID.28)1111 0100 101(ID.18)

Using the Single Filter mode results in accepting four messages and not only the requested two:

n	0	1 (upper 4 bits)	2	3
ACRn	1X11 X100	101X	XXXX XXXX	XXXX XXXX
AMRn	0100 1000	0001	1111 1111	1111 1111
Received message (ID28 ~ ID.18, RTR)	1011 0100 101x 1111 0100 101x (message 2) 1011 1100 101x (message 1) 1111 1100 101x			

('X' = irrelevant, 'x' = any value, only the upper 4 bits of ACR1 and AMR1 are used)

This result does not meet the request for receiving 2 messages without any further decoding.

Using the Dual Filter mode gives the correct result:

	Filter 1			Filter 2	
n	0	1	3 lower 4 bits	2	3 upper 4 bits
ACRn	1011 1100	101X XXXX	... XXXX	1111 0100	101X ...
AMRn	0000 0000	0001 1111	... 1111	0000 0000	0001 ...
Received message (ID28 ~ ID.18, RTR)	1011 1100 101X (message 1)			1111 0100 101X (message 2)	

('X' = irrelevant, 'x' = any value)

Message 1 is accepted by filter 1 and message 2 by filter 2. As messages are accepted and stored into the receive FIFO if they are accepted at least by one of the two filters, this solution meets the request.

Example 3: In this example a group of messages with an extended frame Identifier are filtered using a long single acceptance filter.

n	0	1	2	3 (upper 6 bits)
ACRn	1011 0100	1011 000X	1100 XXXX	0011 0XXX
AMRn	0000 0000	0001 0001	0000 1111	0000 0111
Received message (ID28 ~ ID.18, RTR)	1011 0100 101x 000x 1100 xxxx 0011 0x			

('X' = irrelevant, 'x' = any value, only the upper 6 bits of ACR1 and AMR1 are used)

Example 4: There are systems, which use Standard Frames only and identify messages by the 11-bit identifier and the first two data bytes.

Such a protocol is used, e.g., in the DeviceNet, where the first two data bytes define a message header and the fragmentation protocol, if messages contain more than 8 data bytes. For this system type, the CAN controller can filter two data bytes in single filter mode and one data byte in dual filter mode in addition to the 11-bit identifier and the RTR-bit.

Using the Dual Filter mode, the following example shows effective filtering of messages in such a system:

	Filter 1			Filter 2	
n	0	1	3 lower 4 bits	2	3 upper 4 bits
ACRn	1110 1011	0010 1111	... 1001	1111 0100	XXX0 ...
AMRn	0000 0000	0000 0000	... 0000	0000 0000	1110 ...
Received message (ID28 ~ ID.18, RTR)	1110 1011 0010 + 1111 ... 1001 Identifier RTR + first data byte			1111 0100 Identifier	xxx0 RTR

('X' = irrelevant, 'x' = any value)

- Filter 1 is used for filtering messages with:
  - Identifier '11101011001'
  - RTR = '0' i.e. data frames
  - data byte '1111001' (this means e.g. for the DeviceNet: all fragments for one message are filtered)
- Filter 2 is used for filtering a group of 8 messages with:
  - identifiers '11110100 000' through '11110100111'
  - RTR = '0' i.e. data frames

### Filter group mode configuration

There are 20 filter groups in total. Some filter groups are enabled and disabled by configuring corresponding CAN\_FGAx ( $x = 0 \sim 2$ ). Unused filter groups in the application should remain disabled. The single filter/dual filter mode of the filter group is selected by configuring the FGAx( $x = 0 \sim 19$ ).

There are 20 groups of Acceptance Code registers and Acceptance Mask registers:

AMR0, AMR1, AMR2, AMR3, ACR0, ACR1, ACR2.

Each group of filters are used in the same way. These groups can be used individually or in a combined manner.

### 25.5.6 Message storage

The data to be transmitted on the CAN bus is loaded into the memory area of the CAN controller, called "Transmit Buffer". The data received from the CAN bus is stored in the memory area of the CAN controller, called "Receive Buffer".

These buffers contains 2, 3 or 5 bytes for the identifier and frame information (dependent on mode and frame type) and up to 8 data bytes.

### BasicCAN mode

The buffers are 10-bytes long

- 2 identifier bytes
- 8 data bytes

Table 74. Rx- and Tx-Buffer in BasicCAN mode

Relevant CAN Offset		Register		Composition and Remarks
TX (hexadecimal)	RX (hexadecimal)	TX	RX	
28	50	CAN_TXIDR1	CAN_RXIDR1	8 identifier bits
2C	54	CAN_TXIDR2	CAN_RXIDR2	3 identifier bits, 1 remote transmission request bit, 4 bits for the data length code, indicating the amount of data bytes
30	58	CAN_TXDR1	CAN_RXDR1	up to 8 data bytes as indicated by the data length code
34	5C	CAN_TXDR2	CAN_RXDR2	
38	60	CAN_TXDR3	CAN_RXDR3	
3C	64	CAN_TXDR4	CAN_RXDR4	
40	68	CAN_TXDR5	CAN_RXDR5	
44	6C	CAN_TXDR6	CAN_RXDR6	
48	70	CAN_TXDR7	CAN_RXDR7	
4C	74	CAN_TXDR8	CAN_RXDR8	

## PeliCAN mode

The buffers are 13-bytes long

- 1 byte for frame information
- 2 or 4 identifier bytes (Standard Frame or Extended Frame)
- up to 8 data bytes

## Transmit Buffer

The global layout of transmit buffers is shown below. Please be sure to distinguish between the standard frame format (SFF) and the extended frame format (EFF) configuration.

The transmit buffer allows definition of one transmit message with up to eight data bytes.

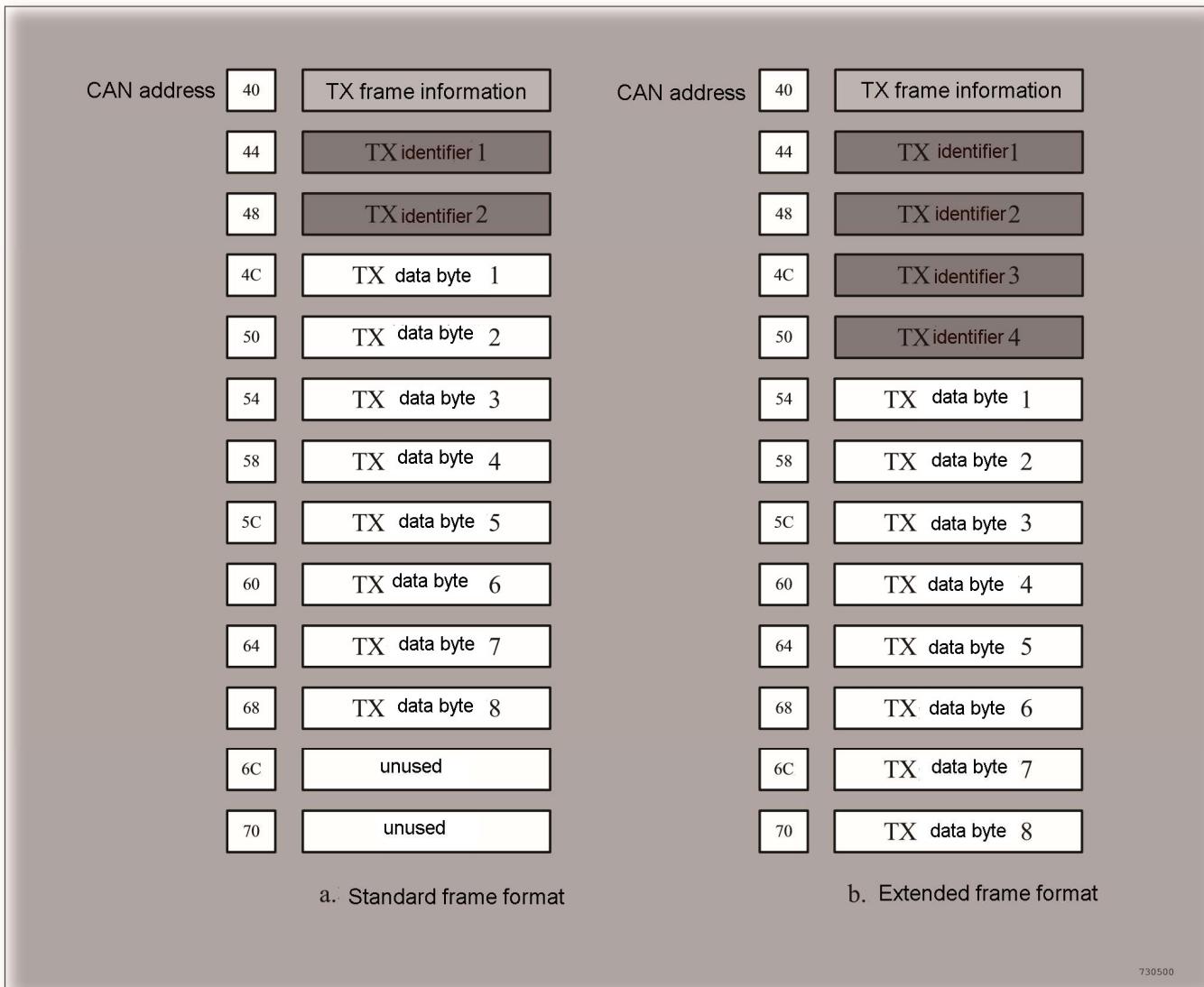


Figure 252. Transmit buffer layout for standard and extended frame format configurations

The FF bit in the frame information decides whether the CAN controller will send the EFF or SFF message.

## Receive Buffer

The global layout of the receive buffer is very similar to the transmit buffer. The receive buffer is the accessible part of the RXFIFO and is located in the range between CAN address 40 and 70. Each message is subdivided into a descriptor and a data field.

Note: the received data length code located in the frame information byte represents the real sent data length code, which may be greater than 8 (depends on sender). Nevertheless the maximum number of received data bytes is 8. This should be taken into account by reading a message from the receive buffer.

As illustrated in the figure below, the RXFIFO has space for 64 message bytes in total. The number of messages that can be stored at one time depends on the length of the individual messages. If there is not enough space for a new message within the RXFIFO, the CAN controller generates a data overrun condition. At this moment, the message is valid and the acceptance test is positive. A message which is partly written into the RXFIFO, when the data overrun condition occurs, is deleted. This situation is indicated to the CPU via the status register and the data overrun interrupt (interrupt enabled).

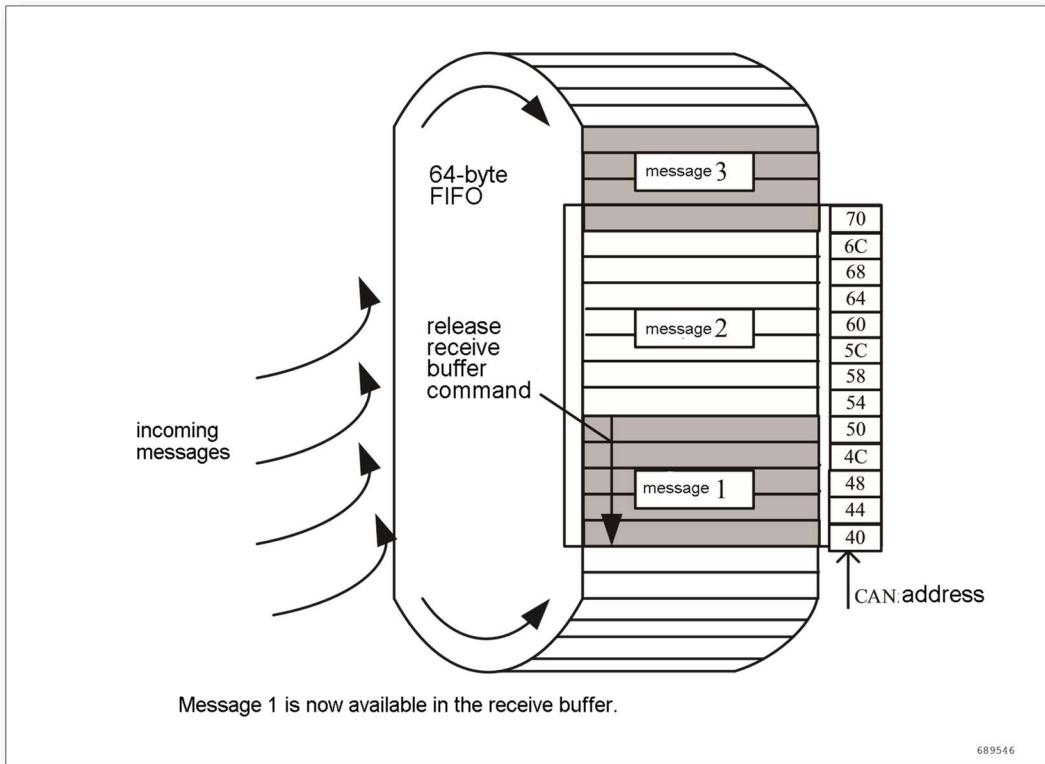


Figure 253. Transmit buffer layout for standard and extended frame format configurations

### 25.5.7 Error management

Depending on the value of the error counters, each CAN controller can operate in one of three possible error states: error active, error passive or bus-off. The CAN controller is error active if both error counters are between 0 to 127. In this case, an active error flag (6 dominant bits) is generated. The CAN controller is error passive if one of the error counters is between 128 and 255. A passive error flag (6 recessive bits) is generated upon detection of an error condition in this case. If the Transmit Error Counter is greater than 255, the bus-off status is reached. In this state, the reset request bit is set automatically and the CAN controller can not influence the bus. The bus-off state can only be terminated with the microcontroller command 'Reset Request = 0'. This will start the bus-off recovery counting where the Transmit Error Counter is used to count 128 occurrences of a bus free signal. At the end of this counting, both error counters are 0 and the device is error active again.

#### Error counter

As described above, the error states of the CAN controller are directly related to the values of the Transmit and Receive Error Counters.

To allow a deep look inside into the error confinement and to support an enhanced error analysis with the CAN controller, the CAN controller provides readable error counters. Additionally, in Reset Mode a write access to both error counters is allowed.

#### Error interrupt

Three interrupt sources have been implemented to signal error conditions to the microcontroller. Each interrupt can be enabled separately in the Interrupt Enable register.

- Bus Error Interrupt: This interrupt is generated upon any error condition on the CAN bus.
- Error Warning Interrupt: The Error Warning Interrupt is generated if the error warning limit is passed. Furthermore, it is generated if the CAN controller enters the bus-off state and upon re-

entry into error active state. The error warning limit of the CAN controller is programmable in reset mode. The default value upon reset is 96.

- Error Passive Interrupt: If the error status changes from error active to error passive or vice versa, an error passive interrupt is signaled.

### Error Code Capture

The CAN controller performs the full error confinement specified in the CAN2.0B specification. As in every CAN controller, the whole process of handling errors is executed fully automatically. However, to provide the user with additional details about a certain error condition, the CAN controller contains the Error Code Capture function. Whenever a CAN bus error occurs, the corresponding bus error interrupt is forced. At the same time, the current bit position is captured into the Error Code Capture register. The captured data is fixed until the host controller has read it. From now on the capture mechanism is activated again. The register contents distinguishes four different types of errors: form errors, stuff errors, bit errors and other errors. As shown in the figure below, the register additionally indicates whether the error occurred during reception or transmission of a message. Five bits in this register indicate the erroneous bit position in the CAN frame, see also the following tables and the data sheet for more details.

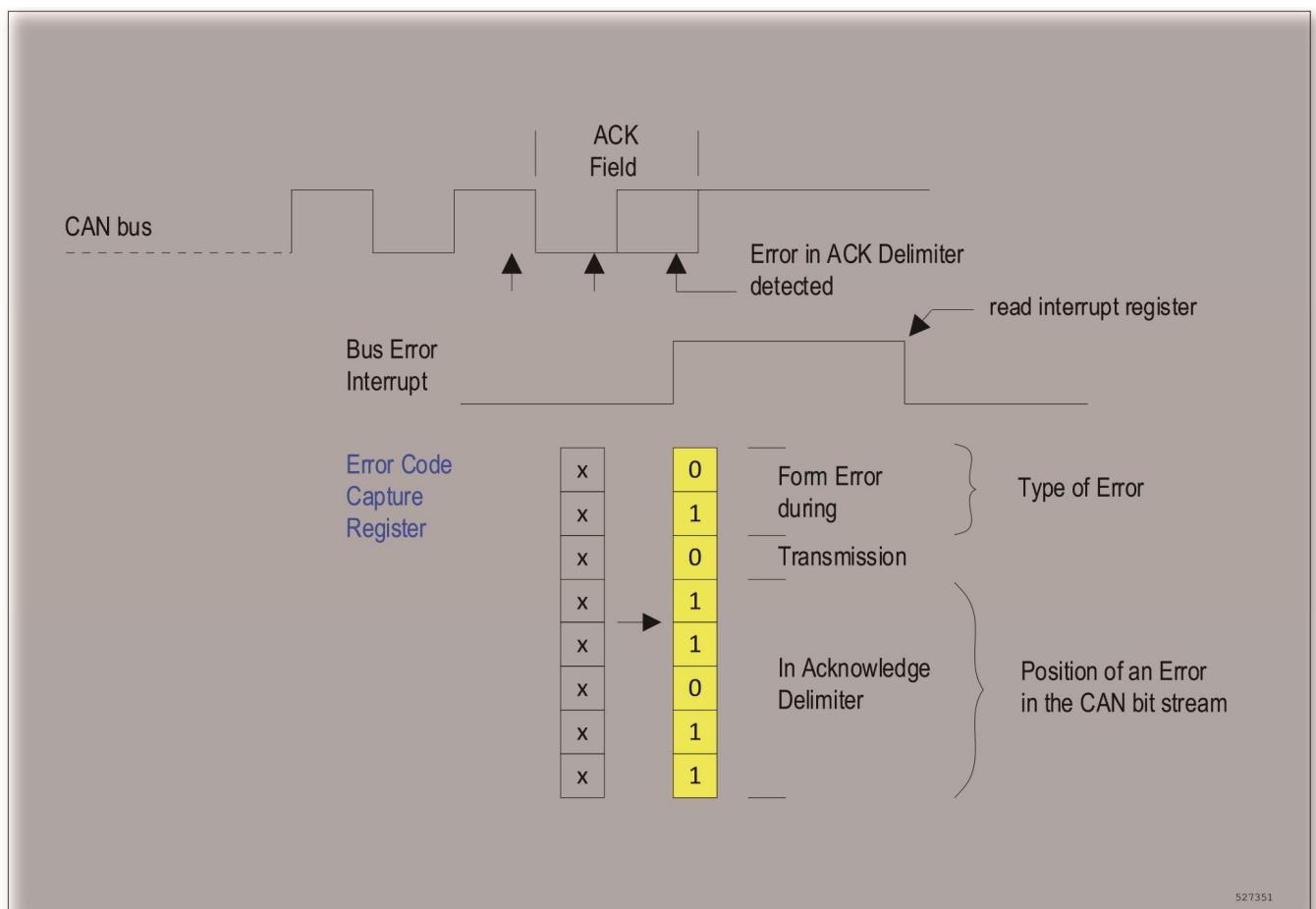


Figure 254. Example for the Error Code Capture Function

As defined in the CAN specification, every single bit on the CAN bus can only have special types of errors. The next two tables show all possible errors during transmission and reception of CAN messages. The left part contains the position and the type of an error, captured by the Error Code Capture register. The right part of each table is a translation from error code into an upper level error description and can be derived directly from the register contents. With the help of these tables, further information concerning error counter change and the erroneous state at the transmit and receive pins of the device can be derived. While using this table, e.g., in the error analysis software, it is possible to analyze every single error situation in

detail. The information about type and position of CAN errors can be used for error statistics and system maintenance or for corrective actions during system optimization.

Table 75. Possible errors during reception

Position of an Error in the CAN Bit Stream	Type of Error	RX Error Count	Error Code Capture	Description
Identifier SRR, IDE and RTR bits				
Reserved Bits				
Data Length Code	Stuff	+1	5 consecutive bits at the same level are received	—
Data Field				
CRC Sequence				
CRC Delimiter	Form Stuff	+1 +1	Rx = dominant more than 5 consecutive bits at the same level are received	bit has to be recessive
Acknowledge Slot	bit	+1	Rx = dominant or CRC error is detected	Critical bus timing or bus length CRC Sequence is incorrect
Acknowledge Delimiter	Format	+1	Rx = dominant or CRC error is detected	Critical bus timing or bus length CRC Sequence is incorrect
End of Frame	Format Other	+1 ±0	RX = dominant in first 6 bits RX = dominant in last bit	reaction: overload flag will be sent, data duplication is possible if transmitter starts re-transmission
Intermission	Other	±0	RX = dominant	reaction: overload flag will be sent by receiver
Active Error Flag	Bit	+8	TX = dominant but RX = recessive	can't write dominant bit
Tolerate Dominant Bits	Other	+8	TX = dominant but RX = recessive	can't write dominant bit
Error Delimiter	Format Other	+8	TX = dominant but RX = recessive	can't write dominant bit
Overload Flag	Bit	+8	TX = dominant but RX = recessive	can't write dominant bit

Table 76. Possible errors during transmission

Position of an Error in the CAN Bit Stream	Type of Error	TX Error Count	Error Code Capture	Description
Start Of Frame	Bit	+8	TX = dominant but RX =	can't write dominant bit

			recessive	
Identifier	Bit Stuff	+ 8 ±0	TX = dominant but RX = recessive TX = recessive but RX = dominant	can't write dominant bit —
SRR Bit	Bit Stuff	+ 8 ±0	TX = dominant but RX = recessive TX = recessive but RX = dominant	can't write dominant bit —
IDE and RTR Bit	Bit Stuff	+ 8 ±8	TX = dominant but RX = recessive TX = recessive but RX = dominant	can't write dominant bit —
Reserved Bits Data Length Code Data Field CRC Sequence	Bit	+8	TX = dominant but RX = recessive	can't write dominant bit
CRC Delimiter	Format	+8	RX = dominant	bit has to be recessive
Acknowledge Slot	Other Other	+ 8 ±0	RX = recessive (error active) RX = recessive (error passive)	no acknowledge no acknowledge, node is probably alone on the bus
Acknowledge Delimiter	Format	+8	RX = dominant	Critical bus timing or bus length
End of Frame	Format Other	+8 +8	RX = dominant in first 6 bits RX = dominant in last bit	— frame has already been received by some nodes, retransmission may result in data duplication in receivers
Intermission	Other	±0	RX = dominant	overload flag from 'old' CAN controllers
Active Error Flag Overload Flag	Bit	+8	TX = dominant but RX = recessive	can't write dominant bit
Tolerate Dominant Bits	Format	+8	Rx = dominant for more than 7 bit times after active error flag or overload flag	—
Error Delimiter	Format Other	+8 ±0	RX = dominant in first 7 bits Bit RX = dominant in last bit of delimiter	—
Passive Error Flag	Other	+ 8	RX = dominant (error passive)	no acknowledge received, node is not alone on the bus.

### Bus-off recovery

If the Transmit Error Counter is greater than 255, the bus-off status is reached. The bus state bit is then set. In this state, the reset request bit is set automatically and the CAN controller can not influence the bus. An error interrupt is generated if it is enabled. This state will last until the CPU resets the reset request bit. Once this is completed the CAN controller will wait the minimum protocol-defined time (128 occurrences of the bus-free signal).

## 25.5.8 Bit timing

The bit timing logic monitors the serial CAN bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

Its operation may be explained simply by splitting nominal bit time into three segments as follows:

- Synchronization segment ( $t_{SYNCSEG}$ ): a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ( $1 \times t_{CAN}$ ).
- Time segment 1 ( $t_{TSEG1}$ ): defines the location of the sample point. It includes the PROP\_SEG and PHASE\_SEG1 of the CAN standard.

Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.

- Time segment 2 ( $t_{TSEG2}$ ): defines the location of the transmit point. It represents the PHASE\_SEG2 of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The period of CAN system clock  $t_{SCL}$  is programmable and determines the corresponding bit timing.

The CAN system clock is calculated in the following formula:  $t_{SCL} = 2 \times t_{CLK} \times (BRP + 1)$ , where  $t_{CLK}$  = APB1 frequency period

The Resynchronization Jump Width (SJW) defines an upper bound to the amount of lengthening or shortening of the bit segments. To compensate for phase shifts between clock oscillators of different bus controllers, all bus controllers must re-synchronize on any relevant signal edge of the current transmission. The synchronization jump width defines the maximum number of clock cycles a bit period may be shortened or lengthened by one re-synchronization.

$t_{SJW} = t_{SCL} \times (SJW + 1)$  Time segment 1 (TSEG1) and time segment 2 (TSEG2) determine the number of clock cycles per bit period and the location of the sample point, where:

$$t_{SYNCSEG} = 1 \times t_{SCL}$$

$$t_{TSEG1} = t_{SCL} \times (TESG1 + 1)$$

$$t_{TSEG2} = t_{SCL} \times (TESG2 + 1)$$

A valid transition is defined as the first transition in a bit time from dominant to recessive bus level provided the CAN itself does not send a recessive bit. If a valid transition is detected in time segment 1 ( $t_{TSEG1}$ ) instead of synchronization segment ( $t_{SYNCSEG}$ ),  $t_{TSEG1}$  is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid transition is detected in time segment 2 ( $t_{TSEG2}$ ) instead of  $t_{SYNCSEG}$ ,  $t_{TSEG2}$  is shortened by up to SJW so that the sample point is moved earlier.

As a safeguard against programming errors, the configuration of the Bit Timing register (CAN\_BTR) is only possible while CAN is in the initialized state.

$$\text{CAN baud rate} = \text{APB1}/(2*(BRP + 1)*(TSEG1 + 1 + TSEG2 + 1 + 1));$$

## 25.5.9 Arbitration lost

On arbitration lost, the corresponding arbitration lost interrupt is forced, if enabled. At the same time, the current Bit position of the bit stream processor is captured into the Arbitration Lost Capture register. The content within this register is fixed until the users software has read out its contents once.

The capture mechanism is then activated again.

The corresponding interrupt flag located in the interrupt register is cleared during the read access to the interrupt register. A new arbitration lost interrupt is not possible until the arbitration lost capture register is read out once.

The figure below is the arbitration lost bit number interpretation.

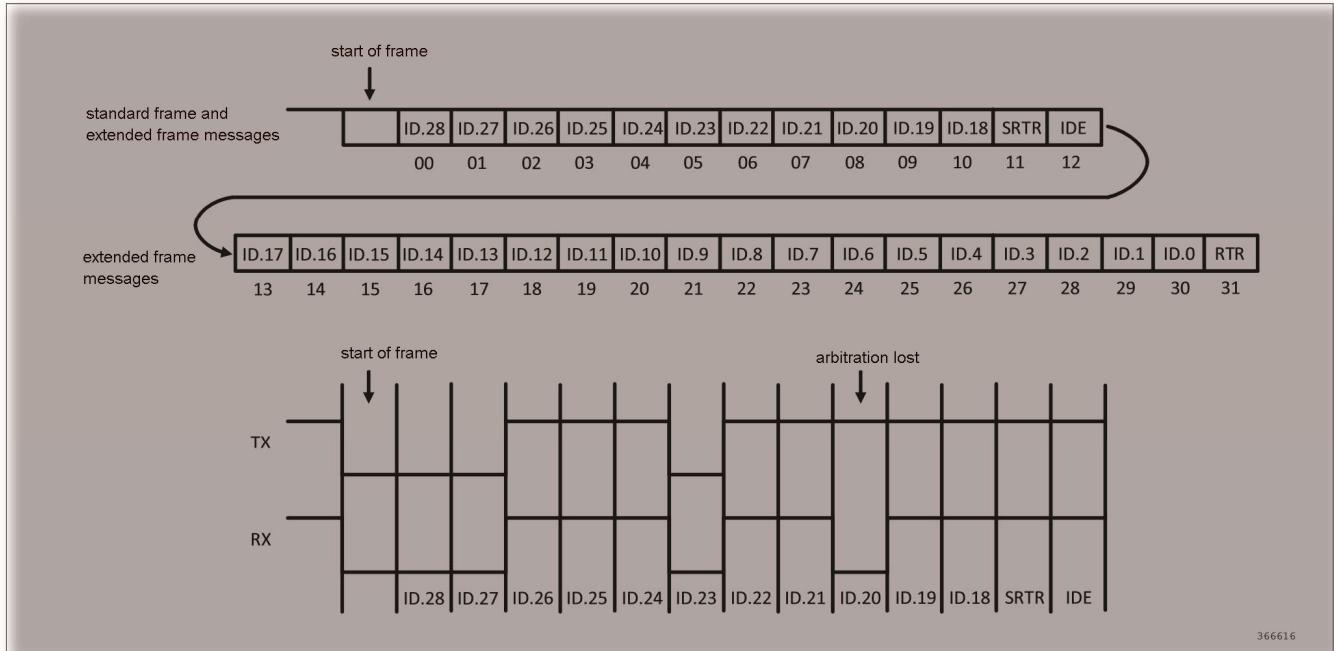


Figure 255. Example of arbitration lost bit number interpretation

### 25.5.10 CAN interrupt

There are 5 interrupts in the BasicCAN mode:

- Receive interrupt

It is generated when the receive FIFO is not empty and the receive interrupt enable bit (bit 1 in the CAN\_CR register) is set. The RI bit in CAN\_IR register is set to '1'.

- Transmit interrupt

It is generated whenever the transmit buffer status changes from 0 to 1 (released) and transmit interrupt enable bit (bit 2 in the CAN\_CR register) is set to 1.

- Error interrupt

This bit is set on a change of either the error status bit or bus status bit if the error interrupt enable bit (bit 3 in the CAN\_CR register) is set to logic 1.

- Data overrun interrupt

This bit is set on a '0-to-1' transition of the data overrun status bit, when the data overrun interrupt enable bit (bit 4 in the CAN\_CR register) is set to logic 1.

- Wakeup interrupt

It is generated when exiting from Sleep mode.

There are 8 different interrupts in the PeliCAN mode:

- Receive interrupt

It is generated when the receive FIFO is not empty and the RIE bit is set within the interrupt register.

- Transmit interrupt

It is generated whenever the transmit buffer status changes from '0-to-1' (released) and the TIE bit is set within the interrupt register.

- Error warning interrupt

It is generated on every change of either the error status bit or bus status bit and when the EIE bit is set within the interrupt register.

- Data overrun interrupt

It is generated on a '0-to-1' transition of the data overrun status bit and when the DOIE bit is set within the interrupt register.

- Error passive interrupt

It is generated whenever the CAN controller has reached the error passive status (at least one error counter exceeds the protocol-defined level of 127) or if the CAN controller is in the error passive status and enters the error active status again and the EPIE bit is set within the interrupt register.

- Arbitration lost interrupt

It is generated when the CAN controller loses the arbitration and becomes a receiver and the ALIE bit is set within the interrupt enable register.

- Bus error interrupt

It is generated when the CAN controller detects an error on the bus and the BEIE bit is set within the interrupt enable register.

- Wakeup interrupt

It is generated when the CAN controller is sleeping and bus activity is detected and the WUIE bit is set within the interrupt register.

## 25.6 CAN register description

Table 77. Overview of CAN registers

Offset	Acronym	Register Name	Reset	Section	Note
0x00	CAN_MOD	CAN mode register	0x00000001	Section 25.6.1	In PeliCAN mode only
0x00	CAN_CR	CAN control register	0x00000021	Section 25.6.2	In BasicCAN mode only
0x04	CAN_CMR	CAN command register	0x000000XX	Section 25.6.3	Reset value: BasicCAN mode: 0x00FF PeliCAN mode: 0x0000
0x08	CAN_SR	CAN status register	0x0000000C	Section 25.6.4	–
0x0C	CAN_IR	CAN interrupt register	0x000000XX	Section 25.6.5	Reset value: BasicCAN mode: 0x00E0 PeliCAN mode: 0x0000
0x10	CAN_IER	CAN interrupt enabled register	0x00000000	Section 25.6.6	In PeliCAN mode only
0x10	GROUP0_ACR	CAN acceptance code register group 0	0x000000XX	Section 25.6.7	BasicCAN mode
0x14	GROUP0_AMR	CAN acceptance mask register group 0	0x000000XX	Section 25.6.8	BasicCAN mode
0x18	CAN_BTR0	CAN bus timing 0	0x00000000	Section 25.6.9	–
0x1C	CAN_BTR1	CAN bus timing 1	0x00000000	Section 25.6.10	–

Offset	Acronym	Register Name	Reset	Section	Note
0x28	CAN_TXID0	CAN transmit identifier register 0	0x000000XX	Section 25.6.11	In BasicCAN mode only; the reset mode is 0xFF
0x2C	CAN_TXID1	CAN transmit identifier register 1	0x000000XX	Section 25.6.12	In BasicCAN mode only; the reset mode is 0xFF
0x2C	CAN_ALC	CAN arbitration lost capture register	0x00000000	Section 25.6.13	In PeliCAN mode only
0x30	CAN_ECC	CAN error code capture	0x00000000	Section 25.6.14	In PeliCAN mode only
0x34	CAN_EWLR	CAN error warning limit register	0x00000060	Section 25.6.15	In PeliCAN mode only
0x38	CAN_RXERR	CAN RX error counter register	0x00000000	Section 25.6.16	In PeliCAN mode only
0x3C	CAN_TXERR	CAN TX error count register	0x00000000	Section 25.6.17	In the PeliCAN mode only
0x40	CAN_SFF	CAN standard frame format register	0x000000XX	Section 25.6.18	In the PeliCAN mode only
0x44	CAN_TXID0	CAN transmit identifier register 0	0x000000XX	Section 25.6.19	In PeliCAN mode only
0x48	CAN_TXID1	CAN transmit identifier register 1	0x000000XX	Section 25.6.20	In PeliCAN mode only
0x4C	CAN_TXDATA0	CAN transmit data register 0	0x000000XX	Section 25.6.21	In the PeliCAN mode only
0x50	CAN_TXDATA1	CAN transmit data register 1	0x000000XX	Section 25.6.22	In the PeliCAN mode only
0x7C	CAN_CDR	CAN clock division register	0x00000000	Section 25.6.23	–
0x80	CAN_AFM0	CAN filter mode register 0	0x00000000	Section 25.6.24	–
0x84	CAN_AFM1	CAN filter mode register 1	0x00000000	Section 25.6.25	–
0x88	CAN_AFM2	CAN filter mode register 2	0x00000000	Section 25.6.26	–
0x8C	CAN_FGA0	CAN filter group enable register 0	0x00000001	Section 25.6.27	–
0x90	CAN_FGA1	CAN filter group enable register 1	0x00000000	Section 25.6.28	–
0x94	CAN_FGA2	CAN filter group enable register 2	0x00000000	Section 25.6.29	–
0x98+(x-1)*0x20	GROUPx_ACR	CAN acceptance code register group x (x = 1~19)	0x000000XX	Section 25.6.30	–
0xA8+(x-1)*0x20	GROUPx_AMR	CAN acceptance mask register group x (x = 1~19)	0x000000XX	Section 25.6.31	–

**25.6.1CAN mode register (CAN\_MOD)**

In PeliCAN mode only

Address offset: 0x00

Reset value: 0x0000 0001

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

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res.

AFM	STM	LO M	RM
-----	-----	---------	----

rw rw rw rw

Bit	Field	Type	Reset	Description
31:4	Reserved			Reserved, always read as 0.
3	AFM	rw	0x00	<p>Acceptance filter mode</p> <p>1: Single; select single acceptance filter (32 bits in length)</p> <p>0: Dual; select dual acceptance filters (16 bits activated for each)</p>
2	STM	rw	0x00	<p>Self test mode</p> <p>1: Self test; This mode detects all nodes and applies the self reception request to nodes with no activity. The CAN controller will send the request successfully even if no acknowledge is received.</p> <p>0: Normal mode; it is required to give an Acknowledge upon successful reception.</p>
1	LOM	rw	0x00	<p>Listen only mode</p> <p>1: Listen only; in this mode, the CAN controller does not send an Acknowledge to the bus even upon successful reception; the error counter retains the current value.</p> <p>0: normal mode</p>
0	RM	rw	0x01	<p>Reset mode</p> <p>1: Reset; when the Reset Mode bit is set and detected, the currently received or transmitted data is aborted and the system enters to the Reset Mode.</p> <p>0: Normal; when the Reset Mode bit receives a '1-0' transition, the CAN controller returns to the operating mode.</p>

**25.6.2CAN control register (CAN\_CR)**

In BasicCAN mode only:

Address offset: 0x00

Reset value: 0x0000 0021

The contents of the control register are used to change the behavior of the CAN controller.

Bits may be set or reset by the microcontroller which uses the control register as a read/write memory.

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

Res.
------

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res
-----

OIE	EIE	TIE	RIE	RR
-----	-----	-----	-----	----

rw	rw	rw	rw	rw
----	----	----	----	----

Bit	Field	Type	Reset	Description
31:5	Reserved			Reserved, and bit 5 is always read as 1.
4	OIE	rw	0x00	Overflow interrupt enable 1: enabled; if the data overflow bit is set, the microcontroller receives an overflow interrupt signal. 0: disabled; the microcontroller receives no overflow interrupt signal from the CAN controller.
3	EIE	rw	0x00	Error interrupt enable 1: enabled; if an error occurs or the bus status changes, the microcontroller receives an error interrupt signal. 0: disabled; the microcontroller receives no error interrupt signal from the CAN controller.
2	TIE	rw	0x00	Transmit interrupt enable 1: enabled; when a message has been successfully transmitted or the transmit buffer is accessible again (e.g. after an abort transmission command), the CAN controller transmits a transmit interrupt signal to the microcontroller. 0: disabled; the microcontroller receives no transmit interrupt signal from the CAN controller.
1	RIE	rw	0x00	Receive interrupt enable 1: enabled; when a message has been received without errors, the CAN controller transmits a receive interrupt signal to the microcontroller. 0: disabled; the microcontroller receives no receive interrupt signal from the CAN controller.
0	RR	rw	0x01	Reset request 1: present; CAN controller detection of a reset request results in aborting the current transmission/reception of a message and entering the reset mode. 0: absent; after the reset request bit receives a falling edge, the CAN controller returns to the operating mode.

### 25.6.3 CAN command register (CAN\_CMR)

Address offset: 0x04

Reset value: BasicCAN mode: 0x0000 00FF

PeliCAN mode: 0x0000 0000

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

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res.												ERB	SRR	CDO	RRB	AT	TR
------	--	--	--	--	--	--	--	--	--	--	--	-----	-----	-----	-----	----	----

w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Bit	Field	Type	Reset	Description
31: 6	Reserved			Reserved
5	ERB	w		<p>Empty rxbuffer</p> <p>1: Clear; clear all contents in the RXFIFO</p> <p>0: No action</p> <p>This command bit is used to clear the data overflow condition. It does not clear the overrun status bit. As long as the data is overrun and this bit is set, all contents in the receive buffer will be released.</p>
4	SRR	w		<p>PeliCAN mode:</p> <p>SRR: Self reset request</p> <p>1: the current message can be transmitted and received simultaneously</p> <p>0: (absent)</p>
3	CDO	w		<p>Clear data overrun</p> <p>1: Clear; data overrun status bit is cleared</p> <p>0: No action</p> <p>This command bit is used to clear the data overrun condition indicated by the data overrun status bit. As long as the data overrun status bit is set, no further data overrun interrupt is generated. It is allowed to give the clear data overrun command at the same time as a release receive buffer commands.</p>
2	RRB	w		<p>Release receive buffer</p> <p>1: Release; the receive buffer, representing the message memory space in the RXFIFO is released</p> <p>0: No action</p> <p>After reading the contents of the receive buffer, the microcontroller can release this memory space of the RXFIFO by setting the release receive buffer bit to 1. This may result in another message becoming immediately available within the receive buffer. This event will force another receive interrupt, if enabled. If there is no other message available, no further receive interrupt is generated and the receive buffer status bit is cleared.</p>

Bit	Field	Type	Reset	Description

1	AT	w	<p>Abort transmission</p> <p>1: present; if not already in progress, a pending transmission request is canceled</p> <p>0: absent; no action</p> <p>The abort transmission bit is used when the CPU requires the suspension of the previously requested transmission, e.g. to transmit a more urgent message before. A transmission already in progress is not stopped. In order to see if the original message had been either transmitted successfully or aborted, the transmission complete status bit should be checked. This should be done after the transmit buffer status bit has been set to 1 (released) or a transmit interrupt has been generated.</p>
0	TR	w	<p>Transmission request</p> <p>1: present; a message will be transmitted</p> <p>0: absent; no action</p> <p>If the transmission request was set to 1 in a previous command, it cannot be canceled by setting the transmission request bit to logic 0. The requested transmission may be canceled by setting the abort transmission bit to 0.</p>

#### 25.6.4 CAN status register (CAN\_SR)

Address offset: 0x08

Reset value: 0x0000 000C

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

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								BS	ES	TS	RS	TCS	TBS	DO S	RBS
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description

Bit	Field	Type	Reset	Description
7	BS	rw	0x00	<p>BS: Bus status</p> <p>1:bus-off; the CAN controller is not involved in bus activities</p> <p>0: bus-on; the CAN controller is involved in bus activities</p> <p>When the transmit error counter exceeds the limit of 255 (the bus status bit is set to '1' - bus-off), the CAN controller will set the reset request bit to '1' (present) and an error interrupt is generated, if enabled. It will stay in this mode until the CPU clears the reset request bit. Once this is completed the CAN controller will wait the minimum protocol-defined time (128 occurrences of the bus-free signal). After that the bus status bit is cleared (bus-on), the error status bit is set to '0' (ok), the error counters are reset and an error interrupt is generated, if enabled.</p>
6	ES	rw	0x00	<p>Error status</p> <p>1: error; at least one of the error counters has reached or exceeded the CPU warning limit</p> <p>0: ok; both error counters are below the warning limit</p> <p>Errors detected during reception or transmission will affect the error counters according to the CAN 2.0B protocol specification. The error status bit is set when at least one of the error counters has reached or exceeded the CPU warning limit of 96. An error interrupt is generated, if enabled.</p>
5	TS	rw	0x00	<p>Transmit status</p> <p>1: transmit; the CAN controller is transmitting a message</p> <p>0: idle; no transmit message is in progress</p> <p>If both the receive status and the transmit status bits are 0 (idle), the CAN-bus is idle.</p>
4	RS	rw	0x00	<p>Receive status</p> <p>1: receive; the CAN controller is receiving a message</p> <p>0: idle; no receive message is in progress</p> <p>If both the receive status and the transmit status bits are 0 (idle), the CAN-bus is idle.</p>
3	TCS	rw	0x01	<p>Transmission complete status</p> <p>1: complete; the last requested transmission has been successfully completed</p> <p>0: incomplete; the previously requested transmission is not yet completed</p> <p>The transmission complete status bit is set to '0' (incomplete) whenever the transmission request bit is set to '1'.</p> <p>The transmission complete status bit will remain at '0' until a message is transmitted successfully.</p>

2	TBS	rw	0x01	<p>Transmit buffer status</p> <p>1: released; the CPU may write a message into the transmit buffer</p> <p>0: locked; the CPU cannot access the transmit buffer; a message is waiting for transmission or is already in process</p> <p>If the CPU tries to write to the transmit buffer when the transmit buffer status bit is at 0 (locked), the written byte will not be accepted and will be lost without being indicated.</p>
1	DOS	rw	0x00	<p>Data overrun status</p> <p>1: overrun; a message was lost because there was not enough space for that message in the RXFIFO</p> <p>0: absent; no data overrun has occurred since the last clear data overrun command was given</p> <p>When a message that shall be received has passed the acceptance filter successfully (i.e. earliest after arbitration field), the CAN controller needs space in the RXFIFO to store the message descriptor. Accordingly there must be enough space for each data byte which has been received. If there is not enough space to store the message, that message will be dropped and the data overrun condition will be indicated to the CPU only, if this received message has no errors until the last but one bit of end of frame (message becomes valid).</p>
0	RBS	rw	0x00	<p>Receive buffer status</p> <p>1: full; one or more messages are available in the RXFIFO</p> <p>0: empty; no message is available</p> <p>After reading a message stored in the RXFIFO and releasing this memory space with the command release receive buffer, this bit is cleared. If there is another message available within the FIFO, this bit is set again with the next bit quantum (<math>t_{SCL}</math>).</p>

### 25.6.5 CAN interrupt register (CAN\_IR)

Address offset: 0x0C

Reset value: BasicCAN mode: 0x0000 00E0

PeliCAN mode: 0x0000 0000

The interrupt register allows the identification of an interrupt source. When one or more bits of this register are set, the interrupt is activated. The interrupt register appears to the microcontroller as a read only memory.

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BEI	ALI	EPI	Res.	DOI	EI	TI	RI

r      r      r      r      r      r      r      r

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7	BEI	r		Basic mode: Reserved and read as 1. PeliCAN mode: BEI: Bus error interrupt 1: set; this bit is set when the CAN controller detects an error on the bus and the BEIE bit is set within the interrupt enable register. 0: reset
6	ALI	r		Basic mode: Reserved and read as 1. PeliCAN mode: ALI: Arbitration lost interrupt 1: set; this bit is set when the CAN controller loses the arbitration and becomes a receiver and the ALIE bit is set within the interrupt enable register. 0: reset
5	EPI	r		Basic mode: Reserved and read as 1. PeliCAN mode: EPI: Error passive interrupt 1: set; this bit is set to '1' whenever the CAN controller has reached the error passive status (at least one error counter exceeds the protocol-defined level of 127) or if the CAN controller is in the error passive status and enters the error active status again and the EPIE bit is set within the interrupt register. 0: reset
4	Reserved			Reserved and always read as 0.
3	DOI	r		Data overrun interrupt 1: set; this bit is set on a '0-to-1' transition of the data overrun status bit, when the data overrun interrupt enable is set to 1. 0: reset; this bit is cleared by any read access of the microcontroller The overrun interrupt bit (if enabled) and the data overrun status bit are set at the same time.

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

2	EI	r	Error interrupt 1: set; this bit is set on a change of either the error status bit or bus status bit if the error interrupt is enabled 0: reset; this bit is cleared by any read access of the microcontroller
1	TI	r	Transmit interrupt 1: set; this bit is set whenever the transmit buffer status changes from 0 to 1 (released) and transmit interrupt is enabled 0: reset; this bit is cleared by any read access of the microcontroller
0	RI	r	Receive interrupt 1: set; this bit is set while the receive FIFO is not empty and the receive interrupt is enabled 0: reset; this bit is cleared by any read access of the microcontroller  The receive interrupt bit (if enabled) and the receive buffer status bit are set at the same time.  It should be noted that the receive interrupt bit is cleared upon a read access, even if there is another message available within the FIFO. The moment the release receive buffer command is given and there is another message valid within the receive buffer, the receive interrupt is set again (if enabled) with the next tscl.

### 25.6.6 CAN interrupt enable register (CAN\_IER)

In PeliCAN mode only

Address offset: 0x10

Reset value: 0x0000 0000

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

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res.	BEIE	ALIE	EPIE	Res.	DOI E	EIE	TIE	RIE
------	------	------	------	------	-------	-----	-----	-----

rw								
----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7	BEIE	rw	0x00	Bus error interrupt enable 1: enabled; if an bus error has been detected, the CAN controller requests the respective interrupt 0: disabled
Bit	Field	Type	Reset	Description

6	ALIE	rw	0x00	Arbitration lost interrupt enable 1: enabled; if the CAN controller has lost arbitration, the respective interrupt is requested 0: disabled
5	EPIE	rw	0x00	Error passive interrupt enable 1: enabled; if the error status of the CAN controller changes from error passive to error active or vice versa, the respective interrupt is requested 0: disabled
4	Reserved			Reserved and always read as 0.
3	DOIE	rw	0x00	Data overrun interrupt enable 1: enabled; if the data overrun status bit is set (see status register), the CAN controller requests the respective interrupt 0: disabled
2	EIE	rw	0x00	Error warning interrupt Enable 1: enabled; if the error or bus status changes (see status register), the CAN controller requests the respective interrupt 0: disabled
1	TIE	rw	0x00	Transmit interrupt enable 1: enabled; when a message has been successfully transmitted or the transmit buffer is accessible again (e.g. after an abort transmission command), the CAN controller requests the respective interrupt. 0: disabled
0	RIE	rw	0x00	Receive interrupt enable 1: enabled; when the receive buffer status is 'full', the CAN controller requests the respective interrupt 0: disabled

### 25.6.7 CAN acceptance code register group 0 (GROUP0\_ACR)

BasicCAN mode: GROUP0\_ACR

Address offset: 0x10

Reset value: 0x0000 00XX

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

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res.

AC

rw														
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.

7:0	AC	rw	0xXX	(Acceptance code) This register can be accessed (read/write), if the reset request bit is set HIGH (present). When a message is received which passes the acceptance test and there is receive buffer space left, then the respective descriptor and data field are sequentially stored in the RXFIFO. When the complete message has been correctly received, the following occurs: The receive status bit is set HIGH (full). If the receive interrupt enable bit is set HIGH (enabled), the receive interrupt is set HIGH (interrupt generated). The acceptance code bits (AC.7 to AC.0) and the eight most significant bits of the message's identifier (ID.10 to ID.3) must be equal to those bit positions which are marked relevant by the acceptance mask bits (AM.7 to AM.0). If the conditions as described in the following equation are fulfilled, acceptance is given: $[(ID.10-ID.3) \equiv (AC.7-AC.0)] \vee (AM.7-AM.0) \equiv 11111111$
-----	----	----	------	---

PeliCAN mode: four acceptance code registers in the group 0 are respectively: GROUP0\_ACR0, GROUP0\_ACR1, GROUP0\_ACR2, GROUP0\_ACR3

GROUP0\_ACR0: address offset: 0x40 reset value: 0x0000 00XX

GROUP0\_ACR1: address offset: 0x44 reset value: 0x0000 00XX

GROUP0\_ACR2: address offset: 0x48 reset value: 0x0000 00XX

GROUP0\_ACR3: address offset: 0x4C reset value: 0x0000 00XX

Note: For details, see the introduction to PeliCAN mode in the identifier filtering. This is Acceptance Code Register group 0. The other groups are described in Section 25.6.30.

### 25.6.8 CAN acceptance mask register group 0 (GROUP0\_AMR)

BasicCAN mode: GROUP0\_AMR

Address offset: 0x14

Reset value: 0x0000 00XX

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

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res.

AM

rw												
----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7:0	AM	rw	0xXX	(Acceptance mask) This register can be accessed (read/write), if the reset request bit is set HIGH (present). The acceptance mask register qualifies which of the corresponding bits of the acceptance code register are 'relevant' or 'don't care' (can be any value) for acceptance filtering.

PeliCAN mode: four acceptance mask registers in the group 0 are respectively:

GROUP0\_AMR0, GROUP0\_AMR1, GROUP0\_AMR2, GROUP0\_AMR3

GROUP0\_AMR0: address offset: 0x50 reset value: 0x0000 00XX

GROUP0\_AMR1: address offset: 0x54 reset value: 0x0000 00XX

GROUP0\_AMR2: address offset: 0x58 reset value: 0x0000 00XX

GROUP0\_AMR3: address offset: 0x5C reset value: 0x0000 00XX

Note: For details, see the introduction to PeliCAN mode in the identifier filtering. This is Acceptance Mask Register group 0. The other groups are described in Section 25.6.31.

### 25.6.9 CAN bus timing register 0 (CAN\_BTR0)

Address offset: 0x18

Reset value: 0x0000 0000

The contents of the bus timing register 0 defines the values of the Baud Rate Prescaler (BRP) and the Synchronization Jump Width (SJW). This register can be accessed (read/write) if the reset mode is active.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.								SJW	BRP							
Bit	Field	Type	Reset	Description												
31:8	Reserved			Reserved, always read as 0.												
7:6	SJW	rw	0x00	Synchronization jump width To compensate for phase shifts between clock oscillators of different bus controllers, all bus controllers must re-synchronize on any relevant signal edge of the current transmission. The synchronization jump width defines the maximum number of clock cycles a bit period may be shortened or lengthened by one re-synchronization. $t_{SJW} = t_{SCL} \times (SJW + 1)$												
5:0	BRP	rw	0x00	Baud rate prescaler The period of CAN system clock $t_{SCL}$ is programmable and determines the corresponding bit timing. The CAN system clock is calculated in the following formula: $t_{SCL} = 2 \times t_{CLK} \times (BRP + 1)$ where $t_{CLK}$ = APB1 clock period												

### 25.6.10 CAN bus timing 1 (CAN\_BTR1)

Address offset: 0x1C

Reset value: 0x0000 0000

The contents of bus timing register 1 defines the length of the bit period, the location of the sample point and the number of samples to be taken at each sample point.

This register can be accessed (read/write) if the reset mode is active.

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

Res.
------

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res	SAM	TESG2	TESG1
-----	-----	-------	-------

.	rw														
---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7	SAM	rw	0x00	<p>Sampling</p> <p>1: triple; the bus is sampled three times; recommended for low/medium speed buses (class A and B) where filtering spikes on the bus line is beneficial</p> <p>0: single; the bus is sampled once; recommended for high speed buses (SAE class C)</p>
6:0	TSEG2 TSEG1	rw	0x00	<p>Time segment 1 and Time segment 2</p> <p>TSEG1 and TSEG2 determine the number of clock cycles per bit period and the location of the sample point, where:</p> $t_{\text{SYNCSEG}} = 1 \times t_{\text{SCL}}$ $t_{\text{TSEG1}} = t_{\text{SCL}} \times (8 \times \text{TSEG1.3} + 4 \times \text{TSEG1.2} + 2 \times \text{TSEG1.1} + \text{TSEG1.0} + 1)$ $t_{\text{TSEG2}} = t_{\text{SCL}} \times (4 \times \text{TSEG2.2} + 2 \times \text{TSEG2.1} + \text{TSEG2.0} + 1)$

### 25.6.11 CAN transmit identifier register 0 (CAN\_TXID0)

BasicCAN mode:

Address offset: 0x28

Reset value: 0x0000 00XX

Note: 0xFF for the Reset Mode.

The transmit identifier register 0 defines the frame format and the data length. This register can be accessed (read/write) if the operating mode is active.

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

Res.
------

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res.	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
------	------	-----	-----	-----	-----	-----	-----	-----

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved and always read as 0.
7:0	IDx	rw	0xXX	CAN identifier byte 10 ~ 3 Note: In the Peli mode, bits [3:0] are DLC[3:0].

### 25.6.12 CAN transmit identifier register 1 (CAN\_TXID1)

In BasicCAN mode only:

Address offset: 0x2C

Reset value: 0x0000 00XX

Note: 0xFF for the Reset Mode.

The transmit identifier register 1 defines the frame format and the data length. This register can be accessed (read/write) if the operating mode is active.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0
rw rw rw rw rw rw rw rw								rw rw rw rw rw rw rw rw							

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7:5	IDx	rw	0x0X	CAN identifier byte 2 ~ 0
4	RTR	rw	0x0X	Frame format (Remote transmission request) 1: Remote; CAN transmits a remote frame 0: Data; CAN transmits a data frame

Bit	Field	Type	Reset	Description
3:0	DLCx	rw	0x0X	Transmitted data field length (Data length code 0~8)

In BasicCAN mode only:

Address offset: 0x30 ~ 0x4C

Reset value: 0x0000 0000

Tx data register CAN\_TXDR0 ~ 7. This register can be accessed (read/write) if the operating mode is active. The receive buffer and transmit buffer have the same data format.

### 25.6.13 CAN arbitration lost capture register (CAN\_ALC)

In PeliCAN mode only

Address offset: 0x2C

Reset value: 0x0000 0000

Controller Area Network (CAN)											UM_MM32F013x_Ver1.00						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.																	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Res.											BITNO						
												RW	RW	RW	RW	RW	

Bit	Field	Type	Reset	Description
31:5	Reserved			Reserved, always read as 0.
4:0	BITNO	rw	0x00	<p>Refer to the table below for values and functions (Bit number)</p> <p>On arbitration lost, the corresponding arbitration lost interrupt is forced, if enabled. At the same time, the current Bit position of the bit stream processor is captured into the Arbitration Lost Capture register. The content within this register is fixed until the users software has read out its contents once. The capture mechanism is then activated again.</p> <p>The corresponding interrupt flag located in the interrupt register is cleared during the read access to the interrupt register.</p> <p>A new arbitration lost interrupt is not possible until the arbitration lost capture register is read out once.</p>

Table 78. Function of bits 4 to 0 of the arbitration lost capture register

Bit					Decimal Value	Function
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	0	0	0	0	arbitration lost in bit 1 of identifier
0	0	0	0	1	1	arbitration lost in bit 2 of identifier
0	0	0	1	0	2	arbitration lost in bit 3 of identifier
0	0	0	1	1	3	arbitration lost in bit 4 of identifier
0	0	1	0	0	4	arbitration lost in bit 5 of identifier
0	0	1	0	1	5	arbitration lost in bit 6 of identifier
0	0	1	1	0	6	arbitration lost in bit 7 of identifier
0	0	1	1	1	7	arbitration lost in bit 8 of identifier
0	1	0	0	0	8	arbitration lost in bit 9 of identifier

Bit					Decimal Value	Function
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	1	0	0	1	9	arbitration lost in bit 10 of identifier
0	1	0	1	0	10	arbitration lost in bit 11 of identifier
0	1	0	1	1	11	arbitration lost in bit SRTR; note 2
0	1	1	0	0	12	arbitration lost in bit IDE
0	1	1	0	1	13	arbitration lost in bit 12 of identifier; note 3
0	1	1	1	0	14	arbitration lost in bit 13 of identifier; note 3
0	1	1	1	1	15	arbitration lost in bit 14 of identifier; note 3
1	0	0	0	0	16	arbitration lost in bit 15 of identifier; note 3
1	0	0	0	1	17	arbitration lost in bit 16 of identifier; note 3
1	0	0	1	0	18	arbitration lost in bit 17 of identifier; note 3
1	0	0	1	1	19	arbitration lost in bit 18 of identifier; note 3
1	0	1	0	0	20	arbitration lost in bit 19 of identifier; note 3
1	0	1	0	1	21	arbitration lost in bit 20 of identifier; note 3
1	0	1	1	0	22	arbitration lost in bit 21 of identifier; note 3
1	0	1	1	1	23	arbitration lost in bit 22 of identifier; note 3
1	1	0	0	0	24	arbitration lost in bit 23 of identifier; note 3
1	1	0	0	1	25	arbitration lost in bit 24 of identifier; note 3
1	1	0	1	0	26	arbitration lost in bit 25 of identifier; note 3
1	1	0	1	1	27	arbitration lost in bit 26 of identifier; note 3
1	1	1	0	0	28	arbitration lost in bit 27 of identifier; note 3
1	1	1	0	1	29	arbitration lost in bit 28 of identifier; note 3
1	1	1	1	0	30	arbitration lost in bit 29 of identifier; note 3
1	1	1	1	1	31	arbitration lost in bit RTR; note 3

Note: Binary coded frame bit number where arbitration was lost. Bit RTR for standard frame messages. Extended frame messages only.

### 25.6.14 CAN error code capture register (CAN\_ECC)

In PeliCAN mode only

Address offset: 0x30

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ERRC	DIR	SEG					
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7:6	ERRC	r	0x00	Error code 00: bit error 01: form error 10: stuff error 11: other type of error
5	DIR	r	0x00	Direction 1: RX; error occurred during reception 0: TX; error occurred during transmission

4:0	SEG	r	0x00	Segment
				00010: ID.28-ID.21
				00011: start of frame
				00100: bit SRTR
				00101: bit IDE
				00110: ID.20-ID.18
				00111: ID.17-ID.13
				01000: CRC sequence
				01001: reserved bit 0
				01010: data field
				01011: data length code
				01100: bit RTR
				01101: reserved bit 1
				01110: ID.4-ID.0
				01111: ID.12-ID.5
				10001: active error flag
				10010: intermission
				10011: tolerate dominant bits
				10110: passive error flag
				10111: error delimiter
				11000: CRC delimiter
				11001: acknowledge slot
				11010: end of frame
				11011: acknowledge delimiter
				11100: overload flag
				Others: reserved

Note: Bit settings reflect the current frame segment to distinguish between different error events.

If a bus error occurs, the corresponding bus error interrupt is always forced, if enabled. At the same time, the current position of the bit stream processor is captured into the error code capture register. The content within this register is fixed until the users software has read out its content. The capture mechanism is then activated again. The corresponding interrupt flag located in the interrupt register is cleared during the access to the interrupt register. A new bus interrupt is not possible until the capture register is read out once.

### 25.6.15 CAN error warning limit register (CAN\_EWLR)

In PeliCAN mode only:

Address offset: 0x34

Reset value: 0x0000 0060

The error warning limit can be defined within this register. The default value (reset value) is 0060. In reset mode, this register appears to the CPU as a read/write memory. In operating mode it is read only.

Note that a content change of the EWLR is only possible, if the reset mode was entered previously. An error status change (see status register) and an error warning interrupt forced by the new register content will not occur until the reset mode is canceled again.

### Controller Area Network (CAN)

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

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

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7:0	EWL	rw	0x60	Programmable error warning limit The error status bit is set when at least one of the error counters has exceeded the programmed error limit.

### 25.6.16 CAN RX error counter register (CAN\_RXERR)

In PeliCAN mode only:

Address offset: 0x38

Reset value: 0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.

7:0	RXERR	rw	0x00	<b>RX error counter register</b> The RX error counter register reflects the current value of the receive error counter. After a hardware reset, this register is initialized to 0. In operating mode, this register appears to the CPU as a read only memory. A write access to this register is possible only in reset mode. If a bus-off event occurs, the RX error counter is initialized to 0. During the bus-off period, writing to this register has no effect. <small>Note: a CPU-forced content change of the RX error counter is only possible, if the reset mode was entered previously. An error status change (see status register), an error warning or an error interrupt forced by the new register content will not occur, until the reset mode is canceled again.</small>
-----	-------	----	------	--

### 25.6.17 CAN TX error counter register (CAN\_TXERR)

In PeliCAN mode only

Address offset: 0x3C

Reset value: 0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description

7:0	TXERR	rw	0x00	<p>TX error counter register</p> <p>The TX error counter register reflects the current value of the transmit error counter.</p> <p>In operating mode, this register appears to the CPU as a read only memory. Write access to this register is possible only in reset mode. After a hardware reset, this register is initialized to logic 0. If a bus-off event occurs, the TX error counter is initialized to 127 to count the minimum protocol-defined time (128 occurrences of the bus-free signal). Reading the TX error counter during this time gives information about the status of the bus-off recovery.</p> <p>If bus-off is active, a write access to TXERR in the range from 0 to 254 clears the bus-off flag and the controller will wait for one occurrence of 11 consecutive recessive bits (bus-free) after the reset mode has been cleared.</p> <p>Writing 255 to TXERR allows to initiate a CPU-driven bus-off event.</p> <p>A CPU-forced content change of the TX error counter is only possible, if the reset mode was entered previously. An error or bus status change (see status register), an error warning or an error interrupt forced by the new register content will not occur until the reset mode is canceled again. After leaving the reset mode, the new TX counter content is interpreted and the bus-off event is performed in the same way, as if it was forced by a bus error event. That means the reset mode is entered again, the TX error counter is initialized to 127, the RX counter is cleared and all concerned status and interrupt register bits are set.</p> <p>Clearing of reset mode now will perform the protocol-defined bus-off recovery sequence (waiting for 128 occurrences of the bus-free signal).</p> <p>If the reset mode is entered again before the end of bus-off recovery (<math>\text{TXERR} &gt; 0</math>), bus-off keeps active and TXERR is frozen.</p>
-----	-------	----	------	---

### 25.6.18 CAN standard frame format register (CAN\_SFF)

In PeliCAN mode only

Address offset: 0x40

Reset value: 0x0000 00XX

The standard frame format register defines the frame format and the data length. This register can be accessed (read/write) if the operating mode is active. It is used as a transmit register for write access or a receive register for read access, with data structure being the same.

Res.

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

Bit	Field	Type	Reset	Description
7	FF	rw	0x0X	Frame format 1: EFF; extended frame format will be transmitted/received by CAN 0: SFF; standard frame format will be transmitted/received by CAN
6	RTR	rw	0x0X	Frame format (Remote transmission request) 1: Remote; remote frame will be transmitted/received by CAN 0: Data; data frame will be transmitted/received by CAN
5:4	Reserved			Reserved.
3:0	DLC	rw	0x0X	Data length code bit The transmitted data field length is 0 to 8.
31:8	Reserved			Reserved, always read as 0.

### 25.6.19 CAN transmit identifier register 0 (CAN\_TXID0)

In PeliCAN mode only:

Address offset: 0x44

Reset value: 0x0000 00XX

The transmit identifier register 0 defines the frame Identifier. This register can be accessed (read/write) if the operating mode is active. It is used as a transmit register for write access or a receive register for read access, with data structure being the same.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
								rw							

Bit	Field	Type	Reset	Description											
31:8	Reserved			Reserved and always read as 0.											
7:0	IDx	rw	0xXX	CAN identifier ID28 ~ ID21 (Identifier bit 28-21) In the standard frame format, ID28 to ID21 plus ID20 to ID18 form 11-bit identifier.											
<b>25.6.20 CAN transmit identifier register 1 (CAN_TXID1)</b>															
In PeliCAN mode only:															
Address offset: 0x48															
Reset value: 0x0000 00XX															
The transmit identifier register 1 defines the frame Identifier. This register can be accessed (read/write) if the operating mode is active. It is used as a transmit register for write access or a receive register for read access, with data structure being the same.															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
rw								rw							
Bit	Field	Type	Reset	Description											
7:5	IDx	rw	0x0X	CAN identifier ID20 ~ ID18 (Identifier bit 20-18)											
4:0	IDx	rw	0x0X	Standard frame: meaningless Extended frame: CAN identifier ID17 ~ ID13 (Identifier bit 17-13)											
31:8	Reserved			Reserved, always read as 0.											
<b>25.6.21 CAN transmit data register 0 (CAN_TXDATA0)</b>															
In PeliCAN mode only															
Address offset: 0x4C															
Reset value: 0x0000 00XX															
The transmit data register 0 is used for transmission and storage of CAN data. This register can be accessed (read/write) if the operating mode is active. It is used as a transmit register for write access or a receive register for read access, with data structure being the same.															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								DATA0							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
7:0	DATA0	rw	0xXX	CAN transmit data 0 extended frame ID12~5
31:8	Reserved			Reserved, always read as 0.

### 25.6.22 CAN transmit data register 1 (CAN\_TXDATA1)

In PeliCAN mode only:

Address offset: 0x50

Reset value: 0x0000 00XX

The transmit data register 1 is used for transmission and storage of CAN data. This register can be accessed (read/write) if the operating mode is active. It is used as a transmit register for write access or a receive register for read access, with data structure being the same.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								DATA1							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
7:0	DATA1	rw	0xXX	CAN transmit data 1 extended frame ID4~0, 3 least significant bits are meaningless
31:8	Reserved			Reserved, always read as 0.

In PeliCAN mode only:

Address offset: 0x54 ~ 0x70

Reset value: 0x0000 00XX

These offset addresses all belong to CAN data registers. In the extended frame format, the CAN transmit data 0 is stored in the remaining DATA2 data and so forth. These registers are used as transmit registers for write access or receive registers for read access, with data structure being the same.

### 25.6.23 CAN clock division register (CAN\_CDR)

Address offset: 0x7C

Reset value: 0x0000 0000

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

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res.

MODE

Res.

rw

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7	MODE	rw	0x00	CAN mode If MODE is 0, the CAN controller operates in the BasicCAN mode. Else, it operates in the PeliCAN mode. A write access to this register is possible only in reset mode.
6:0	Reserved			Reserved and always read as 0.

### 25.6.24 CAN filter mode register 0 (CAN\_AFM0)

Address offset: 0x80

Reset value: 0x0000 0000

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

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res

.	AFM7	AFM6	AFM5	AFM4	AFM3	AFM2	AFM1	Res.
---	------	------	------	------	------	------	------	------

rw

rw

rw

rw

rw

rw

rw

rw

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7: 1	AFMx	rw	0x00	Acceptance filter mode, x= 1~ 7 1: Single; select single acceptance filter (32 bits in length) 0: Dual; select dual acceptance filters (16 bits activated for each)
0	Reserved			Reserved and always read as 0.

### 25.6.25 CAN filter mode register 1 (CAN\_AFM1)

Address offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								AFM15	AFM14	AFM13	AFM12	AFM11	AFM10	AFM9	AFM8
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7: 0	AFMx	rw	0x00	Acceptance filter mode, x= 8~ 15 1: Single; select single acceptance filter (32 bits in length) 0: Dual; select dual acceptance filters (16 bits activated for each)

### 25.6.26 CAN filter mode register 2 (CAN\_AFM2)

Address offset: 0x88

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								AFM19	AFM18	AFM17	AFM16	rw	rw	rw	rw
.								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
-----	-------	------	-------	-------------

31:4	Reserved			Reserved, always read as 0.
3: 0	AFMx	rw	0x00	Acceptance filter mode, x = 16~19  1: Single; select single acceptance filter (32 bits in length) 0: Dual; select dual acceptance filters (16 bits activated for each)

### 25.6.27 CAN filter group enable register 0 (CAN\_FGA0)

Address offset: 0x8C

Reset value: 0x0000 0001

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

Res.

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

Res	FGA7	FGA6	FGA5	FGA4	FGA3	FGA2	FGA1	Res.
.	rw							

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7: 1	FGAx	rw	0x00	Group filtering enabled (x = 1~7)  0: No effect 1: Effective Group filtering
0	Reserved			Reserved and always read as 1.

### 25.6.28 CAN filter group enable register 1 (CAN\_FGA1)

Address offset: 0x90

Reset value: 0x0000 0000

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

Res.

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

Res.	FGA15	FGA14	FGA13	FGA12	FGA11	FGA10	FGA9	FGA8
.	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.

7: 0 FGAx rw 0x00 Group filtering enabled (x = 8~15)  
0: No effect  
1: Effective Group filtering

### **25.6.29 CAN filter group enable register 2 (CAN\_FGA2)**

Address offset: 0x94

Reset value: 0x0000 0000

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
												Res	.	FGA19	FGA18	FGA17	FGA16

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7: 0	FGAx	rw	0x00	Group filtering enabled (x = 16~19) 0: No effect 1: Effective Group filtering

### **25.6.30 CAN acceptance code register group x (x = 1~19)(GROUPx\_ACR)**

## BasicCAN mode: GROUPx ACR

Address offset: 0x098+(x-1)\*0x20

Reset value: 0x0000 00XX

Res.

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
31:8	Reserved			Reserved, always read as 0.

7:0	AC	rw	0xXX	(Acceptance code) This register can be accessed (read/write), if the reset request bit is set HIGH (present). When a message is received which passes the acceptance test and there is receive buffer space left, then the respective descriptor and data field are sequentially stored in the RXFIFO. When the complete message has been correctly received, the following occurs: The receive status bit is set HIGH (full). If the receive interrupt enable bit is set HIGH (enabled), the receive interrupt is set HIGH (interrupt generated). The acceptance code bits (AC.7 to AC.0) and the eight most significant bits of the message's identifier (ID.10 to ID.3) must be equal to those bit positions which are marked relevant by the acceptance mask bits (AM.7 to AM.0). If the conditions as described in the following equation are fulfilled, acceptance is given: $[(ID.10-ID.3) \equiv (AC.7-AC.0)] \vee (AM.7-AM.0) \equiv 11111111$
-----	----	----	------	---

PeliCAN mode: four acceptance code registers in each group are respectively: GROUPx\_ACR0, GROUPx\_ACR1, GROUPx\_ACR2, GROUPx\_ACR3

GROUPx\_ACR0: address offset: 0x098 + (x - 1) \* 0x20 reset value: 0x0000 00XX

GROUPx\_ACR1: address offset: 0x09C + (x - 1) \* 0x20 reset value: 0x0000 00XX

GROUPx\_ACR2: address offset: 0x0A0 + (x - 1) \* 0x20 reset value: 0x0000 00XX

GROUPx\_ACR3: address offset: 0x0A4 + (x - 1) \* 0x20 reset value: 0x0000 00XX

Note: The acceptance code register group 0 is described in Section 25.6.7.

### 25.6.31 CAN acceptance mask register group x (x = 1~19)(GROUPx\_AMR)

BasicCAN mode: GROUPx\_AMR

Address offset: 0xA8+(x-1)\*0x20

Reset value: 0x0000 00XX

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

Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Res.

AM

rw								
----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7:0	AM	rw	0xXX	(Acceptance mask) This register can be accessed (read/write), if the reset request bit is set HIGH (present). The acceptance mask register qualifies which of the corresponding bits of the acceptance code register are 'relevant' or 'don't care' (can be any value) for acceptance filtering.

PeliCAN mode: four acceptance mask registers in each group are respectively:  
GROUPx\_AMR0, GROUPx\_AMR1, GROUPx\_AMR2, GROUPx\_AMR3

GROUPx\_AMR0: address offset:  $0x0A8 + (x - 1) * 0x20$

reset value: 0x0000 00XX

GROUPx\_AMR1: address offset:  $0x0AC + (x - 1) * 0x20$

reset value: 0x0000 00XX

GROUPx\_AMR2: address offset:  $0x0B0 + (x - 1) * 0x20$

reset value: 0x0000 00XX

GROUPx\_AMR3: address offset:  $0x0B4 + (x - 1) * 0x20$

reset value: 0x0000 00XX

Note: The acceptance mask register group 0 is described in Section 25.6.8.

### 26.1 USB introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus.

USB suspend/resume operations are supported, which allows to stop the device clocks for low-power consumption.

### 26.2 USB main features

- USB specification version 2.0 full-speed compliant
- Supports full-speed 12M mode
- One control transfer endpoint and four independent general endpoints, used for interrupt transfer and bulk transfer
- At most 64 bytes in a package can be transferred in the control, bulk or interrupt transfer.
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Supports USB suspend/resume operations
- Supports DMA transfer

The block diagram of the USB peripheral is shown as below:

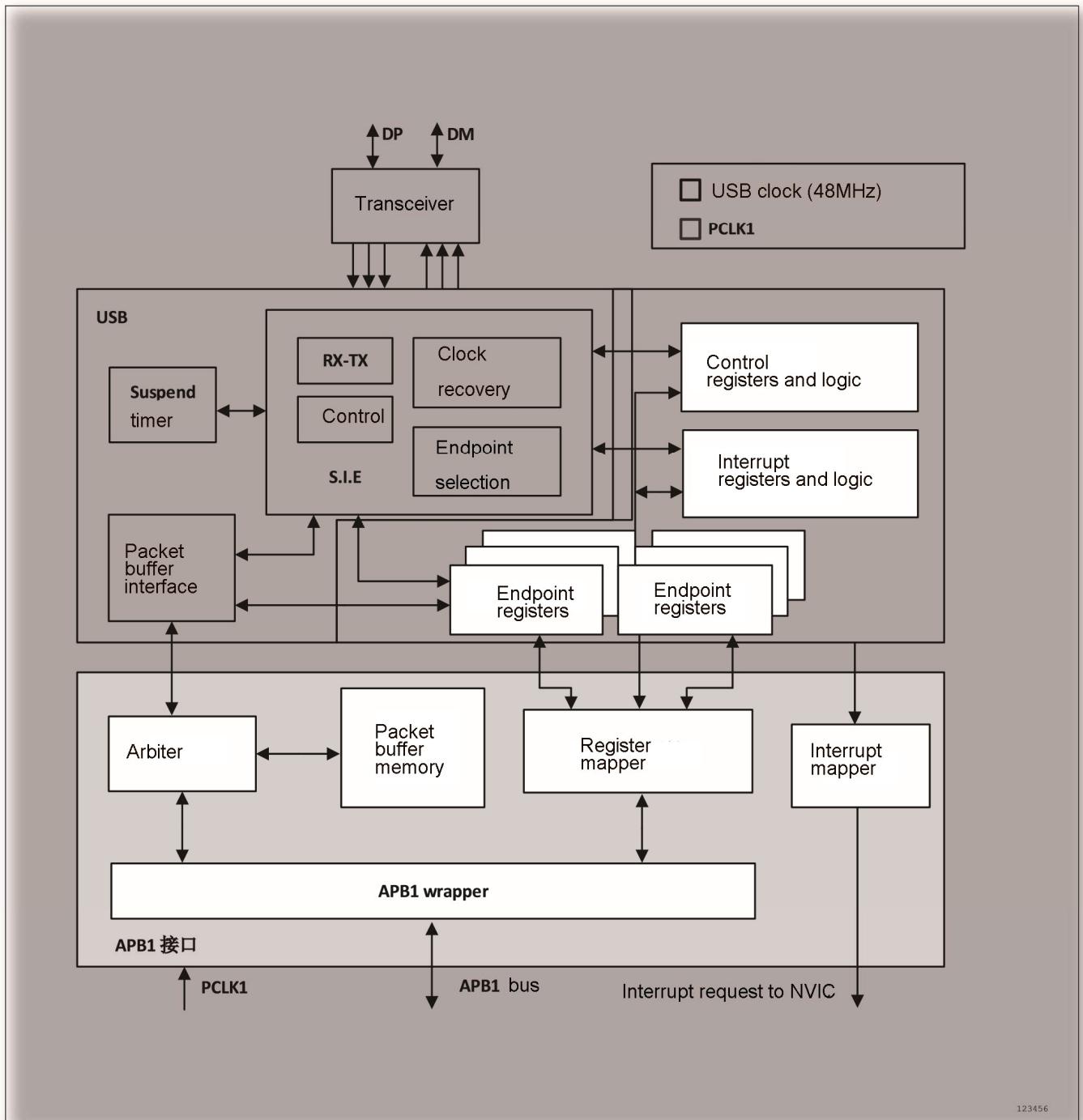


Figure 256. USB peripheral block diagram

## 26.3 USB functional description

The USB peripheral provides an USB-compliant connection between the host PC and the function implemented by the microcontroller. Data transfer between the host PC and the microcontroller occurs through a buffer memory. The USB peripheral interfaces with the PC host, detecting token packets, handling data transmission/reception, and processing

handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint is associated with a buffer description block of 64 bytes. This buffer is inside the USB peripheral and is not accessed directly by the CPU. When a token packet for a valid function/endpoint is recognized by the USB peripheral, the related data transfer (if required and if the endpoint is configured) takes place. The data buffered by the USB peripheral is loaded in an internal register and memory access to the dedicated buffer is performed. When all the data has been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which transaction type is requested by the host,
- which endpoint has to be served,
- which type of transaction took place,
- the response of the endpoint
- whether the transaction is completed

The USB peripheral can be placed in low-power mode (SUSPEND mode), by writing in the USB\_POWER register, whenever the unit is not required. At this time, all dynamic power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of data transaction on the USB line, while in low-power mode, wakes the USB peripheral up. The USB system can also be awaken through software configuration. A special interrupt input source can be connected directly to a wakeup line to allow the system to immediately restore the normal clock system and support direct clock start/stop.

### 26.3.1 Description of USB functional modules

The USB peripheral contains 8-bit FIFO of 320 bytes, with each endpoint of 64-byte FIFO. On the APB1 bus, each register or stored data uses the 8 lower bits of the 32-bit bus.

The USB peripheral includes the following registers:

- USB register, which is used to configure the USB parameters and check status
- Endpoint status register
- Endpoint setting register
- Setup data register, which is used to store the data of the SETUP packet
- Endpoint data control register, which is used to control the data stream

When the APB1 bus or DMA writes/reads data to/from the data endpoint, the number of FIFO data will increase or decrease. The FIFO pointer will change only when the endpoint receives or transmits data successfully. Each endpoint has one FIFO data register as the entry address for write/read access to FIFO. Each endpoint also has a dedicated register to query the number of effective data inside the FIFO at present.

Only endpoint 1 and 2 support the DMA transfer.

## 26.4 Programming considerations

---

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

### 26.4.1 General description of USB transfer

The relationship between packet, transaction and transfer is shown below.

Packet is a fundamental unit of information transmission in the USB system. All data are packed and transferred on the bus.

A fundamental USB packet is illustrated as below, such as a token packet, a data packet and a handshake packet.

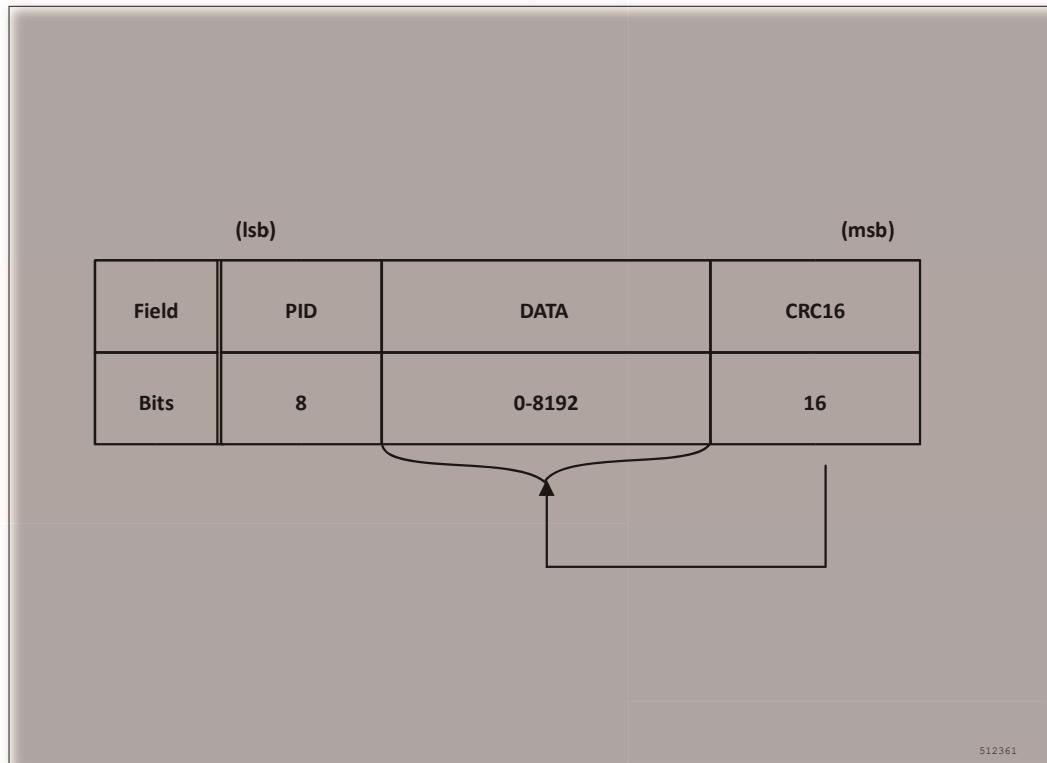


Figure 257. Basic format of a packet

One reception or transmission process of data in the USB is referred as Transaction. The transaction types include IN Transaction, OUT transaction, SETUP transaction etc.

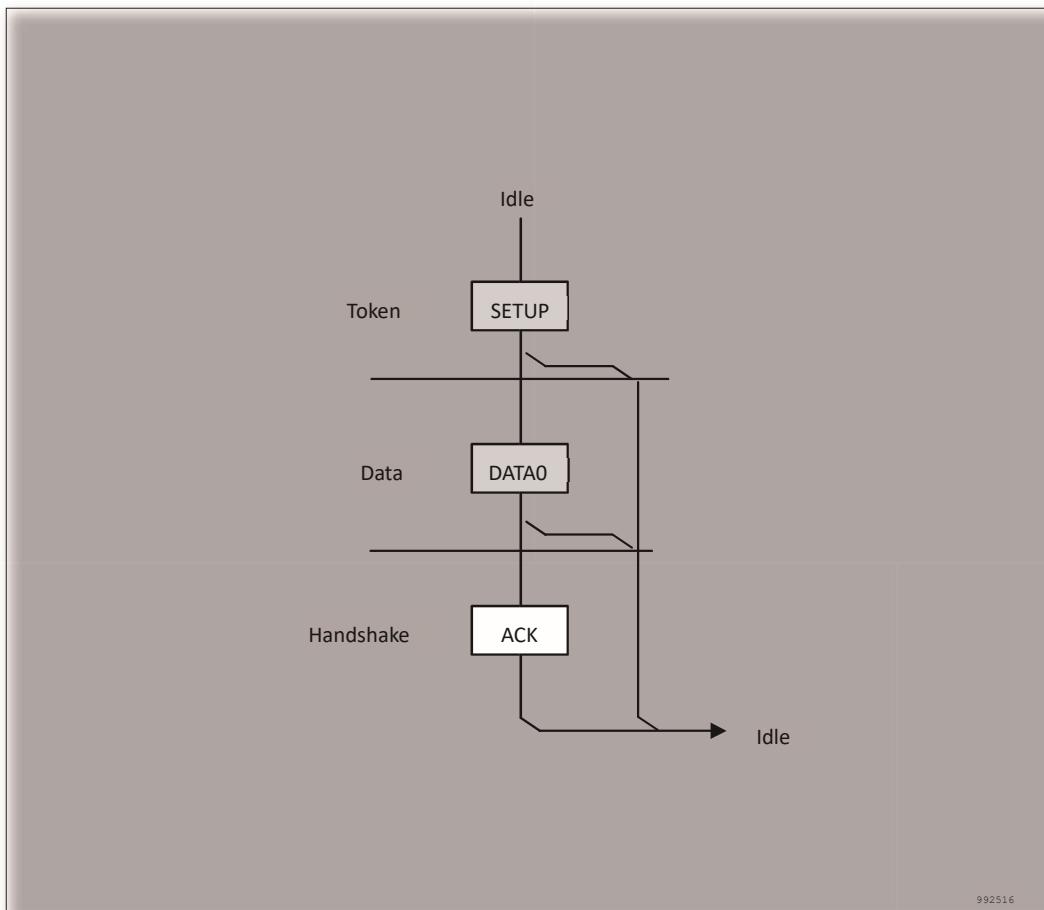


Figure 258. USB transaction

The following figure describes a complete transfer process of one endpoint. For bulk/control transfer, one transfer should start after the previous transfer is completed.

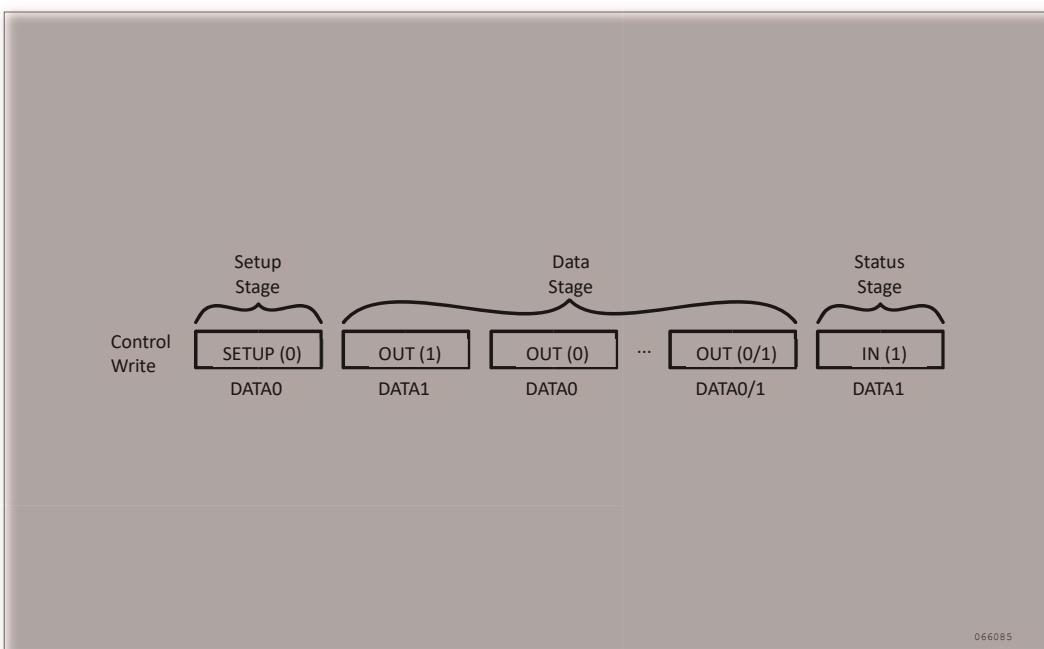


Figure 259. USB transfer

When the USB controller starts to work, the USB stays in the IDLE state and waits for a bus command. As soon as a bus command such as Reset or Configuration is received, an interrupt is generated. CPU will query the type of this command. For example, if it is a reset command, the CPU will clear and reset all programmable statuses. If it is a transfer request command, the CPU will write to/read from the FIFO in the USB controller. With regard to bulk transfer or control transfer as inputs, after the CPU or DMA prepares the data, the CPU will send an ACK handshake signal or STALL handshake signal to put an end to the transfer. With regard to bulk transfer as outputs, after the data is put in the corresponding FIFO, the USB will send an ACK signal automatically.

During the transfer process, if the data size exceeds the maximum packet size, the data will be split into more than one packet to transfer. Otherwise, only one packet will be transferred.

#### 26.4.2 USB enumeration

The Host sends various requests to the device via endpoint 0 in a controlled transfer (Control Transfer). After the device received these requests, it restores relevant information and implements the enumeration.

After the system power-on reset, the USB controller is configured initially:

1. Set the endpoint enable bit
2. Enable the reset and endpoint interrupt
3. Enable the connect bit (CONNECT bit in the USB\_TOP register)

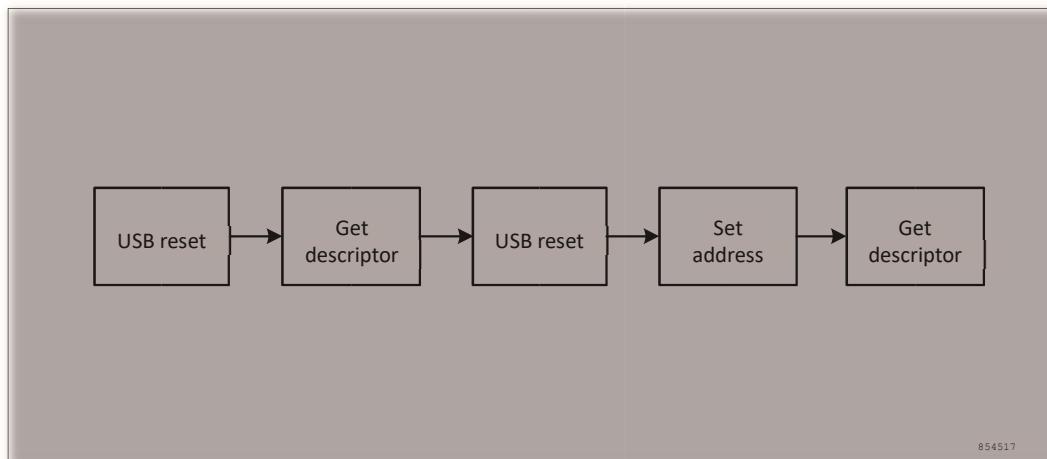


Figure 260. Enumeration process

When the endpoint 0 receives the SETUP packet, the system will read 8 configured bytes and determines the next step. As for the SetAddress descriptor, the USB controller will automatically load the new address.

The first-round process in the enumeration is analyzed below:

After detecting the device, the host initiates a bus reset. It is different from the USB power-on reset or the system reset. Instead, it is a reset that is conveyed to the user by SIE according to the bus status. The device generates a reset interrupt and the device firmware program decides how to handle it.

The host initiates the first control transfer:

1. Host SETUP packet (sending to address 0 endpoint 0), host data packet (request device descriptor), device handshake packet ACK.

The device generates an endpoint 0 data output interrupt and the firmware program should make preparation according to the host requirements in the data packet. In this case, it should prepare the device descriptor in the endpoint 0 input buffer.

2. In the data process, the host will first send an IN token packet, then the device sends a data packet (after SIE receives the IN token, it will send the already prepared data directly to the bus, which is not intervened by the user at that time), and then the host sends an ACK packet.

At this moment, the SIE generates an endpoint 0 data input interrupt, which means the host has taken the data prepared by the device. Then the user can operate in this interrupt response routine. (SIE, referring to serial interface engine, is an internal core for all USB controllers. The SIE is responsible for handling the underlying protocol, such as stuff bit, CRC generation and checking, and it can provide an error report. The main task of SIE is to convert a low-level signal to bytes for the controller to use)

At this point, the host only receives data once, with at least 8 bytes. If the user data is not completely transmitted and more data is prepared in the control endpoint input buffer, the host will ignore it.

3. Status process: The host sends an OUT packet first (notifying the device to output), and then a 0-byte status packet (this 0-byte packet means the device descriptor is received by the host). Then the device sends a handshake ACK packet.

At the moment, the device will not generate an endpoint 0 data output interrupt and there is no data.

The second-round process in the enumeration is setting the address.

When the first-round process is completed successfully, the host will reset the bus again. Here is the address setting phase of control transfer.

1. Host SETUP packet (sending to address 0 endpoint 0), host data packet (request device address), device handshake packet ACK. Therefore, the SETUP packet is always followed by a packet demonstrating the SETUP purpose (GET or SET).

The device generates an endpoint 0 data output interrupt and the firmware program should make preparation according to the host requirements in the data packet. In this case, it should write the address sent from the host to its address control register.

2. Data process: there is no data in this transfer.

3. Status process: The host sends an IN packet (notifying the device to return data), the device sends a 0-byte status packet (showing the address setting is successful), and then the host sends a handshake ACK packet (the address setting has taken effect).

At the moment, the device will not generate an endpoint 0 data input interrupt and there is no data.

The third-round process in the enumeration is the host used the new address to obtain a complete device descriptor.

The host initiates the first control transfer by using the new address:

1. Host SETUP packet (sending to the new address endpoint 0), host data packet (request device descriptor), device handshake packet ACK.

The device generates an endpoint 0 data output interrupt and the firmware program should make preparation according to the host requirements in the data packet. In this case, it should prepare the device descriptor in the endpoint 0 input buffer.

2. In the data process, the host will first send an IN token packet, then the device sends a data packet (after SIE receives the IN token, it will send the already prepared data directly to

the bus, which is not intervened by the user at that time), and then the host sends an ACK packet.

At this moment, the SIE generates an endpoint 0 data input interrupt, which means the host has taken the data prepared by the device. Then the user can operate in this interrupt response routine as below: If the descriptor is not sent completely at one time, the endpoint 0 input buffer should be filled with the remaining content.

Second data transfer: The host sends an IN token again, the device sends a data packet, and then the host sends an ACK packet.

At the moment, the device will generate an endpoint 0 data input interrupt again. If all data is sent, then there will be no more operation. Next is the status process.

3. Status process: The host sends an OUT packet first (notifying the device to output), and then a 0-byte status packet (this means the device descriptor is received by the host). Then the device sends a handshake ACK packet.

#### 26.4.3 USB transfer handling

The system should configure a loop with no operation to wait for the request from the USB host. The request from the USB host will set an interrupt bit. The system can jump out of the loop by continuously querying the interrupt bit or entering the ISR. While detecting an interrupt bit, the system jumps to a relevant sub-program to handle it.

Here is a typical flow chart of USB transfer:

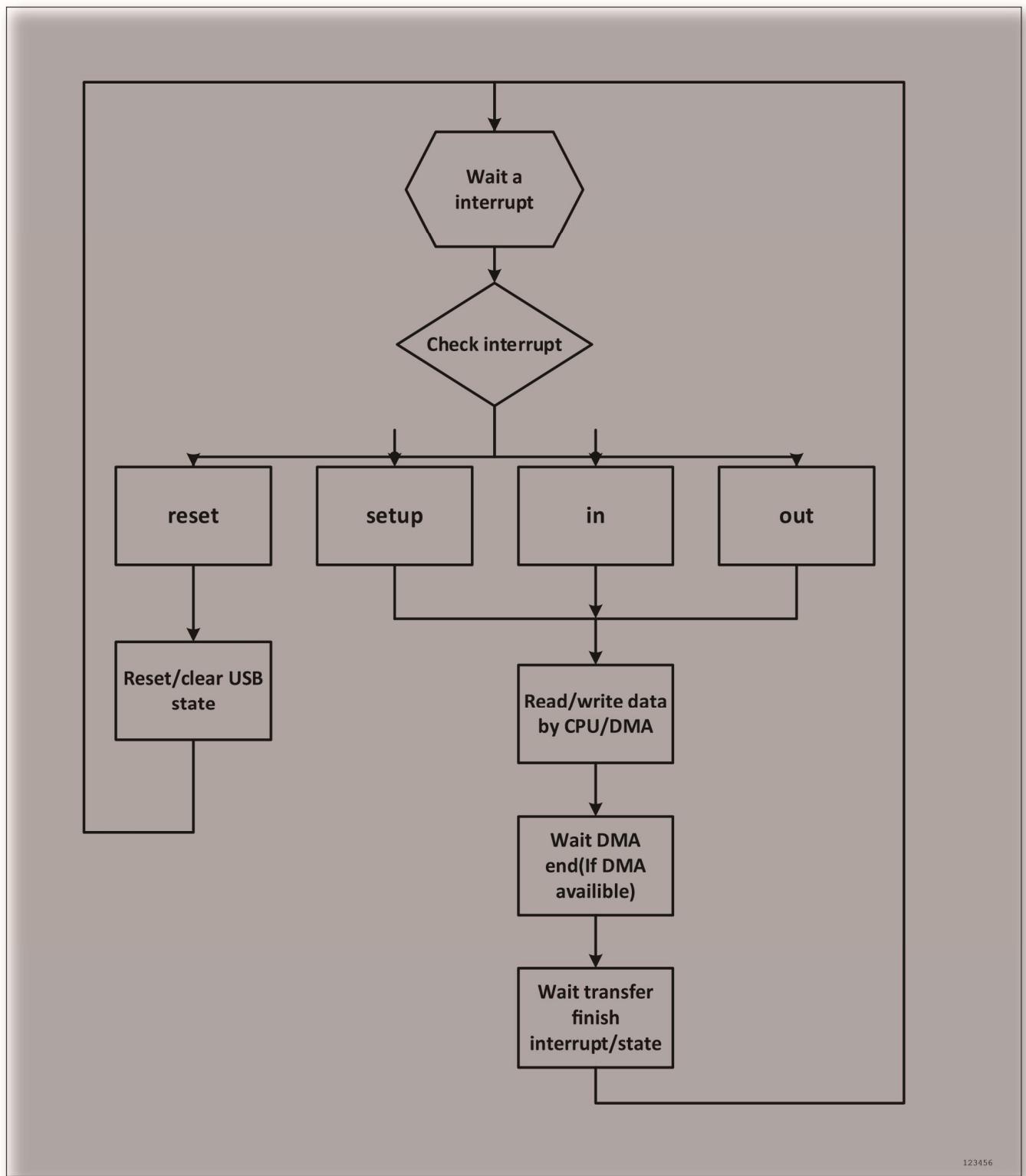


Figure 261. Flow chart of USB transfer

#### 26.4.4 IN token packets

When receiving an IN request from the host, if a NACK is received, the USB host will be re-submitting the IN request until the endpoint confirmation request is valid. If the received address is compliant with a configured endpoint address, follow these steps:

1. If the data in the FIFO is smaller than the size configured in the register EPX\_CTRL[6:0] , or the EPX\_CTRL[7] is not set to 1, the USB module will be sending a NACK automatically until the data is prepared.
2. CPU receives the IN\_NACK status.
3. CPU writes the data to the FIFO.
4. CPU sets the size of the data to be sent and enables the transmission in the EPX\_CTRL register.
5. USB enables the data transfer in the FIFO automatically as soon as it receives the next IN request. After the final data byte is transferred, the calculated CRC is sent automatically.
6. CPU will receive the IN\_ACK status and then the END status after the transfer is completed.
7. If the endpoint is not enabled or the EP\_HALT register is set to suspend, the USB module will response the IN\_STALL instead of sending data.

#### 26.4.5 OUT token packet

When receiving an OUT request from the host, if a NACK is received, the USB host will be re-submitting the OUT request until the endpoint confirmation request is valid. If the received address is compliant with a configured endpoint address, follow these steps:

1. If the FIFO space is smaller than the size of the packet sent from the Host, the USB module will be sending a NACK automatically until CPU fetches the data from the FIFO.
2. CPU receives the OUT\_ACK status.
3. CPU reads data from the FIFO.
4. If the endpoint is not enabled or the EP\_HALT register is set to halt, the USB module will response the OUT\_STALL instead of sending data.

### 26.5 USB register description

Table 79. Overview of USB registers

Offset	Acronym	Register Name	Reset	Section
0x00	USB_TOP	USB TOP register	0x00000000	Section 26.5.1
0x04	USB_INT_STATE	USB interrupt state register	0x00000000	Section 26.5.2
0x08	EP_INT_STATE	USB endpoint interrupt state register	0x00000000	Section 26.5.3
0x0C	EP0_INT_STATE	USB endpoint 0 interrupt state register	0x00000000	Section 26.5.4
0x10	USB_INT_EN	USB interrupt enable register	0x00000000	Section 26.5.5
0x14	EP_INT_EN	USB endpoint interrupt enable register	0x00000000	Section 26.5.6
0x18	EP0_INT_EN	USB endpoint 0 interrupt enable register	0x00000000	Section 26.5.7

Offset	Acronym	Register Name	Reset	Section
0x20~ 0x2C	EPX_INT_STATE	USB endpoint X interrupt state register	0x00000000	Section 26.5.8
0x40~ 0x4C	EPX_INT_EN	USB endpoint X interrupt enable register	0x00000000	Section 26.5.9
0x60	USB_ADDR	USB address register	0x00000000	Section 26.5.10
0x64	EP_EN	USB endpoint enable register	0x00000003	Section 26.5.11
0x68	EP_DMA_DIR	USB endpoint DMA direction register	0x00000001	Section 26.5.12
0x6C	EP_TYPE	USB endpoint type register	0x00000000	Section 26.5.13
0x70	EP_INDEX1_2	USB endpoint 12 index register	0x00000021	Section 26.5.14
0x74	EP_INDEX3_4	USB endpoint 34 index register	0x00000043	Section 26.5.15
0x78	TOG_CTRL1_4	USB data toggle control register	0x00000000	Section 26.5.16
0x7C	TOG_STAT1_4	USB data toggle state register	0x00000000	Section 26.5.17
0x80 ~ 0x9C	SETUPX	USB setup data register	0x00000000	Section 26.5.18
0xA0	PACKET_SIZEL	USB packet size low register	0x00000040	Section 26.5.19
0xA4	PACKET_SIZEH	USB packet size high register	0x00000040	Section 26.5.20
0x100 ~ 0x110	EPX_AVAIL	USB endpoint X available data register	0x00000000	Section 26.5.21
0x120	DMA_ADDR0	USB endpoint 2 DMA address 0 register	0x00000000	Section 26.5.22
0x124	DMA_ADDR1	USB endpoint 2 DMA address 1 register	0x00000000	Section 26.5.23
0x128	DMA_ADDR2	USB endpoint 2 DMA address 2 register	0x00000000	Section 26.5.24
0x12C	DMA_ADDR3	USB endpoint 2 DMA address 3 register	0x00000000	Section 26.5.25
0x130	DMA_NUML	USB endpoint 2 DMA number low register	0x00000000	Section 26.5.26
0x134	DMA_NUMH	USB endpoint 2 DMA number high register	0x00000000	Section 26.5.27
0x140 ~ 0x150	EPX_CTRL	USB endpoint X control register	0x00000000	Section 26.5.28
0x160 ~ 0x170	EPX_FIFO	USB endpoint X FIFO register	0x00000000	Section 26.5.29
0x180	EP_MEM	USB endpoint data memory register	0x00000000	Section 26.5.30
0x184	EP_DMA	USB endpoint DMA enable register	0x00000000	Section 26.5.31
0x188	EP_HALT	USB endpoint halt register	0x00000000	Section 26.5.32
0x1C0	USB_POWER	USB power control register	0x00000027	Section 26.5.33
0x1C4	USB_AHB_DMA	USB AHB DMA register	0x00000070	Section 26.5.34
0x1C8	USB_AHB_RST	USB AHB RST register	0x00000000	Section 26.5.35

### 26.5.1 USB TOP register (USB\_TOP)

Address offset: 0x00

Reset value: 0x0000 0000

## 26.5.2USB interrupt state register (USB\_INT\_STATE)

Address offset: 0x04

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EP INTF	SOFF	RESUM F	SUS PENDF	RSTF	
										r	rc_w1	rc_w1	rc_w1	rc_w1	
15:5	Reserved														always read as 0.
4	EPINTF	r					0x00								EP interrupt flag bit (EP interrupt received)
															This bit is set to '1' if any EP generates an interrupt. For details, refer to the EP_INT_STATE description. This bit is read-only.

3	SOFF	rc_w1	0x00	SOF detection flag bit (BUS received) When the SOF is detected, this bit is set to '1'. It is cleared by writing it to '1'.
2	RESUMF	rc_w1	0x00	Wakeup flag bit (BUS resume received) When the USB bus is activated, this bit is set to '1'. It is cleared by writing it to '1'.
1	SUSPENDF	rc_w1	0x00	USB bus suspend flag bit (BUS suspend received) When the bus suspend state is detected, this bit is set to '1'. It is cleared by writing it to '1'.
0	RSTF	rc_w1	0x00	USB bus reset request flag bit (BUS reset received) When the reset signal of the bus is detected, this bit is set to '1'. It is cleared by writing it to '1'.

**26.5.3 USB endpoint interrupt state register (EP\_INT\_STATE)**

Address offset: 0x08

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												EP4F	EP3F	EP2F	EP1F	EP0F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bit	Field	Type	Reset	Description
15:7	Reserved			always read as 0.
4	EP4F	r	0x00	EP4 interrupt flag bit (EP4 interrupt received)
3	EP3F	r	0x00	EP3 interrupt flag bit (EP3 interrupt received)
2	EP2F	r	0x00	EP2 interrupt flag bit (EP2 interrupt received)
1	EP1F	r	0x00	EP1 interrupt flag bit (EP1 interrupt received)
0	EP0F	r	0x00	EP0 interrupt flag bit (EP0 interrupt received)

**26.5.4 USB endpoint 0 interrupt state register (EP0\_INT\_STATE)**

Address offset: 0x0C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved												OUT-STALLF	OUT-ACKF	OUT-NACKF	IN-STALLF	IN-ACKF	IN-NACKF
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	END	SET UP				

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0.

7	OUT-STALLF	rc_w1	0x00	OUT packet response STALL flag (OUT-STALL received) After the EP receives an OUT packet from the host, the controller responds with a STALL. If the register EP_HALTI[0] is set to '1' and the EP0_CTRL[7] is enabled, the USB controller will respond with a STALL. It is cleared by writing it to '1'.
6	OUT-ACKF	rc_w1	0x00	OUT packet response ACK flag (OUT-ACK received) If the EP0 receives an OUT packet from the host and the FIFO has enough space, then the host will complete the current transfer. The USB controller will response with an ACK automatically. It is cleared by writing it to '1'.
5	OUT-NACKF	rc_w1	0x00	OUT packet response NACK flag (OUT-NACK received) If the EP receives an OUT packet from the host and there is not enough space to store the data from the host, the USB controller will response with a NACK automatically. It is cleared by writing it to '1'.
4	IN-STALLF	rc_w1	0x00	IN packet response STALL flag (IN-STALL received) After the EP receives an IN packet from the host, the controller responds with a STALL. If the register EP_HALTI[0] is set to '1' and the EP0_CTRL[7] is enabled, the the USB controller will respond with a STALL. It is cleared by writing it to '1'.
3	IN-ACKF	rc_w1	0x00	IN packet response ACK flag (IN-ACK received) If the EP receives an IN packet from the host and the FIFO has enough data, the host will complete the current transfer. The USB controller will response with an ACK automatically. It is cleared by writing it to '1'. This bit is set if the EP receives an IN packet from the host and then the host completes the current transfer.

Bit	Field	Type	Reset	Description
2	IN-NACKF	rc_w1	0x00	IN packet response NACK flag (IN-NACK received) If the EP receives an IN packet from the host and the FIFO does not have enough data to complete the current transfer, the USB controller will response with a NACK automatically. It is cleared by writing it to '1'.
1	END	rc_w1	0x00	Transfer complete flag (Status stage finished) This bit is set to '1' after the EP transfer is completed. It is cleared by writing it to '1'.

0	SETUP	rc_w1	0x00	SETUP packet flag received After this bit is set, the content of the SETUP packet can be read from the register SETUP0 ~ 7. It is cleared by writing it to '1'.
---	-------	-------	------	---

### **26.5.5USB interrupt enable register (USB\_INT\_EN)**

Address offset: 0x10

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								INT MASK	Reserved	EPIE	SOF IE	RESUM IE	SUSPENDIE	RST IE	

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0.
7	INTMASK	rw	0x00	<p>Interrupt mask bit</p> <p>1: The interrupt flag bit of every interrupt register is controlled by the corresponding interrupt enable flag bit in every interrupt enable register.</p> <p>0: The interrupt flag bit of every interrupt register is not controlled by the corresponding interrupt enable flag bit in every interrupt enable register.</p>
6:5	Reserved			always read as 0
4	EPIE	rw	0x00	EP interrupt enable bit
3	SOFIE	rw	0x00	SOF detection interrupt enable bit
2	RESUMIE	rw	0x00	BUS resume interrupt enable bit
1	SUSPENDIE	rw	0x00	USB BUS suspend interrupt enable bit
0	RSTIE	rw	0x00	USB BUS reset interrupt enable bit

### 26.5.6USB endpoint interrupt enable register (EP INT EN)

Address offset: 0x14

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							Reserved				EP4IE	EP3IE	EP2IE	EP1IE	EP0IE

Bit	Field	Type	Reset	Description
15:5	Reserved			always read as 0.
4	EP4IE	rw	0x00	EP4 interrupt enable bit

3	EP3IE	rw	0x00	EP3 interrupt enable bit
2	EP2IE	rw	0x00	EP2 interrupt enable bit
1	EP1IE	rw	0x00	EP1 interrupt enable bit
0	EP0IE	rw	0x00	EP0 interrupt enable bit

### 26.5.7 USB endpoint 0 interrupt enable register (EP0\_INT\_EN)

Address offset: 0x18

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
								Reserved	OUT-STALLIE	OUT-ACKIE	OUT-NACKIE	IN-STALLIE	IN-ACKIE	IN-NACKIE	ENDIE	SETUPIE

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0.
7	OUT-STALLIE	rw	0x00	OUT packet response STALL interrupt enable bit (OUT-STALL interrupt enable)
6	OUT-ACKIE	rw	0x00	OUT packet response ACK interrupt enable bit (OUT-ACK interrupt enable)
5	OUT-NACKIE	rw	0x00	OUT packet response NACK interrupt enable bit (OUT-NACK interrupt enable)
4	IN-STALLIE	rw	0x00	IN packet response STALL interrupt enable bit (IN-STALL interrupt enable)

Bit	Field	Type	Type	Description
3	IN-ACKIE	rw	0x00	IN packet response ACK interrupt enable bit (IN-ACK interrupt enable)
2	IN-NACKIE	rw	0x00	IN packet response NACK interrupt enable bit (IN-NACK interrupt enable)
1	ENDIE	rw	0x00	Transfer complete interrupt enable bit (Status stage finished interrupt enable)
0	SETUPIE	rw	0x00	SETUP packet received interrupt enable bit (SETUP packet interrupt enable)

**26.5.8 USB endpoint X interrupt state register (EPX\_INT\_STATE)(X=1~4)**

Address offset: 0x20~ 0x2C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								OUT-STALLF	OUT-ACKF	OUT-NACKF	IN-STALLF	IN-ACKF	IN-NACKF	END	Res.
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0.
7	OUT-STALLF	rc_w1	0x00	<p>OUT packet response STALL flag (OUT-STALL received)  After the EP receives an OUT packet from the host, the controller responds with a STALL. If the register EP_HALTx[x] is set to '1' and the EPx_CTRL[7] is enabled, the USB controller will respond with a STALL.  It is cleared by writing it to '1'.</p>
6	OUT-ACKF	rc_w1	0x00	<p>OUT packet response ACK flag (OUT-ACK received)  If the EP x receives an OUT packet from the host and the FIFO has enough data, the host will complete the current transfer. The USB controller will response with an ACK automatically.  It is cleared by writing it to '1'.</p>
5	OUT-NACKF	rc_w1	0x00	<p>OUT packet response NACK flag (OUT-NACK received)  If the EP receives an OUT packet from the host and there is not enough space to store the data sent from the host, the USB controller will response with a NACK automatically.  It is cleared by writing it to '1'.</p>
4	IN-STALLF	rc_w1	0x00	<p>IN packet response STALL flag (IN-STALL received)  After the EP receives an IN packet from the host, the controller responds with a STALL. If the register EP_HALTx[x] is set to '1' and the EPx_CTRL[7] is enabled, the USB controller will respond with a STALL.  It is cleared by writing it to '1'.</p>
3	IN-ACKF	rc_w1	0x00	<p>IN packet response ACK flag (IN-ACK received)  If the EP receives an IN packet from the host and the FIFO has enough data, then the host will complete the current transfer. The USB controller will response with an ACK automatically.  It is cleared by writing it to '1'.  This bit is set if the EP receives an IN packet from the host and then the host completes the current transfer.</p>
Bit	Field	Type	Reset	Description

2	IN-NACKF	rc_w1	0x00	IN packet response NACK flag (IN-NACK received) If the EP receives an IN packet from the host and there is not enough data to complete the current transfer, the USB controller will response with a NACK automatically. It is cleared by writing it to '1'.
1	END	rc_w1	0x00	Transfer complete flag (Status stage finished) This bit is set to '1' after the EP transfer is completed. It is cleared by writing it to '1'.
0	Reserved			always read as 0

### 26.5.9 USB endpoint X interrupt enable register (EPX\_INT\_EN) (X = 1 ~ 4)

Address offset: 0x40~ 0x4C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
								Reserved	OUT-STALLIE	OUT-ACKIE	OUT-NACKIE	IN-STALLIE	IN-ACKIE	IN-NACKIE	ENDIE	Res.

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0.
7	OUT-STALLIE	rw	0x00	OUT packet response STALL interrupt enable bit (OUT-STALL interrupt enable)
6	OUT-ACKIE	rw	0x00	OUT packet response ACK interrupt enable bit (OUT-ACK interrupt enable)
5	OUT-NACKIE	rw	0x00	OUT packet response NACK interrupt enable bit (OUT-NACK interrupt enable)
4	IN-STALLIE	rw	0x00	IN packet response STALL interrupt enable bit (IN-STALL interrupt enable)
3	IN-ACKIE	rw	0x00	IN packet response ACK interrupt enable bit (IN-ACK interrupt enable)
2	IN-NACKIE	rw	0x00	IN packet response NACK interrupt enable bit (IN-NACK interrupt enable)
1	ENDIE	rw	0x00	Transfer complete interrupt enable bit (Status stage finished interrupt enable)
0	Reserved			always read as 0

### 26.5.10 USB address register (USB\_ADDR)

Address offset: 0x60

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	ADDR	rw	rw	rw	rw	rw	rw
----------	------	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
15:7	Reserved			always read as 0.
6:0	ADDR	rw	0x00	USB address After receiving the address setting descriptor sent from the host, the hardware will load the address to this register automatically. After receiving the bus reset command, the hardware will clear this register automatically.

### 26.5.11 USB endpoint enable register (EP\_EN)

Address offset: 0x64

Reset value: 0x0000 0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EP4EN	EP3EN	EP2EN	EP1EN	EP0EN	
										rw		rw		rw	

Bit	Field	Type	Reset	Description
15:5	Reserved			always read as 0.
4	EP4EN	rw	0x00	Enable End Point 4
3	EP3EN	rw	0x00	Enable End Point 3
2	EP2EN	rw	0x00	Enable End Point 2
1	EP1EN	rw	0x00	Enable End Point 1
0	EP0EN	rw	0x00	Enable End Point 0

### 26.5.12 USB endpoint DMA direction register (EP\_DMA\_DIR)

Address offset: 0x68

Reset value: 0x0000 0001

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										DMA_DIR4	DMA_DIR3	DMA_DIR2	DMA_DIR1		
										rw		rw		rw	

Bit	Field	Type	Reset	Description
15:4	Reserved			always read as 0.
3	DIR4	rw	0x00	Endpoint 4 DMA Direction 0: Write (Host to Device) 1: Read (Device to Host)
2	DIR3	rw	0x00	Endpoint 3 DMA Direction 0: Write (Host to Device) 1: Read (Device to Host)
1	DIR2	rw	0x00	Endpoint 2 DMA Direction 0: Write (Host to Device) 1: Read (Device to Host)
0	DIR1	rw	0x01	Endpoint 1 DMA Direction 0: Write (Host to Device) 1: Read (Device to Host)

### 26.5.13 USB endpoint type register (EP\_TYPE)

Address offset: 0x6C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EP4_TYPE	EP3_TYPE	EP2_TYPE	EP1_TYPE		
										rw	rw	rw	rw		

Bit	Field	Type	Reset	Description
15:4	Reserved			always read as 0
3	EP4_TYPE	rw	0x00	Endpoint 4 type

Bit	Field	Type	Reset	Description
2	EP3_TYPE	rw	0x00	Endpoint 3 type
1	EP2_TYPE	rw	0x00	Endpoint 2 type
0	EP1_TYPE	rw	0x00	Endpoint 1 type 0: No effect 1: Effective

**26.5.14 USB endpoint 12 index register (EP\_INDEX1\_2)**

Address offset: 0x70

Reset value: 0x0000 0021

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								INDEX2				INDEX1			
								rw	rw	r	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:4	INDEX2	rw	0x02	Endpoint 2 index
3:0	INDEX1	rw	0x01	Endpoint 1 index, the EP number received by the USB host

**26.5.15 USB endpoint 34 index register (EP\_INDEX3\_4)**

Address offset: 0x74

Reset value: 0x0000 0043

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								INDEX4				INDEX3			
								rw	rw	r	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:4	INDEX4	rw	0x04	Endpoint 4 index
3:0	INDEX3	rw	0x03	Endpoint 3 index, the EP number received by the USB host

**26.5.16 USB data toggle control register (TOG\_CTRL1\_4)**

Address offset: 0x78

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DTOG 4EN	DTOG 4	DTOG 3EN	DTOG 3	DTOG 2EN	DTOG 2	DTOG 1EN	DTOG 1
								W	RW	W	RW	W	RW	W	RW

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0.
7	DTOG4EN	w	0x00	EP 4 data toggle enable bit (Set End Point 4 enable) 0: do not change the EP 4 data toggle bit 1: set bit 6 DTOG4 as the EP 4 data toggle bit
6	DTOG4	rw	0x00	EP 4 data toggle bit (Set End Point 4 Toggle) 0: DATA0 1: DATA1
5	DTOG3EN	w	0x00	EP 3 data toggle enable bit (Set End Point 3 enable) 0: do not change the EP 3 data toggle bit 1: set bit 4 DTOG3 as the EP 3 data toggle bit
4	DTOG3	rw	0x00	EP 3 data toggle bit (Set End Point 3 Toggle) 0: DATA0 1: DATA1
3	DTOG2EN	w	0x00	EP 2 data toggle enable bit (Set End Point 2 enable) 0: do not change the EP 2 data toggle bit 1: set bit 2 DTOG2 as the EP 2 data toggle bit
2	DTOG2	rw	0x00	EP 2 data toggle bit (Set End Point 2 Toggle) 0: DATA0 1: DATA1
1	DTOG1EN	w	0x00	EP 1 data toggle enable bit (Set End Point 1 enable) 0: do not change the EP 1 data toggle bit 1: set bit 0 DTOG1 as the EP 1 data toggle bit
0	DTOG1	rw	0x00	EP 1 data toggle bit (Set End Point 1 Toggle) 0: DATA0 1: DATA1

**26.5.17 USB data toggle state register (TOG\_STAT1\_4)**

Address offset: 0x7C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Reserved			OUT_TOG4	IN_TOG4	OUT_TOG3	IN_TOG3	OUT_TOG2	IN_TOG2	OUT_TOG1	IN_TOG1

r r r r r r r r

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0.
7	OUT_TOG4	r	0x00	EP 4 data toggle OUT state bit (End Point 4 Toggle State) 0: No effect 1: Receive OUT enabled
6	IN_TOG4	r	0x00	EP 4 data toggle IN state bit (End Point 4 Toggle State) 0: No effect 1: Transmit IN enabled
5	OUT_TOG3	r	0x00	EP 3 data toggle OUT state bit (End Point 3 Toggle State) 0: No effect 1: Receive OUT enabled
4	IN_TOG3	r	0x00	EP 3 data toggle IN state bit (End Point 3 Toggle State) 0: No effect 1: Transmit IN enabled
3	OUT_TOG2	r	0x00	EP 2 data toggle OUT state bit (End Point 2 Toggle State) 0: No effect 1: Receive OUT enabled
2	IN_TOG2	r	0x00	EP 2 data toggle IN state bit (End Point 2 Toggle State) 0: No effect 1: Transmit IN enabled
1	OUT_TOG1	r	0x00	EP 1 data toggle OUT state bit (End Point 1 Toggle State) 0: No effect 1: Receive OUT enabled
0	IN_TOG1	r	0x00	EP 1 data toggle IN state bit (End Point 1 Toggle State) 0: No effect 1: Transmit IN enabled

### 26.5.18 USB setup data register (SETUPX) (X = 0 ~ 7)

Address offset: 0x80~ 0x9C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Reserved								SETUPDX						

r r r r r r r i

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	SETUPDX	r	0x00	USB setup data bit (x = 0,1,2,...,7)(Setup Data X) 64-Bit SETUP data, which is set by hardware automatically according to the data sent from the host

### 26.5.19 USB packet size low register (PACKET\_SIZEL)

Address offset: 0xA0

Reset value: 0x0000 0040

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Reserved								SIZE0						
								RW	RW	RW	RW	RW	RW	RW

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	SIZE0	rw	0x40	USB Max transmit packet size register low bit (USB Max Packet Size) Up to 64 bytes can be set.

**26.5.20 USB packet size high register (PACKET\_SIZEH)**

Address offset: 0xA4

Reset value: 0x0000 0040

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Reserved								SIZE1						
								rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	SIZE1	rw	0x40	USB Max transmit packet size register high bit (USB Max Packet Size) Up to 64 bytes can be set.

**26.5.21 USB endpoint X available data register (EPX\_AVAIL)**

Address offset: 0x100~ 0x110

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EPXAVAIL							
								r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	EPXAVAIL	r	0x00	USB EPX FIFO available data number

**26.5.22 USB endpoint 2 DMA address 0 register (DMA\_ADDR0)**

Address offset: 0x120

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	DMA_ADDR0
r	r

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	DMA_ADDR0	rw	0x00	USB endpoint 2 DMA transmit address [7:0]

### 26.5.23 USB endpoint 2 DMA address 1 register (DMA\_ADDR1)

Address offset: 0x124

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	DMA_ADDR1
r	r

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	DMA_ADDR1	rw	0x00	USB endpoint 2 DMA transmit address [15:8]

### 26.5.24 USB endpoint 2 DMA address 2 register (DMA\_ADDR2)

Address offset: 0x128

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	DMA_ADDR2
r	r

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	DMA_ADDR2	rw	0x00	USB endpoint 2 DMA transmit address [23:16]

**26.5.25 USB endpoint 2 DMA address 3 register (DMA\_ADDR3)**

Address offset: 0x12C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	DMA_ADDR1
	r r r r r r r r

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	DMA_ADDR3	rw	0x00	USB endpoint 2 DMA transmit address [31:24]

**26.5.26 USB endpoint 2 DMA number low register (DMA\_NUML)**

Address offset: 0x130

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	DMA_NUML
	rw rw rw rw rw rw rw rw

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	DMA_NUML	rw	0x00	USB endpoint 2 DMA number low bit (EP2 DMA number 0-7)

**26.5.27 USB endpoint 2 DMA number high register (DMA\_NUMH)**

Address offset: 0x134

Reset value: 0x0000 0000

15	14	13	12	11	10	98	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	---	---	---	---	---	---	---	---

Reserved	DMA_NUMH
	rw rw rw rw rw rw rw rw

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	DMA_NUMH	rw	0x00	USB endpoint 2 DMA number high bit (EP2 DMA number 8-15)

### 26.5.28 USB endpoint X control register (EPX\_CTRL)

Address offset: 0x140~ 0x150

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								TRAN EN	TRANCOUNT							
								rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0.
7	TRANEN	rw	0x00	<p>USB EPX transfer enable bit</p> <p>If this bit is set to '1', the EPX will respond with the data number defined in the 6:0 following the IN transfer; else, the EPX will respond with a NACK.</p> <p>If the EPX FIFO does not have enough data, the EPX will still respond with a NACK.</p> <p>If the EPX HALT enable bit is set to '1', the EPX will respond with a STALL automatically.</p> <p>This bit will turn to '0' automatically by the end of the transfer.</p>
6:0	TRANCOUNT	rw	0x00	<p>EPX transfer number (EPX transfer counter)</p> <p>The data number to be transferred from EPX. Data is stored in the FIFO of each endpoint. The maximum transfer number should not exceed the maximum packet size defined in the PACKAGE_SIZE register. The transfer number of the final packet may be smaller than the largest packet, or even be 0 which means an empty packet is transferred.</p>

### 26.5.29 USB endpoint X FIFO register (EPX\_FIFO)

Address offset: 0x160~ 0x170

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EPX_FIFO							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:8	Reserved			always read as 0
7:0	EPX_FIFO	rw	0x00	EPX FIFO data port

### 26.5.30 USB endpoint data memory register (EP\_MEM)

Address offset: 0x180

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												EP_MEM			
												rw rw rw			

Bit	Field	Type	Reset	Description
15:3	Reserved			always read as 0.
2:0	EP_MEM	rw	0x00	Endpoint x data storage address. 0: EP0 data storage address 0x00-0x40 1: EP1 data storage address 0x40-0x80 2: EP2 data storage address 0x80-0xc0 3: EP3 data storage address 0xc0-0x100 4: EP4 data storage address 0x100-0x140

### 26.5.31 USB endpoint DMA enable register (EP\_DMA)

Address offset: 0x184

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												DMA4 EN	DMA3 EN	DMA2 EN	DMA1 EN	DMA0 EN
												rw rw rw rw				

Bit	Field	Type	Reset	Description
15:5	Reserved			always read as 0.
4	DMA4EN	rw	0x00	EP4 DMA enable bit
3	DMA3EN	rw	0x00	EP3 DMA enable bit
2	DMA2EN	rw	0x00	EP2 DMA enable bit
1	DMA1EN	rw	0x00	EP1 DMA enable bit
0	DMA0EN	rw	0x00	EP0 DMA enable bit

Note: The USB controller only supports DMA operation in endpoint 1 and endpoint 2.

### 26.5.32 USB endpoint halt register (EP\_HALT)

Address offset: 0x188

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										HALT4	HALT3	HALT2	HALT1	HALT0	
										rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15:5	Reserved			always read as 0.
4	HALT4	rw	0x00	<p>EP4 halt bit</p> <p>If this bit is set to '1', the device will respond with a STALL automatically following an IN/OUT transfer.</p> <p>When a token packet is received, this bit is cleared by hardware automatically.</p>
3	HALT3	rw	0x00	<p>EP3 halt bit</p> <p>If this bit is set to '1', the device will respond with a STALL automatically following an IN/OUT transfer.</p> <p>When a token packet is received, this bit is cleared by hardware automatically.</p>
2	HALT2	rw	0x00	<p>EP2 halt bit</p> <p>If this bit is set to '1', the device will respond with a STALL automatically following an IN/OUT transfer.</p> <p>When a token packet is received, this bit is cleared by hardware automatically.</p>
1	HALT1	rw	0x00	<p>EP1 halt bit</p> <p>If this bit is set to '1', the device will respond with a STALL automatically following an IN/OUT transfer.</p> <p>When a token packet is received, this bit is cleared by hardware automatically.</p>
0	HALT0	rw	0x00	<p>EP0 halt bit</p> <p>If this bit is set to '1', the device will respond with a STALL automatically following an IN/OUT transfer.</p> <p>When a token packet is received, this bit is cleared by hardware automatically.</p>

### 26.5.33 USB power control register (USB\_POWER)

Address offset: 0x1C0

Reset value: 0x0000 0027

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WKUP	Res.	SUSP	SUSP EN				
												rw	rw	rw	
Bit	Field	Type	Reset	Description											
15:4	Reserved			always read as 0.											
3	WKUP	rw	0x00	Enable controller wake up from suspend state											
				1: wake up											
				0: do not wake up											
2	Reserved			always read as 1											
1	SUSP	rw	0x01	Suspend											
				1: Normal operating mode											
				0: Suspend mode											
0	SUSPEN	rw	0x01	BUS suspend enable bit											
				1: The controller determines directly whether the USB is suspended according to the bit 1 state.											
				0: The controller determines the suspend signal.											

### 26.5.34 USB AHB DMA register (USB\_AHB\_DMA)

Address offset: 0x1C4

Reset value: 0x0000 0070

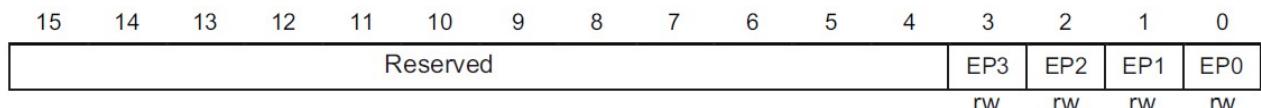
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CH3_BS	CH2_BS	CH1_BS	CH0_BS				
								rw	rw	rw	rw	rw	rw	rw	rw
Bit	Field	Type	Reset	Description											
15:4	Reserved			always read as 0.											
7:6	CH3_BS	rw	0x00	Channel transmit setting (Channel 3 burst set)											
				0: AHB transmits one byte at one time											
				1: AHB transmits 2 bytes at one time											
				2: AHB transmits 4 bytes at one time											
5:4	CH2_BS	rw	0x00	Channel transmit setting (Channel 2 burst set)											
				0: AHB transmits one byte at one time											
				1: AHB transmits 2 bytes at one time											
				2: AHB transmits 4 bytes at one time											
3:2	CH1_BS	rw	0x00	Channel transmit setting (Channel 1 burst set)											
				0: AHB transmits one byte at one time											
				1: AHB transmits 2 bytes at one time											
				2: AHB transmits 4 bytes at one time											

Bit	Field	Type	Reset	Description
1:0	CH0_BS	rw	0x00	Channel transmit setting (Channel 0 burst set) 0: AHB transmits one byte at one time 1: AHB transmits 2 bytes at one time 2: AHB transmits 4 bytes at one time

### 26.5.35 USB AHB RST register (USB\_AHB\_RST)

Address offset: 0x1C8

Reset value: 0x0000 0000



Bit	Field	Type	Reset	Description
15:4	Reserved			always read as 0.
3	EP3	rw	0x00	Endpoint 3 reset
2	EP2	rw	0x00	Endpoint 2 reset
1	EP1	rw	0x00	Endpoint 1 reset
0	EP0	rw	0x00	Endpoint 0 reset

## 27

# Clock Recovery System (CRS)

## Clock Recovery System (CRS)

### 27.1 Introduction

The clock recovery system (CRS) is an advanced digital controller acting on the high-speed internal clock HSI48. It can be used to calibrate the clock. The CRS provides a powerful means for oscillator output frequency evaluation, based on comparison with a selectable synchronization signal. It is capable of doing automatic adjustment of oscillator trimming based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In such case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, which is sent by a USB host at precise 1-ms intervals.

The synchronization signal can also be derived from the LSE clock, from an external pin, or it can be generated by user software.

### 27.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity
  - External pin
  - USB SOF packet reception
- Possibility to generate synchronization pulses by software
- Automatic trimming capability with no need of CPU action
- Manual control option for faster start-up convergence
- 16-bit frequency error counter with automatic error value capture and data loading
- Programmable limit for frequency error value evaluation and status reporting
- Maskable interrupts/events
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

### 27.3 CRS functional description

### 27.3.1 CRS block diagram

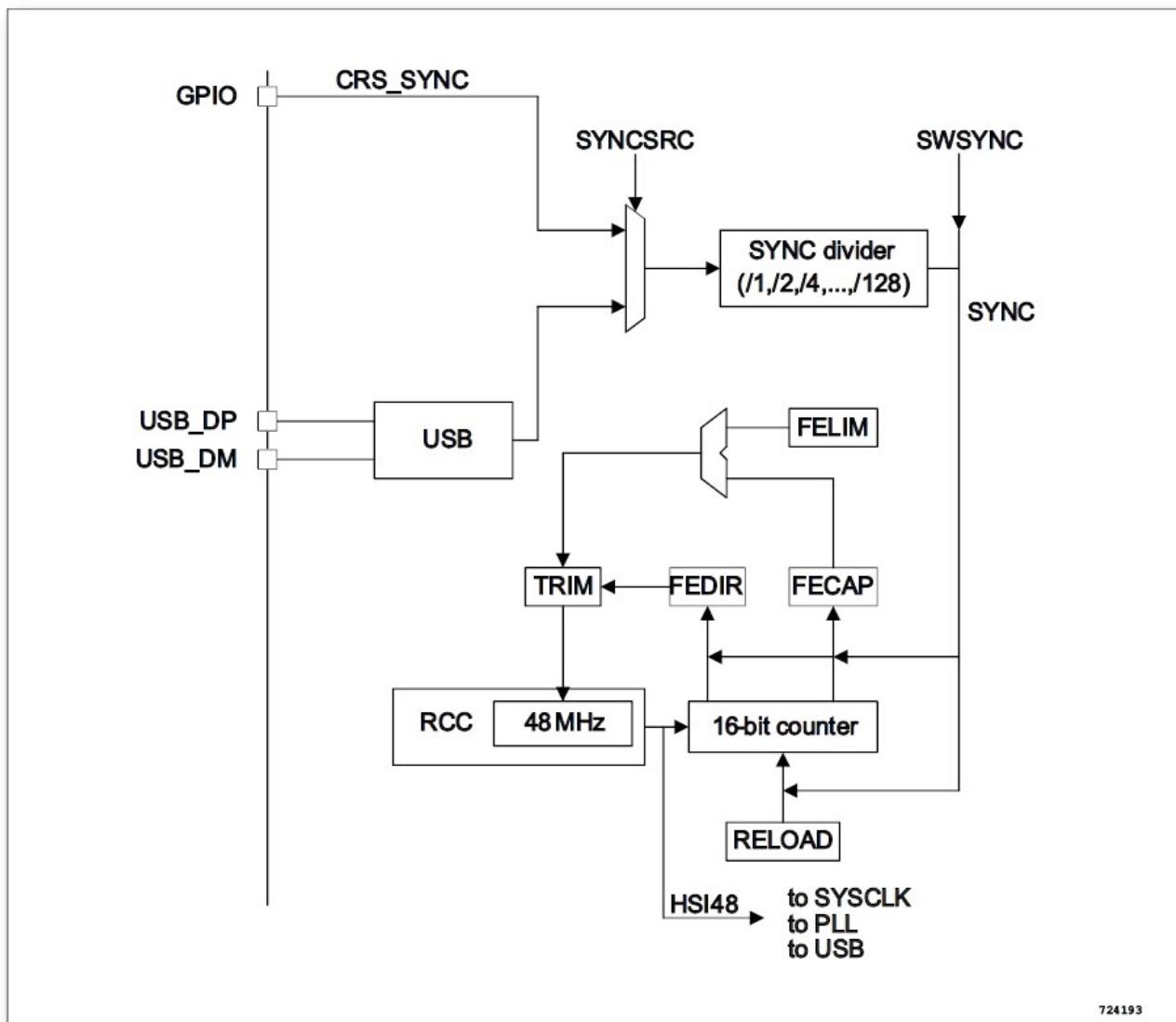


Figure 262. CRS block diagram

### 27.3.2 Synchronization input

The CRS synchronization (SYNC) source, selectable through the CRS\_CFGR register, can be the signal from the external CRS\_SYNC pin or the USB SOF packet. For a better robustness of the SYNC input, a simple 2-level digital filter (sampled by the HSI48 clock) is implemented to filter out any glitches.

This source signal also has a configurable polarity and can then be divided by a programmable prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 KHz).

### 27.3.3 Frequency error measurement

The frequency error counter is a 16-bit down/up counter which is reloaded with the RELOAD value on each event. It starts counting down till it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit where it eventually stops (if no SYNC event is received) and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS\_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP field and in the FEDIR bit of the CRS\_ISR register. When the SYNC event is detected during the downcounting phase (before reaching the zero value), it means that the actual frequency is lower than the target (and so, that the TRIM value should be incremented), while when it is detected during the upcounting phase it means that the actual frequency is higher than the target (and that the TRIM value should be decremented).

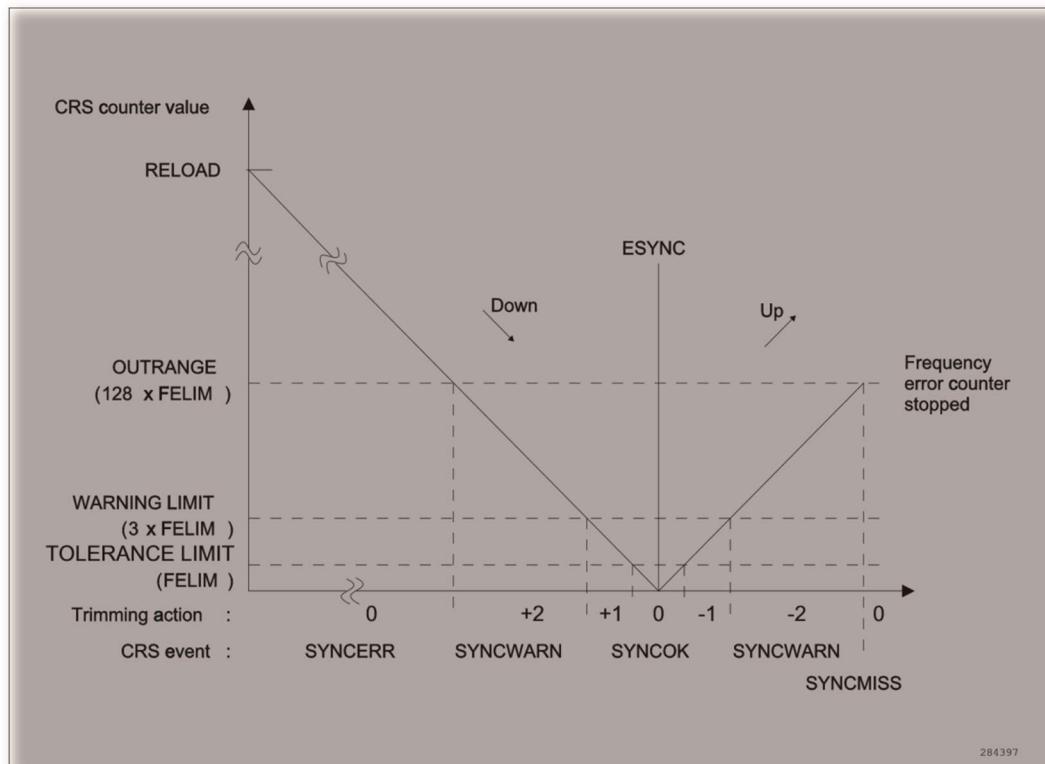


Figure 263. CRS counter behavior

#### 27.3.4 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS\_CFG register
- WARNING LIMIT, defined as  $3 * \text{FELIM}$  value
- OUTRANGE (ERROR LIMIT), defined as  $128 * \text{FELIM}$  value

The result of this comparison is used to generate the status indication and also to control the automatic trimming which is enabled by setting the AUTOTRIM bit in the CRS\_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value is the optimal one and that then, no trimming action is necessary.
  - SYNCOK flag bit is set
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK flag bit is set
  - TRIM value adjusted by one trimming step in AUTOTRIM mode

- When the frequency error is above or equal to the warning limit but below the error limit, it means that a stronger trimming action is necessary, and there is a risk that the optimal TRIM value will not be reached for the next synchronization period.
  - SYNCWARN flag bit is set
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, it means that the frequency error is out of the trimming range. This can also happen when the SYNC input is not clean or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS flag bit is set
  - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is so close to its limits that the automatic trimming would force it to overflow, then the TRIM value is set just to the limit and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIM bit set in the CRS\_CR register), the TRIM field is adjusted by hardware and is read-only.

### 27.3.5 CRS initialization and configuration

#### RELOAD value

The RELOAD value should be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. It is then decreased by one in order to reach the expected synchronization on the zero value. The formula is the following:

$$\text{RELOAD} = (f_{\text{TARGET}}/f_{\text{SYNC}}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48MHz and a synchronization signal frequency of 1 KHz (SOF signal from USB).

#### FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of HSI48 oscillator clock ticks. The following formula can be used:

$$\text{FELIM} = (f_{\text{TARGET}}/f_{\text{SYNC}}) * \text{STEP}[\%]/100\%/2$$

The result should be always rounded up to the nearest integer value in order to obtain the best trimming response. If frequent trimming actions are not wanted in the application, the trimming hysteresis can be increased by increasing slightly the FELIM value.

The reset value of the FELIM field corresponds to  $(f_{\text{TARGET}} / f_{\text{SYNC}}) = 48000$  and to a typical trimming step size of 0.14%.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields which can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than  $128 * \text{FELIM}$  value.

### 27.4 CRS low-power modes

Table 80. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS ISRs cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop or Standby mode is exited and the HSI48 oscillator is restarted.

## 27.5 CRS interrupts

Table 81. Interrupt control bits

Interrupt Event	Event Flag	Enable Control Bit	Clear Control Bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCIE
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOK
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

## 27.6 CRS register description

Table 82. Overview of CRS registers

Offset	Acronym	Register Name	Reset	Section
0x00	CRS_CR	CRS control register	0x00001000	Section 27.6.1
0x04	CRS_CFGR	CRS configuration register	0x2022BB7F	Section 27.6.2
0x08	CRS_ISR	CRS interrupt status register	0x00000000	Section 27.6.3
0x0C	CRS_ICR	CRS interrupt flag clear register	0x00000000	Section 27.6.4

### 27.6.1 CRS control register (CRS\_CR)

Address offset: 0x00

Reset value: 0x0000 1000

31	30	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved															
15	14	12	11	10	9	8	7	6	5	4	3	2	1	0	

Reserved	TRIM				SWSYNC	AUTO TRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNC WARNIE	SYNC OKIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:13	Reserved			always read as 0.
12:8	TRIM	rw	0x10	<p>HSI48 oscillator smooth trimming</p> <p>These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI48.</p> <p>The default value is 16, which corresponds to the middle of the trimming interval. The trimming step is around 67KHz between two consecutive TRIM steps. A higher TRIM value corresponds to a higher output frequency.</p> <p>When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.</p>
7	SWSYNC	rw	0x00	<p>Generate software SYNC event</p> <p>This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: A software SYNC behavior is triggered (A software SYNC event is generated)</p>
6	AUTOTRIMEN	rw	0x00	<p>Automatic trimming enable</p> <p>This bit calculates TRIM values according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value.</p> <p>0: Automatic trimming disabled, TRIM bits can be adjusted by the user.</p> <p>1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.</p>
5	CEN	rw	0x00	<p>Frequency error counter enable</p> <p>This bit enables the oscillator clock for the frequency error counter.</p> <p>0: Frequency error counter disabled</p> <p>1: Frequency error counter enabled</p> <p>When this bit is set, the CRS_CFG register is write-protected and cannot be modified.</p>
4	Reserved			Reserved and always read as 0.
3	ESYNCIE	rw	0x00	<p>Expected SYNC interrupt enable</p> <p>0: Expected SYNC (ESYNCF) interrupt disabled</p> <p>1: Expected SYNC (ESYNCF) interrupt enabled</p>

2	ERRIE	rw	0x00	Synchronization or trimming error interrupt enable 0: Synchronization or trimming error (ERRF) interrupt disabled 1: Synchronization or trimming error (ERRF) interrupt enabled
Bit	Field	Type	Reset	Description
1	SYNCWARNIE	rw	0x00	SYNC warning interrupt enable 0: SYNC warning (SYNCWARNF) interrupt disabled 1: SYNC warning (SYNCWARNF) interrupt enabled
0	SYNCOKIE	rw	0x00	SYNC event OK interrupt enable 0: SYNC event OK (SYNCOKF) interrupt disabled 1: SYNC event OK (SYNCOKF) interrupt enabled

### 27.6.2 CRS configuration register (CRS\_CFGR)

Address offset: 0x04

Reset value: 0x2022 BB7F

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

Bit	Field	Type	Reset	Description
31	SYNCPOL	rw	0x00	<p>SYNC polarity selection</p> <p>This bit is set and cleared by software to select the input polarity for the SYNC signal source.</p> <p>0: SYNC active on rising edge (default) 1: SYNC active on falling edge</p>
30	Reserved			Reserved and always read as 0.
29: 28	SYNCSRC	rw	0x10	<p>SYNC signal source selection</p> <p>These bits are set and cleared by software to select the SYNC signal source.</p> <p>00: GPIO selected as SYNC signal source 01: Reserved 10: USB SOF selected as SYNC signal source (default) 11: Reserved</p>
27	Reserved			Reserved and always read as 0.
26: 24	SYNCDIV	rw	0x00	<p>SYNC divider</p> <p>These bits are set and cleared by software to control the division factor of the SYNC signal.</p> <p>000: SYNC not divided 001: SYNC divided by 2 010: SYNC divided by 4 011: SYNC divided by 8 100: SYNC divided by 16 101: SYNC divided by 32 110: SYNC divided by 64 111: SYNC divided by 128</p>
23: 16	FELIM	rw	0x22	<p>Frequency error limit</p> <p>FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP bits of the CRS_ISR register.</p>
15: 0	RELOAD	rw	0xBB7F	<p>Counter reload value</p> <p>RELOAD is the value to be loaded in the frequency error counter with each SYNC event.</p>

### 27.6.3 CRS interrupt status register (CRS\_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEDIR	Reserved				TRIM OVF	SYNC MISS	SYNC ERR	Reserved				ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31:16	FECAP	r	0x00	<p>Frequency error capture</p> <p>FECAP is the frequency error counter value latched by the end of each SYNC event.</p>
15	FEDIR	r	0x00	<p>Frequency error direction</p> <p>FEDIR is the counting direction of the frequency error counter latched by the end of each SYNC event. It shows whether the actual frequency is below or above the target.</p> <p>0: Upcounting direction, the actual frequency is above the target.</p> <p>1: Downcounting direction, the actual frequency is below the target.</p>
14: 11	Reserved			Reserved and always read as 0.
10	TRIMOVF	r	0x00	<p>Trimming overflow or underflow</p> <p>This flag is set by hardware when the automatic trimming tries to over- or underflow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.</p> <p>0: No trimming error signalized</p> <p>1: Trimming error signalized</p>
9	SYNCMISS	r	0x00	<p>SYNC missed</p> <p>This flag is set by hardware when the frequency error counter reached value FELIM * 128 and no SYNC was detected, meaning either that a SYNC pulse was missed or that the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, and that some other action should be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC) and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.</p> <p>0: No SYNC missed error signalized</p> <p>1: SYNC missed error signalized</p>

Bit	Field	Type	Reset	Description
8	SYNCERR	r	0x00	<p>SYNC error</p> <p>This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action should be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.</p> <p>0: No SYNC error signalized 1: SYNC error signalized</p>
7: 4	Reserved			Reserved and always read as 0.
3	ESYNCF	r	0x00	<p>Expected SYNC flag</p> <p>This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.</p> <p>0: No expected SYNC signalized 1: Expected SYNC signalized</p>
2	ERRF	r	0x00	<p>Error flag</p> <p>This flag is set by hardware in case of any synchronization or trimming error.</p> <p>It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.</p> <p>0: No synchronization or trimming error signalized 1: Synchronization or trimming error signalized</p>

Bit	Field	Type	Reset	Description

1	SYNCWARNF	r	0x00	<p>SYNC warning flag</p> <p>This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more.</p> <p>An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register.</p> <p>It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.</p> <p>0: No SYNC warning signalized 1: SYNC warning signalized</p>
0	SYNCOKF	r	0x00	<p>SYNC event OK flag</p> <p>This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.</p> <p>0: No SYNC event OK signalized 1: SYNC event OK signalized</p>

#### 27.6.4 CRS interrupt flag clear register (CRS\_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
												ESYNCC	ERRC	SYNC WARNC	SYNC OKC

Bit	Field	Type	Reset	Description
31:4	Reserved			Reserved and always read as 0.

3	ESYNCC	w	0x00	Expected SYNC clear flag Writing 1 to this bit clears the ESYNCNF flag in the CRS_ISR register.
2	ERRC	w	0x00	Error clear flag Writing 1 to this bit clears TRIMOVF, SYNCMISS and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.
1	SYNCWARNC	w	0x00	SYNC warning clear flag Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.
0	SYNCOKC	w	0x00	SYNC event OK clear flag Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

**28**

# System Configuration Controller (SYSCFG)

## System Configuration Controller (SYSCFG)

The devices feature a set of system configuration registers. The main purposes of the registers are the following:

- Remapping DMA trigger sources of TIM16, TIM17, CMS, UART1 and ADC to different DMA channels.
- Managing the external interrupt line connection to the GPIOs.
- Remapping the memory located at the beginning of the code area.
- Configuring the external interrupt pins.

### 28.1 SYSCFG register description

Table 83. Overview of SYSCFG registers

Offset	Acronym	Register Name	Reset	Section
0x00	SYSCFG_CFGR	SYSCFG configuration register	0x0000000X	Section 28.1.1
0x08	SYSCFG_EXTICR1	External interrupt configuration register 1	0x00000000	Section 28.1.2
0x0C	SYSCFG_EXTICR2	External interrupt configuration register 2	0x00000000	Section 28.1.3
0x10	SYSCFG_EXTICR3	External interrupt configuration register 3	0x00000000	Section 28.1.4
0x14	SYSCFG_EXTICR4	External interrupt configuration register 4	0x00000000	Section 28.1.5
0x18	SYSCFG_PADHYS	PAD configuration register	0x00000000	Section 28.1.6

#### 28.1.1 SYSCFG configuration register (SYSCFG\_CFGR)

This register is used for specific configurations of the beginning of memory and DMA requests remap. This register has two control bits that are used to configure the storage area type of memory start address 0x0000 0000. These bits are configured by software to bypass the BOOT selection. After reset, these bits take the value selected by the actual BOOT mode configuration.

Address offset: 0x00

Reset value: 0x0000 000X (X is the control bit selected by the actual BOOT mode configuration)

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

Res.	CSMCH2 _DMA _RMP	CSMCH1 _DMA _RMP	TIM17 _DMA _RMP	TIM16 _DMA _RMP	UART1 _RX _DMA _RMP	UART1 _TX _DMA _RMP	ADC _DMA _RMP	Reserved		PA12 _RERMP	PA11 _RERMP	Res.	MEM_MODE		
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	

Bit	Field	Type	Reset	Description
31:15	Reserved			always read as 0.
14	CSMCH2 _DMA_RMP	rw	0x00	<p>CSMCH2_DMA_RMP: CSMCH2 DMA request remapping bit.</p> <p>This bit is set and cleared by software. It controls the remapping of CSMCH2 DMA channel requests.</p> <p>0: No remap (CSMCH2 requests mapped on DMA channel 3)</p> <p>1: Remap (CSMCH2 requests mapped on DMA channel 5)</p>
13	CSMCH1 _DMA_RMP	rw	0x00	<p>CSMCH1_DMA_RMP: CSMCH1 DMA request remapping bit.</p> <p>This bit is set and cleared by software. It controls the remapping of CSMCH1 DMA channel requests.</p> <p>0: No remap (CSMCH1 requests mapped on DMA channel 2)</p> <p>1: Remap (CSMCH1 requests mapped on DMA channel 4)</p>
12	TIM17_DMA _RMP	rw	0x00	<p>TIM17 DMA request remapping bit</p> <p>This bit is set and cleared by software. It controls the remapping of TIM17 DMA channel requests.</p> <p>0: No remap (TIM17_CH1 and TIM17_UP DMA requests mapped on DMA channel 1)</p> <p>1: Remap (TIM17_CH1 and TIM17_UP DMA requests mapped on DMA channel 2)</p>
11	TIM16_DMA _RMP	rw	0x00	<p>TIM16 DMA request remapping bit</p> <p>This bit is set and cleared by software. It controls the remapping of TIM16 DMA channel requests.</p> <p>0: No remap (TIM16_CH1 and TIM16_UP DMA requests mapped on DMA channel 3)</p> <p>1: Remap (TIM16_CH1 and TIM16_UP DMA requests mapped on DMA channel 4)</p>

Bit	Field	Type	Reset	Description
10	UART1_RX _DMA_RMP	rw	0x00	<p>UART1_RX DMA request remapping bit</p> <p>This bit is set and cleared by software. It controls the remapping of UART1_RX DMA channel requests.</p> <p>0: No remap (UART1_RX requests mapped on DMA channel 3)</p> <p>1: Remap (UART1_RX requests mapped on DMA channel 5)</p>
9	UART1_TX _DMA_RMP	rw	0x00	<p>UART1_TX DMA request remapping bit</p> <p>This bit is set and cleared by software. It controls the remapping of UART1_TX DMA channel requests.</p> <p>0: No remap (UART1_TX requests mapped on DMA channel 2)</p> <p>1: Remap (UART1_TX requests mapped on DMA channel 4)</p>
8	ADC_DMA _RMP	rw	0x00	<p>ADC DMA request remapping bit</p> <p>This bit is set and cleared by software. It controls the remapping of ADC DMA channel requests.</p> <p>0: No remap (ADC DMA requests mapped on DMA channel 1)</p> <p>1: Remap (ADC DMA requests mapped on DMA channel 2)</p>
7:5	Reserved			always read as 0
4	PA12_RMP	rw	0x00	<p>PA12 remapping bit</p> <p>This bit is set and cleared by software. After it is set, PA10 is mapped to PA12.</p> <p>Note: After it is mapped, PAD is controlled by the PA10 alternate function register.</p> <p>0: No mapping (PA12)</p> <p>1: Mapping (PA10)</p>
3	PA11_RMP	rw	0x00	<p>PA11 remapping bit</p> <p>This bit is set and cleared by software. After it is set, PA9 is mapped to PA11.</p> <p>Note: After it is mapped, PAD is controlled by the PA9 alternate function register.</p> <p>0: No mapping (PA11)</p> <p>1: Mapping (PA9)</p>
1:0	MEM_MODE	rw	0x00	<p>Memory mapping selection bits</p> <p>These bits are set and cleared by software. They control the memory internal mapping at address 0x0000 0000. After reset, these bits take on the value selected by the BOOT0 pin configuration and nBOOT1 bit.</p> <p>x0: Main Flash memory mapped at 0x0000 0000</p> <p>01: System Flash memory mapped at 0x0000 0000</p> <p>11: Embedded RAM mapped at 0x0000 0000</p>

### 28.1.2 External interrupt configuration register 1 (SYSCFG\_EXTICR1)

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3				EXTI2				EXTI1				EXTI0			
rw	rw	rw	rw												

Bit	Field	Type	Reset	Description
31:16	Reserved			always read as 0.
15:0	EXTIx	rw	0x00	EXTI x configuration bits (x = 0 to 3) These bits are written and read by software to select the input source for the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin

### 28.1.3 External interrupt configuration register 2 (SYSCFG\_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7				EXTI6				EXTI5				EXTI4			
rw	rw	rw	rw												

Bit	Field	Type	Reset	Description
31:16	Reserved			always read as 0.
15:0	EXTIx	rw	0x00	EXTI x configuration bits (x = 4 to 7) These bits are written and read by software to select the input source for the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin

#### 28.1.4 External interrupt configuration register 3 (SYSCFG\_EXTICR3)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11				EXTI10				EXTI9				EXTI8			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:16	Reserved			always read as 0.
15:0	EXTIx	rw	0x00	EXTI x configuration bits (x = 8 to 11) These bits are written and read by software to select the input source for the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin

#### 28.1.5 External interrupt configuration register 4 (SYSCFG\_EXTICR4)

Address offset: 0x14

Reset value: 0x0000 0000

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

Reserved
----------

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

EXTI15	EXTI14	EXTI13	EXTI12
--------	--------	--------	--------

rW															
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Field	Type	Reset	Description
31:16	Reserved			always read as 0.
15:0	EXTIx	rW	0x00	<p>EXTIx configuration bits (x = 12 to 15)  These bits are written and read by software to select the input source for the EXTIx external interrupt.</p> <p>0000: PA[x] pin  0001: PB[x] pin  0010: PC[x] pin  0011: PD[x] pin</p>

### 28.1.6 PAD configuration register (SYSCFG\_PADHYS)

Address offset: 0x18

Reset value: 0x0000 0000

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

Reserved	I2C1_MODE_SEL
----------	---------------

rW
----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved
----------

Bit	Field	Type	Reset	Description
31:17	Reserved			always read as 0.
16	I2C1_MODE_SEL	rW	0x00	<p>I2C1 port mode selection bit</p> <p>0: Open-drain mode  1: Push-pull mode</p>
15:0	Reserved			always read as 0

**29**

# Device Electronic Signature (Device)

## Device Electronic Signature (Device)

The electronic signature is stored in the system memory area in the Flash memory module, and can be read using the JTAG, SWD or the CPU.

It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match its interface to the characteristics of the microcontroller.

### 29.1 Memory size registers

#### 29.1.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software encryption-decryption algorithm before programming the internal Flash memory
- to activate secure boot processes

The 96-bit unique device identifier provides a reference number which is unique for any microcontroller and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes (8 bits)/half-words (16 bits)/words (32 bits) in different ways.

### 29.2 UID register description

Table 84. Overview of memory size register description

Offset	Acronym	Register Name	Reset	Section
0x00	UID1	Unique ID	0xFFFFFFFF	Section 29.2.1
0x02	UID2	Unique ID	0xFFFFFFFF	Section 29.2.2
0x04	UID3	Unique ID	0xFFFFFFFF	Section 29.2.3
0x08	UID4	Unique ID	0xFFFFFFFF	Section 29.2.4

### 29.2.1 Unique ID (UID1)

Base address: 0xFFFF F7E8

Address offset: 0x00

Read-only and factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID															

Bit	Field	Type	Reset	Description
15:0	U_ID	r		U_ID: 15:0 unique ID bits This field value is also reserved for a future feature.

### 29.2.2 Unique ID (UID2)

Address offset: 0x02

Read-only and factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID															

Bit	Field	Type	Reset	Description
15:0	U_ID	r		U_ID: 31:16 unique ID bits This field value is also reserved for a future feature.

### 29.2.3 Unique ID (UID3)

Address offset: 0x04

Read-only and factory-programmed

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID															

Bit	Field	Type	Reset	Description
31:0	U_ID	r		U_ID: 63:32 unique ID bits This field value is also reserved for a future feature.

### 29.2.4 Unique ID (UID4)

Address offset: 0x08

Read-only and factory-programmed

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

Bit	Field	Type	Reset	Description
31:0	U_ID	r		U_ID: 95:64 unique ID bits This field value is also reserved for a future feature.

# 30

# Debug Support (DBG)

## Debug Support (DBG)

### 30.1 Overview

This series is built around a core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is completed, the core and the peripherals may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the microcontroller.

Supports:

- Serial wire debug port

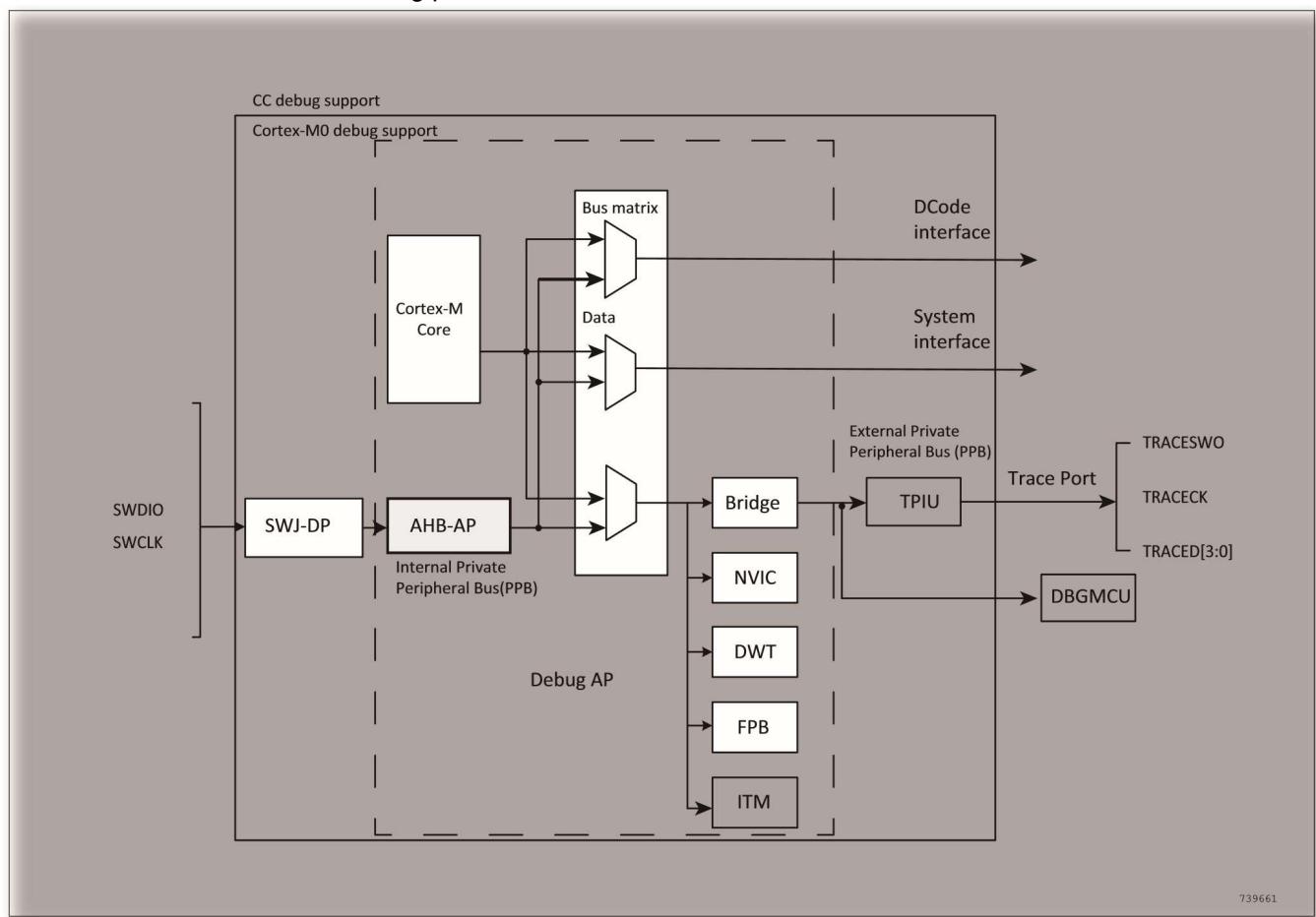


Figure 264. Block diagram of MM32 series and CPU level debug support

The CPU core provides integrated on-chip debug support. It is comprised of:

- SW-DP: serial wire debug port
- AHP-AP: AHB access port

- ITM: instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface

## 30.2 Pinout and debug port pins

This chip is available in various packages with different numbers of available pins. As a result, some functionality related to pin availability may differ between packages.

### 30.2.1 SWD debug port pins

Two general-purpose I/O ports from the chip are used as pins for the SW-DP. These pins are available on all packages.

Table 85. SWJ debug port pins

SWJ-DP Pin Name	SW Debug Port		Pin Assignment
	Type	Debug Assignment	
SW Debug Port	Input/Output	Serial Wire Data Input/Output	PA13
SWCLK	Input	Serial Wire Clock	PA14

### 30.2.2 Internal pull-up and pull-down on SWD pins

It is necessary to ensure that the SWD input pins are not floating since they are directly connected to D triggers to control the debug mode features. Special care must be taken with the SWCLK pin which is directly connected to the clock of some of these D triggers.

To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the SWD input pins.

- SWDIO: internal pull-up
- SWCLK: input pull-down

The software can then use these I/Os as standard GPIOs.

## 30.3 ID codes and locking mechanism

There are several ID codes inside the MCU.

### 30.3.1 MCU device ID code

The MCU integrates an MCU ID code. This ID identifies the MCU part number and the die revision. It is part of the DBG\_MCU component and is mapped on the external APB bus. This code is accessible using the SW debug port (two pins) or by the user software.

DBGMCU\_IDCODE

Address: 0x40013400  
Only 32-bits access supported. Read-only.

Table 86.

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

DEV_ID															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
DEV_ID															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

bits 31: 0	DEV_ID: Device identifier
------------	---------------------------

### 30.3.2 Cortex JEDEC-106 ID code

The CPU integrates a JEDEC-106 ID code. It is located in the 4KB ROM table mapped on the internal APB bus at address 0xE00FF000\_0xE00FFFF.

The table below shows the ID codes of this series:

Table 87. ID codes

ID name	Chip
DEV_ID	0xCC5680C7
CPU TAP SW ID	0x0BB11477

## 30.4 SW debug port

### 30.4.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional data signal

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO is bidirectional, the line must be pulled-up on the board (100 K recommended).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

### 30.4.2 SW protocol sequence

Each sequence consists of three phases:

- Packet request (8 bits) transmitted by the host
- Acknowledge response (3 bits) transmitted by the target
- Data transfer phase (33 bits) transmitted by the host or the target

Table 88. Packet request (8-bits)

Bit	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request 1: Read Request
4:3	A[3:2]	Address field of the DP or AP registers
5	Parity	Parity bit of preceding bits
6	Stop	0
7	Park	Not driven by the host. Must be read as “1” by the target because of the pull-up.

Refer to the CPU Technical Reference Manual for a detailed description of DPACC and APACC registers. The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drives the line.

Table 89. ACK response (3 bits)

Bit	Name	Description
0 .. 2	ACK	001: FAULT 010: WAIT 100: OK

The ACK Response must be followed by a turnaround time only if it is a READ transaction or if a WAIT or FAULT acknowledge has been received.

Table 90. DATA transfer (33 bits)

Bit	Name	Description
0 .. 31	WDATA/RDATA	Write or Read data
32	Parity	Parity bit of the 32 data bits

The DATA transfer must be followed by a turnaround time only if it is a READ transaction.

### 30.4.3 SW-DP state machine (Reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard.

The SW-DP state machine is inactive until the debugger reads this ID code.

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the DP has switched from to SWD or after the line is high for more than 50 cycles.
- The state machine switches to the IDLE STATE if the line is low for at least two cycles when the state machine is in the RESET state.
- After RESET state, it is mandatory to first enter into an IDLE state AND to perform a READ access of the DP-SW ID register. Otherwise, the debugger will receive a FAULT acknowledge response on another transactions.

### 30.4.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the data can be obtained by the debugger immediately (if ACK = OK) or can be delayed (if ACK = WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result.
- The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to notify the debugger if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP and AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the acknowledge response obtained by the debugger is "WAIT". IDCODE read or CTRL/STAT read or ABORT write are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state). This is particularly important when writing the CTRL/STAT for a power-up request. Otherwise, the next transaction (requiring a power-up) occurs immediately and leads to failure.

### 30.4.5 SW-DP registers

Access to these registers are initiated when APnDP = 0. Table 91.

A[3:2]	Read/Write	CTRLSEL bit in the SELECT register	Register	Description
00	Read		IDCODE	Fixed at 0x1BA01477 (identifies the SW-DP).
00	Write		ABORT	
01	Read/Write	0	DP-CTRL /STAT	Request a system or debug power-up; configure the transfer operation for AP accesses; control the compare and verify operations; read some status flags (overrun, power-up acknowledges).

A[3:2]	Read/Write	CTRLSEL bit in the SELECT register	Register	Description
01	Read/Write	1	WIRE CONTROL	Configure the physical serial port protocol (like the duration of the turnaround time).
10	Read		READ RESEND	Enables recovery of the data from a corrupted debugger transfer, without repeating the original AP transfer.
10	Write		SELECT	Select the current access port and the active 4-words register window.
11	Read/Write		READ BUFFER	This register is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction.

### 30.4.6 SW-AP registers

Access to these registers are initiated when APnDP=1.

The access address of AP registers is comprised of two parts:

- The value A[3:2]
- The current value of the DP SELECT register

## 30.5 MCU debug component (MCUDBG)

The MCU debug component helps the debugger provide support for:

- Low-power mode
- Clock control for timers and watchdog during a breakpoint
- Control of the trace pins assignment

### 30.5.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed. The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU. The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior.

- In Sleep mode, DBG\_SLEEP bit of DBGMCU\_CR register must be previously set by the debugger. This will feed HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG\_STOP must be previously set by the debugger. This will enable the internal oscillator clock to feed FCLK and HCLK in STOP mode.

### 30.5.2 Debug support for timers and watchdog

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

### 30.5.3 Debug MCU configuration register

This register allows the configuration of the MCU under DEBUG.

This concerns:

- Low-power mode support
- Timer and watchdog counter support
- Trace pin assignment

This DBGMCU\_CR is mapped on the external APB bus at base address 0x4001 3404. It is asynchronously reset by the PORESET (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support these features, it is still possible for the user software to write to these registers.

## 30.6 DBG register description

Table 92. Overview of DBG registers

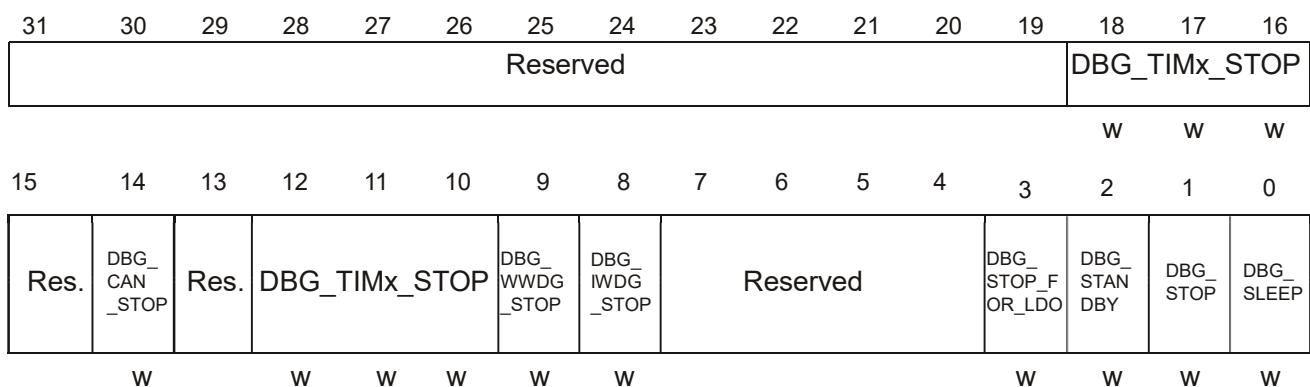
Offset	Acronym	Register Name	Reset	Section
0x00	DBG	DBG control register	0x00000000	Section 30.6.1

### 30.6.1 DBG control register (DBG\_CR)

Address: 0x40013404

Only 32-bits access supported.

POR Reset: 0x0000 0000 (not reset by system reset)



Bit	Field	Type	Reset	Description
31:19	Reserved			always read as 0.
18:16	DBG_TIMx_STOP	w	0x00	TIMx counter stopped when core is halted ( $x = 16, 17, 14$ ) 0: The involved Timer Counter is fed even if the core is halted 1: The involved Timer Counter is stopped when the core is halted
15	Reserved			always read as 0
14	DBG_CAN_STOP	w	0x00	CAN stopped debug when core is halted 0: Same behavior as in normal mode 1: The CAN receive registers are frozen
13	Reserved			always read as 0
12:10	DBG_TIMx_STOP	w	0x00	TIMx counter stopped when core is halted ( $x = 1..3$ ) 0: The involved Timer Counter is fed even if the core is halted 1: The involved Timer Counter is stopped when the core is halted
9	DBG_WWDG_STOP	w	0x00	Debug window watchdog stopped when core is halted 0: The window watchdog counter clock continues even if the core is halted 1: The window watchdog counter clock is stopped when the core is halted
8	DBG_IWDG_STOP	w	0x00	Debug independent watchdog stopped when core is halted 0: The watchdog counter clock continues even if the core is halted 1: The watchdog counter clock is stopped when the core is halted
7:4	Reserved			always read as 0
3	DBG_STOP_FOR_LDO	w	0x00	Debug Stop mode STOP LDO
2	DBG_STAN_DBY	w	0x00	Debug Standby mode 0: (FCLK = Off, HCLK = Off) The whole digital part is unpowered. From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bits indicated that the MCU is exiting Standby) 1: (FCLK= Off, HCLK=Off) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RL oscillator which remains active. In addition, the MCU generates a system reset during Standby mode so that exiting from Standby is identical than fetching from reset.

Bit	Field	Type	Reset	Description
1	DBG_STOP	w	0x00	<p>Debug Stop mode</p> <p>0: (FCLK = Off, HCLK = Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (HSI clock divided by 4 is used as the PLL input clock). Consequently, the software does not have to reprogram the clock controller. When exiting STOP mode, the software must reprogram the clock controller, if HSE clock is used as the PLL input clock.</p> <p>1: (FCLK=Off, HCLK=Off) In this case, when entering STOP mode, FCLK and HCLK are clocked by the internal oscillator. When exiting STOP mode, the software must reprogram the clock controller.</p>
0	DBG_SLEEP	w	0x00	<p>Debug Sleep mode</p> <p>0: (FCLK = Off, HCLK = Off) In Sleep mode, FCLK is clocked by the system clock as previously configured while HCLK is disabled. In Sleep mode, the clock controller configuration is not reset. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.</p> <p>1: (FCLK = On, HCLK = On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured).</p>

# 31

# Revision History

## Revision History

Table 93. Revision history

Date	Revision	Changes
2020/08/11	Rev1.00	Initial release.